Journal of Intelligent Learning Systems and Applications, **2011, 3, 139-154**
doi:10.4236/jilsa.2011.33016 Published Online August 2011 (http://www.scirp.org/journal/jilsa)

Identifying Causes Helps a Tutoring System to Better Adapt to Learners during Training Sessions

Usef Faghihi[1*], Philippe Fournier-Viger[1], Roger Nkambou[1], Pierre Poirier[2]

[1]Department of Computer Science, Université du Québec à Montréal, Montréal, Canada; [2]Cognitive Science Institute, Université du Québec à Montréal, Montréal, Canada.
Email: {*Usef.faghihi, Philippe Fournier-viger}@courrier.uqam.ca, {Roger.Nkambou, Pierre.Poirer}@uqam.ca

Received August 11th, 2010; revised June 7th, 2011; accepted June 14th, 2011.

ABSTRACT

This paper describes a computational model for the implementation of causal learning in cognit ive agents. The Conscious Emotional Learning Tutoring System (CELTS) is able to provide dynamic fine-tuned assistance to users. The integration of a Causal Learning mechanism within CELTS allows CELTS to first establish, through a mix of datamining algorithms, gross user group models. CELTS then uses these models to find the cause of users' mistakes, evaluate their performance, predict their future behavior, and, through a pedagogical knowledge mechanism, decide which tutoring intervention fits best.

Keywords: *Cognitive Agents, Computational Causal Modeling and Learning, Emotions*

1. Introduction

Conscious Tutoring System (CTS, [1]) to which we added Emotions (E) and Learning (L) is a general cognitive architecture designed to be put to work as a Tutor for astronauts learning to manipulate Canadarm2. Canadarm2 is a robotic arm installed on the International Space Station's (ISS). ISS has been designed and implemented to accommodate scientific experiments and life in the space. Thus, it needs to be supplied constantly with foods, fuel, inspections, etc. For instance, astronauts may use Canadarm2 to charge or discharge the received food from the space shuttles. Thus, manipulating Canadarm2 is a difficult task, which requires astronauts to undergo a serious amount of training. The seven degrees of freedom of the Canadarm2 is the first difficulty to overcome, as it considerably increases the number of possible operations. The second difficulty is sight limitation. It is impossible to have an overall view of the station; therefore, theastronaut can only see the arm through a steady climb camera installed on the station and on the Canadarm2. Furthermore, the astronaut must choose among these cameras because there are only three screens [2].

CELTS' emotional mechanism simulates the peripheralcentral theory of emotions [2]. The peripheral-central approach takes into account both the short and long route of information processing and reactions, as in humans. Both the short and long routes perform in a parallel and complementary fashion in CELTS' architecture. The Emotional Mechanism (EM) learns and at the same time contributes emotional valences (positive or negative) to the description of the situation. It also contributes to the decisions made and the learning achieved by the system.

CELTS' Episodic Mechanism (EPM) simulates the multiple-trace theory of memory consolidation [2]. The multiple-trace theory postulates that every time an event causes memory reactivation, a new trace for the activated memory is created in the hippocampus. Memory consolidation occurs through the reoccurring loops of episodic memory traces in the hippocampus and the construction of semantic memory traces in the cortex. Thus, the cortical neurons continue to rely on the hippocampus even after encoding. We used sequential pattern mining algorithms to simulate this behavior of memory consolidation in CELTS. To do so, every informationbroadcasted in the system during a training session between CELTS and learners is assigned to a specific time. Thus, CELTS' EPM extracts information from registered events in the system. Given a problem, EPM is capable of finding the best solution among different solutions conceived by the expert in its BN [2].

One of CELTS' most significant limitations in its current implementation is its incapacity to find out why a learner made a mistake-the causes of the mistake. To address this issue, we propose to implement a Causal Learning mechanism (CLM) in CELTS and combine it with its existing Emotional Learning mechanism (see [3, 4] for more details).

In humans, the process of inductive reasoning stems in part from activity in the left prefrontal cortex and the amygdala; it is a multimodular process [5]. We base our proposed improvements to CELTS' architecture on this same logic. CELTS' modular and distributed organization is ideal for the use of distinct mathematical methods and algorithms that can be tailored to the specific requirements of the Emotional Learning mechanism and the newly integrated CLM. In humans, causal memory is influenced by the information retained by the episodic memory. Inversely, new experiences are influenced by our causal memory [6-9].

Furthermore, causal learning is the process through which we come to infer and memorize an event's reasons or causes based on previous beliefs and current experience that either confirm or invalidate previous beliefs [10]. In the context of CELTS, we refer to Causal Learning as the use of inductive reasoning to generalize causal rules from sets of experiences. CELTS' observes learners' behavior without complete information regarding the reasons for their behavior. Our prediction is that, through inductive reasoning, CELTS can infer the proper set of causal relations from its observations of the learners' behavior. It must be noted that CELTS in its Episodic Learning mechanism uses sequential pattern mining algorithms as a means for deductive reasoning. That is, from the information exchanged between learners and CELTS, the Episodic learning mechanism infers that if a user forgets to adjust the camera before any Canadarm2 movement and chooses a bad joint, he or she will make a collision risk on the ISS.

The goal of CELTS' Causal Learning Mechanism (CLM) is two-fold: 1) is to find causal relations between events during training sessions in order to better assist users; and 2) to implement partial procedural learning in CELTS' Behavior Network (BN)[1], which is based on Maes' Behavior Network [11]. To implement CELTS' CLM, we draw inspiration from Maldonado's work [10] that defines three hierarchical levels of causal learning: 1) the lowest level, responsible for the memorization of task execution; 2) the middle level, responsible for the computation of retrieved information; and 3) the highest level, responsible for the integration of this evidence with previous causal knowledge.

In the present paper, we begin with a brief review of the existing work concerning the implementation of Causal Learning in cognitive agents (Section 2). In section 3, we propose our new architecture combining elements of the Emotional mechanism and Causal Learning, focusing especially on the two-fold aspect of the causal learning mechanism described above. Finally, we present results from our experiments with CELTS.

2. Causal Learning Models and Their Implementation in Cognitive Agents

Scientists propose causal Bayes nets (acyclic graphs) as an alternative approach to establishing causal relation between events. Different methods are proposed for finding causal relations between events such as scientific experiments, statistical relations, temporal order, prior knowledge, and so forth [12].The key issue for the construction of a causal Bayes net is finding conditional probability between events. Mathematics is used to describe conditional and unconditional probabilities between a graph's variables. The structure of a causal graph restricts the conditional and unconditional probabilities between the graph's variables. We can find the restriction between variables using the Causal Markov Assumption (CMA). The CMA suggests that every node in an acyclic graph is conditionally independent of its ascendants, given the node's parents (direct causes). For instance, suppose one observes that each time one forgets to adjust his car's side and front mirrors (M), he tends to have poor control over the wheel (W) and cause collisions (C) with other cars. We can link these variables in the following way: 1) $M \rightarrow W \rightarrow C$; and 2) $W \leftarrow M \rightarrow C$. The first graph shows that the probability of for- getting mirror adjustment is independent of the probability of making a collision with other cars, conditional on the occurrence of poor wheel control. The second graph demonstrates that the probability of poor wheel control is independent of the probability of making a collision with other cars and is conditional on forgetting mirror adjustment. The CMA establishes such separation between nodes to all acyclic graphs' nodes. Thus, knowing a graphs's structure and the values of some of the variables, we are capable of predicting the conditional probability of other variables. Causal Bayes nets are also capable of predicting the consequences of direct external interventions on their nodes. When, for instance, an external intervention occurs on a node (N), it must solely change its value and not affect other node values in the graph except through the node N's influences. In conclusion, one can generate a causal structure from sets of effects and conversely predict sets of effects from a causal structure [13].

To our knowledge, two research groups have attempted to incorporate Causal Learning mechanisms in their cognitive architecture. The first is Schoppek with

[1]CELTS' Behavior Network (BN) (**Figure 4(D)**) is a high-level procedural memory, a network of partial plans that analyses the context to decide what to do, which behavior to set off.

the ACT-R architecture [14], who has not included a role for emotions in his causal learning and retrieval processes. ACT-R constructs the majority of its information according to the I/O knowledge base method. It also uses a sub-symbolic form of knowledge to produce associations between events. As explained by Schoppek [15] causal learning in ACT-R occurs through implicit learning of information. To learn causes in ACT-R, sub-symbolic knowledge applies its influence through activition processes that are inaccessible to production rules. However, the causal model created by Schoppek in ACT-R "overestimates discrimination between old and new states" and every assumption for the creation of a causal model must be detailed by a programmer. The second is Sun [16], who proposed the CLARION architecture. In CLARION's current version, during bottom-up learning, the propositions (premises and actions) are already present in top level (explicit) modules before the learning process starts, and only the links between these nodes emerge from the implicit level (rules). Thus, there is no unsupervised causal learning for the new rules created in CLARION [17].

As it is mentioned above, various causal learning models have been proposed, such as Gopnik's [18]. All proposed models use a Bayesian approach for the construction of knowledge. Bayesian networks work with hidden and non-hidden data and learn with little data. However, Bayesian networks need experts to assign predefined values to variables, and this is often a very difficult and time-consuming task [19]. In the context of a tutoring agent like CELTS, this is a serious issue, because we wish that CELTS could learn and adapt its knowledge of causes automatically without any human intervention. Another problem for Bayesian learning, crucial in the present context, is the risk of combinatory explosion in the case of large amounts of data. In the case of our agent, constant interaction with learners create the large amount of data stored in CELTS modules. For this last reason, we believe that a combination of sequential pattern mining (SPM) algorithms with association rules (AR) is more appropriate to implement a causal learning mechanism in CELTS.

The other advantage of causal learning using the combination of AR and SPM is that CELTS can then learn in a real-time incremental manner—that is, the system can update its information by interacting with various users. A final reason for choosing the combination of AR and SPM is that the aforementioned problem explained by Schoppek, which occurs with ACT-R, cannot occur when using association rules for causal learning. However, it must be noted that although data mining algorithms learn faster than Bayesian networks when all data is available, they have problems with hidden data. Furthermore, like Bayesian learning, there is a need for experts, since the rules found by data mining algorithms must be verified by a domain expert [19].

3. Causal Memory and Causal Learning in CELTS' Architecture

CELTS architecture relies on the functional "consciousness" [20] mechanism for much of its operations. It also bears some functional similarities with the physiology of the nervous system. Its modules communicate with one another by contributing information to its Working Memory through information codelets[2] [21] (see [22] for more details). Before explaining CELTS' Causal Learning, we will describe in the next subsection in detail our causal model for cognitive architectures.

CELTS' causal model takes into account the existence of specific cognitive processes—be they associative or causal.

CELTS' Causal Learning takes place during its cognitive cycles.A cognitive cycle starts by a perception and usually ends with an action. It is conceived as an iterative, cyclical, active process that allows interactions between the different components of the architecture

After the information from the environment reaches CELTS' Perceptual mechanism (**Figure 1**), it is sent to the Working memory (WM). CELTS' WM (**Figure 1**) is monitored by expectation codelets and other types of codelets (see CELTS' emotional mechanism for more details [23]). If expectation codelets observe information coming in WM confirming that the behaviors expected result failed, then the failure brings CELTS' Emotional and Attention mechanisms back to that information. To deal with the failure, emotional codelets that monitor WM first send a portion of emotional valences sufficient to get CELTS' Attention mechanism to select information about the failed result and bring it back to consciousness. The influence of emotional codelets at this point remains for the next cognitive cycles, until CELTS finds a solution or has no remedy for the failure. Since relevant resources need to be recruited to allow CELTS' modules to analyze the cause of the failure and to allow deliberation to take place concerning supplementary and/or alternative actions, the consciousness mechanism broadcasts this information to all modules. Among different modules inspecting the broadcasted information by the consciousness mechanism, the Episodic and Causal Learning mechanisms are also collaborating to find previous sequences of events that occurred before

[2]Based on Hofstadter *et al.*'s idea, a codelet is a very simple agent, "a small piece of code that is specialized for some comparatively simple task". Implementing Baars theory's simple processors, codelets do much of the processing in the architecture. In our case, each information codelet possesses an activation value and an emotional valence specific to each cognitive cycle.

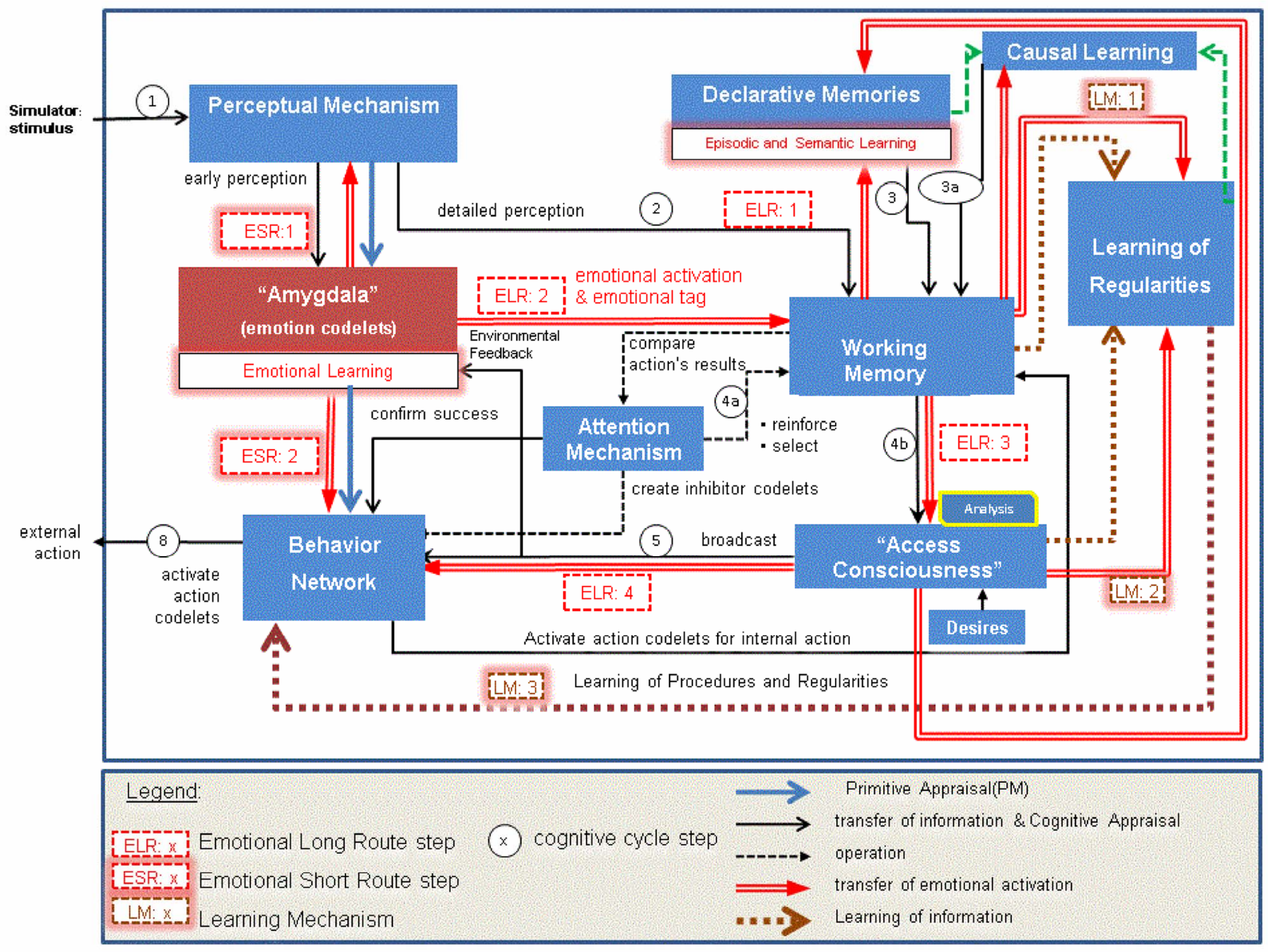
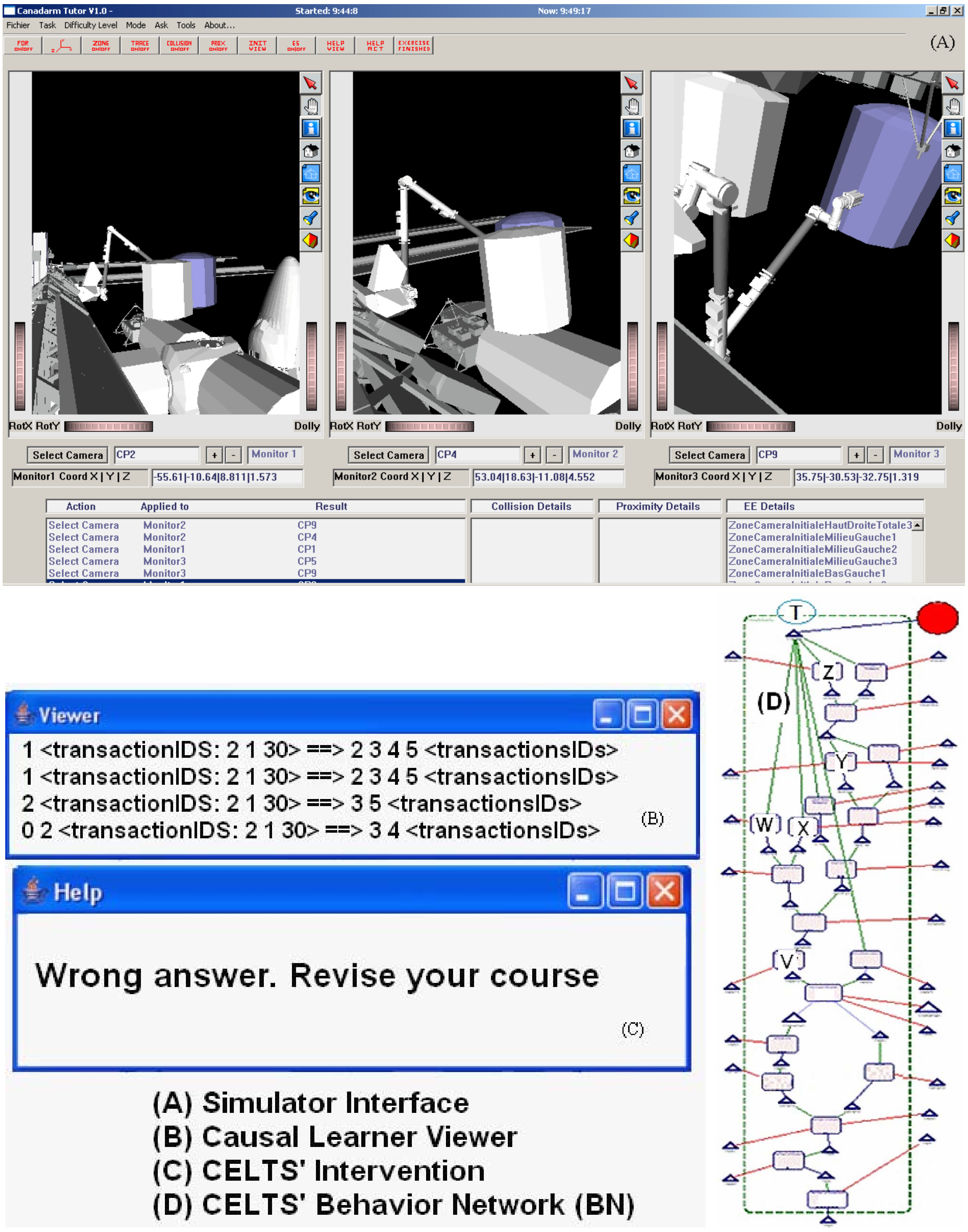


Figure 1. CELTS' architecture.

the failure of the action. These sequences of events are the interactions that took place between CELTS and users during Canadarm2 manipulation by users in the virtual world (**Figures 2(B)-(D)**). They are saved to different CELTS' Memories respecting the temporal ordering of the events that occurred between users and CELTS. The retrieved sequences of events contain the nodes (**Figures 2(B)-(D)**). Each node contains at least an event and an occurrence time (see CELTS' Episodic Learning [23] for more information). For instance, in **Figure 2(D)**, different interactions may occur between users and CELTS depending on whether the nodes' preconditions in the Behavior Network (BN) become true. Through this information, CELTS, using sequential pattern mining algorithms, extracts useful information. For example, if a user forgets to adjust the camera before he or she moves Canadarm2 and consequently chooses an incorrect joint, then the user will make collision risk. Such aforementioned information is gained through deductive reasoning in CELTS. Given such information, we were interested in finding the causes of the problem produced by the users in the virtual world. To do so, from all past events, the Causal Learning mechanism (CLM) constantly extracts association rules (e.g. $X \rightarrow Y$) between sets of events with their confidence and support[3] [24]. These rules indicate the groups of events that are frequently associated to other groups of events. From these association rules, CLM then eliminates the association rules that do not meet a minimum confidence and support according to the temporal ordering of events within a given time interval. This eliminates a large amount of non-causal rules from the retrieved sequences of events. After finding the candidate rule as the cause of the failure, CELTS' CLM re-executes it and waits for the user feed-

[3]Given a transaction database D, defined as a set of transactions $T = \{t_1, t_2, \cdots, t_n\}$ and a set of items $I = \{i_1, i_2, \cdots, i_n\}$, where $t_1, t_2, \cdots, t_n \subseteq I$, the support of an itemset $X \subseteq I$ for a database is denoted as sup(X) and is calculated as the number of transactions that contain X. The support of a rule $X \Rightarrow Y$ is defined as $\sup(X \cup Y)/|T|$. The confidence of a rule is defined as $\mathrm{conf}(X \Rightarrow Y) = \sup(X \cup Y)/\sup(X)$.



Select Camera CP2 Monitor 1 — Monitor1 Coord X | Y | Z: -55.61|-10.64|8.811|1.573

Select Camera CP4 Monitor 2 — Monitor2 Coord X | Y | Z: 53.04|18.63|-11.08|4.552

Select Camera CP9 Monitor 3 — Monitor3 Coord X | Y | Z: 35.75|-30.53|-32.75|1.319

Action	Applied to	Result
Select Camera	Monitor2	CP9
Select Camera	Monitor2	CP4
Select Camera	Monitor1	CP1
Select Camera	Monitor3	CP5
Select Camera	Monitor3	CP9

Collision Details

Proximity Details

EE Details:
ZoneCameraInitialeHautDroiteTotale3
ZoneCameraInitialeMilieuGauche1
ZoneCameraInitialeMilieuGauche2
ZoneCameraInitialeMilieuGauche3
ZoneCameraInitialeBasGauche1

Viewer

```
1 <transactionIDS: 2 1 30> ==> 2 3 4 5 <transactionsIDs>
1 <transactionIDS: 2 1 30> ==> 2 3 4 5 <transactionsIDs>
2 <transactionIDS: 2 1 30> ==> 3 5 <transactionsIDs>
0 2 <transactionIDS: 2 1 30> ==> 3 4 <transactionsIDs>
```

Help

Wrong answer. Revise your course

Figure 2. Virtual world simulator of Canadarm2 and CELTS' BN.

back. However, if after the execution of the candidate rule it turns out that the rule did not help the user to solve the problem, then CELTS' CLM writes in a failure in the WM. The failure leads CELTS' Causal Learning to examine other related nodes to the current failure with the highest support and confidence. Each time a new node is proposed by Causal Learning and executed by BN, an expectation node brings back to the consciousness mechanism the confirmation from users to make sure that the found rule was the cause of the failure. Finally, if a new cause is found, it will be integrated in CELTS' Causal Memory. In the end, if o solution can be found, the Causal Learning mechanism puts the following message in WM: "I have no solution for this problem."

This way of finding causal relations between different events is in accordance with what is suggested at the onset of this document such as statistical relations, temporal order, and prior knowledge.

After having proposed our causal model for CELTS, we now explain in detail the intervention of the causal process in CELTS' cognitive cycles. It is important to remember that two routes are possible during CELTS' cognitive cycle—a short route[4] (no causal learning occurs in this route) and a long route (various types of learning occur in this route such as episodic, causal and procedural). In both cases, the cycle begins with the perceptual mechanism. Hereafter, we briefly summarize each step in the cycle and in italics describe the influence of emotions (here called pseudo-amygdala[5] or EM and/or of CLM).

For a visual representation of the process, please refer to **Figure 1**.

Step 1: The first stage of the cognitive cycle is to perceive the environment, that is, to recognize and interpret the stimulus (see [1] for more information)[6].

EM: All incoming information is evaluated by the Emotional Mechanism when low-level features recognized by the perceptual mechanism are relayed to the emotional codelets, which in turn feed activation to emotional nodes in the Behavior Network (BN). Strong reactions from the "pseudo-amygdala" may cause an immediate reflex reaction in CELTS [23,25].

Step 2: The percept enters Working Memory (WM): The percept is brought into WM as a network of information codelets that covers the many aspects of the situation (see [1] for more information).

CLM: CLM also inspects and fetches WM relevant information. Relevant traces from different memories are automatically retrieved. These will be sequences of events in the form of a list relevant to the new information. The sequences include the current event, its relevant rules and the residual information from previous cognitive cycles in WM. The retrieved traces contain codelet links with other codelets. Each time new information codelets enter WM, the memory traces are updated depending on the new links created between these traces and the new information codelets. Once information is enriched, CLM sends it back to the WM.

Step 3: Memories are probed and other unconscious resources contribute: All these resources react to the last few consciousness broadcasts (internal processing may take more than one single cognitive cycle).

Step 4: Coalitions assemble: In the reasoning phase, coalitions of information are formed or enriched. Attention codelets join specific coalitions and help them compete with other coalitions toward entering "consciousness."

EM: Emotional codelets observe the WM's content, trying to detect and instill energy to codelets believed to require it and attach a corresponding emotional tag. As a result, emotions influence which information comes to consciousness and modulate what will be explicitly memorized.

Step 5: The selected coalition is broadcasted: The Attention mechanism spots the most energetic coalition in WM and submits it to the "access consciousness," which broadcasts it to the whole system. With this broadcast, any subsystem (appropriate module or team of codelets) that recognizes the information may react to it.

CLM: CLM starts by retrieving the past frequently reappearing information that best matches the current information resident in WM, ignoring their temporal part. This occurs by constantly extracting associated rules from the broadcasted information and the list of events previously consolidated. Then, CLM eliminates the rules that do not meet the temporal ordering of events.

Steps 6 and 7: Unconscious behavioral resources (action selection) are recruited. Among the modules that react to broadcasts is the Behavior Network (BN): BN plans actions and, by an emergent selection process, decides upon the most appropriate act to adopt. The selected Behavior then sends away the behavior codelets linked to it.

EM: When CELTS' BN starts a deliberation, for instance to build a plan, the plan is emotionally evaluated

[4]The short route is a percept-reaction direct process, which takes place when the information received by the perceptual mechanism is strongly evaluated by the pseudo-amygdala. The short route is described elsewhere see [3,4]. The long route is CELTS' full cognitive cycle.

[5]Let us note that in CELTS, a "pseudo-amygdala" is responsible for emotional reactions [2].

[6]The following steps, in bold characters, describe CELTS' full cognitive cycle. They are the same as in [2]. We restate them here; the difference is in what occurs during those steps from the causal learning perspective, which is in italics.

as it is built, the emotions playing a role in the selection of the steps. If the looping concerns the evaluation of a hypothesis, it gives it an emotional evaluation, perhaps from learned lessons from past experiences.

CLM: The extraction of the rules in step 5, may invoke a stream of behaviors related to the current event, with activation passing through the links between them (**Figure 2(D)**). *At this point CLM waits for CELTS' Behavior Network and CELTS' Episodic Learning Mechanism solution for the ongoing situation*) [23]. *Then, CLM puts its proposition as a solution in CELTS' WM, if the propositions from the decision making and the episodic learning mechanisms are not energetic enough to be chosen by CELTS' Attention Mechanism.*

Step 8: Action execution: Motor codelets stimulate the appropriate muscles or internal processes.

EM: Emotions influence the execution, for instance in the speed and the amplitude of the movements.

*CLM: The stream of behaviors activated in CELTS' BN (Step 7) may receive inhibitory energies, from CLM, for some of their particular behaviors. This means that, according to CELTS' experiences, CLM may use a shortcut (i.e. eliminate some intermediate nodes) between two nodes in behavior Network (BN) to achieve a goal (e.g. in **Figure (2D)** two points v and z). In some cases, again according to CELTS' experiences, CLM may prevent the execution of unnecessary behaviors in CELTS' BN during the execution of a stream of behaviors.*

4. The Causal Learning Process

The following subsections explain in detail the three phases of the Causal Learning mechanism as it is implemented in CELTS' architecture.

4.1. The Memory Consolidation Process

The Causal Memory consolidation process, which occurs in the step 2 of CELTS' cognitive cycle, takes place during each CELTS's cognitive cycle. CELTS' Causal Learning mechanism (CLM) extracts frequently occurring events from its past experiences, as they were recorded in its different memories [26]. In our context, CELTS extracts sequences of events during training sessions for Canadarm2 manipulation by astronauts inthe virtual world [27] (**Figure 2(A)**).To do so, a trace of what occurred in the system is recorded in CELTS' different memories during consciousness broadcasts [23]. For instance, each event $X = (t_i, A_i)$ in CELTS represents what happened during a cognitive cycle. The timestamp t_i of an event indicates the cognitive cycle number. The set of items A_i of an event contains an item that represents the coalition of information codelets (see Step 4 of CELTS' cognitive cycle) that were broadcasted during the cognitive cycle.

For example, one partial sequence recorded during our experimentations was (t = 1, c_2), (t = 2, c_4). This sequence shows that during cognitive cycle 1 the coalition c_2 (indicating that user forgot to adjust camera in the virtual world) was broadcasted, followed by the broadcast of c_4 (indicating that user brought about an imminent collision in the virtual world). If this subsequence appears several times during interactions of learners with CELTS, the following rule could be discovered: {Forget-camera adjustment (F), Bad joint (B) $\Rightarrow$ {Collision risk(C)}.

4.2. Learning by Extracting Rules from What is Broadcasted in CELTS

The second phase of Causal learning, which occurs in Step 5 of CELTS' cognitive cycle, deals with mining rules from the sequences of events recorded for all of CELTS' executions. To do so, the algorithm presented in takes as input the sequence database which contains sequences of coalitions that were broadcasted for each execution of CELTS, minsup, minconf and User Trace which are the traces of what occurs between the current user and CELTS. CELTS' uses the first three parameters to discover the set of all causal rules ($R_1, R_2, \cdots, R_n$) contained in the database (**Figure 3** Step 1). It then tries to inspect rules that match with the interactions between the current user and CELTS (User trace) in order to discover probable causes that could explain the user's behavior (Step 2). When CELTS does, one cause is returned.

The algorithm (**Figure 3**) performs as follows. 1) In STEP1, it saves in a sequence database the sequences of nodes (the coalitions) that are broadcasted by CELTS' Behavior Network (BN) during interactions with users to solve a problem. Then, in STEP2, the algorithm uses the Apriori algorithm [23] for mining association rules between nodes. This uncovers association rules of the form R_i: NODE$_i$ $\Rightarrow$ NODE$_f$, where NODE$_f$ and NODE$_i$ are potential causes and effects of the failure. The meaning of an association rule R_i is that if NODE$_f$ appears, we are likely to also find NODE$_i$ in the same sequence. But NODE$_i$ can appear before or after NODE$_f$. For this reason, the algorithm reads the original sequence database one more time to eliminate rules that do not respect the temporal ordering of the events.

To do this, we use two user-defined thresholds that a rule should meet in order to be kept. These thresholds are called minimal causal support and minimal causal confidence. Let s be the number of sequences in the sequence

```
Select Causal Nodes (LTMPatterns, MinSupp, MinConf, UserTrace)
STEP 1 "Discovering all causal rules":
        Apply the Apriori algorithm [34] to find all the
        association rules meeting a minimum support and a
        minimum confidence threshold.
    FOR each association rule found
        Calculate its causal support and causal confidence
        by looking at the sequence database again.
        Eliminate the rule if the causal support and causal
        confidence are lower than minimum thresholds for
        causal confidence and causal support.
        END FOR
STEP 2 "Searching for the most probable cause":
MaxCE = 0.0.
FOR each rule $R_i$ found in STEP1
IF R.LeftPart ⊆ UserTrace
            CE: = R.causalSupport × R.causalConfidence
            IF (CE > MaxCE)
                MaxCE: = CE
            Candidate Coalition.add: = R.Right
        END IF
        END IF
END FOR
RETURN CandidateCoalition, MaxCE
```

Figure 3. Causal Learning algorithm.

database. The causal support and confidence of a rule are defined respectively as *sup*(NODE$_i$ ⇒ NODE$_f$)/s and sup (NODE$_i$ ⇒ NODE$_f$)/sup(NODE$_i$), where sup(X ⇒ Y) denotes the number of sequences such that NODE$_i$ appears before (⇒) NODE$_f$ and sup(NODE$_i$) represents the number of sequences containing NODE$_i$ (see [28] for more details).

After eliminating the association rules that do not meet the minimum causal support and confidence thresholds, the set of rules that is kept is the set of all causal rules. A causal rule NODE$_i$ ⇒ NODE$_f$ is interpreted as: if NODE$_i$ occurs, then NODE$_f$ is likely to occur thereafter. In that case we will call NODE$_i$ the cause of the failure and NODE$_f$ the effect. Note that this definition can be extended for rules where the left part and the right part can contain more than one node. We here only provide examples with rules between two nodes. In Step 2 of the algorithm, CLM tries to select the more likely cause for a failure. To do so, this algorithm first sets a variable named MaxCE to zero. It then computes the causal estimation (CE) for each rule that matches with User Trace by multiplying its support and confidence.

Causal estimation (CE) of R_i
= (support of R_i × confidence of R_i)

This calculation is done for each rule to determine which node is the most likely to be the cause (the left part of the rule having the highest CE). The CE of a rule represents the causal estimation of a rule according to all the information that broadcasts in the system. For each such rule *r*, if the CE is higher than MaxCE, MaxCEis set to that new value and a variable candidate coalition is set to the right part of *r*. If the CE is lower than MaxCE, MaxCE remains intact. Finally, when the algorithm finishes iterating over the set of rules, the algorithm returns to CELTS' working memory the node (coalition) CandidateCoalitions contained in the right part of the rule which has the highest CE value.

Using this method for each node of the retrieved sequence, CELTS' CLM finds the most probable causes of the problem produced by the user while manipulating Canadarm2 in the virtual world. This node (coalition) will be broadcasted next by CELTS' consciousness mechanism to the user for further confirmation (see the next subsection for detail).

4.3. Construction of CELTS' Causal Memory

The creation of CELTS' Causal Memory (CM) occurs in Steps 7 and 8 of the cognitive cycle. The main elements of Causal Memory are the rules of the form X ⇒ Y. Like CELTS' Behavior Network (BN), the rules' left and right parts are nodes which are the coalitions broadcasted during CELTS' interactions with users. Each rule has a support and a confidence (used to calculate the CE, as described in the previous subsection).

Each new node (such as NODE$_p$) includes a context, an action, a result, and one or more causes. The context in this newly created node describes an ongoing event. The left part of the rule is filled by the node that caused the failure. The right part of the rule is considered as the effect. In what follows, we explain in detail how causal memory is formed. The algorithm is presented in **Figure 4**. It takes as parameters the sequence database, the

```
Causal Memory Constructor (LTM Patterns, MaxCE, $NODE_f$)
STEP 1 "Create a new rule in causal memory":
    Create a rule R such that R.RIGHT= $NODE_f$.
    FIND the node $NODE_P$ in LMPatterns that was executed prior to $NODE_f$ which has caused the error by looking in the BN.
    Attach an expectation codelet to $Node_P$. Then send $Node_P$ to WM.
    IF (Get User Confirmation For Cause Of Problem () = True)
        R.LEFT: = $NODE_P$
        $NODE_f$. Cause: = $NODE_P$. Action
STEP 2 "If not the cause, then search for the node with the next highest CE value"
    ELSE
        Search for the node $NODE_n$ in LM Patterns such that CE = MaxCE
        Attach an expectation codelet to $Node_n$.Then send $Node_n$ to WM.
        IF (Get User Confirmation For Cause Of Problem ( ) = True)
            R.LEFT: =$Node_n$
            $NODE_f$. Cause: = $Node_n$. Action
            save The Traces Of What Happened ()
        END IF
STEP 3 "If still no cause is found, then recursively examine other nodes":
        ELSE
            Initialize variables Node Temp: = null and MaxCE: = 0.
            FOR EACH node $NODE_k$ from $NODE_{n-1}$ to $NODE_1$ of LM patterns
                IF ($NODE_k$. CE > MaxCE and $NODE_k$. CE <$Node_n$. CE)
                    MaxCE: = $CE_n$. NodeTemp = $Node_K$.
                END IF
            END FOR
            Attach an expectation codelet to NodeTemp. Then send NodeTemp to WM
            IF (Get User Confirmation For Cause Of Problem () = True) and MaxCE > 0 )
                $NODE_f$. Cause: = NodeTemp. Action
                R.LEFT: = NodeTemp
            END IF
            ELSE
                Show message "I'm sorry, I cannot help you at this moment" to user.
END
```

Figure 4. Causal Memory Construction Algorithm.

maximum causal estimated node (MaxCE) calculated in the previous section, and the $NODE_f$ which is brought about by the user's error. Given the node $NODE_f$ that caused the error after execution by CELTS' BN, CLM creates (see **Figure 4.** Step 1) an empty rule (R) in CELTS' Casual Memory (CM) and copies the information in $NODE_f$ into the right part of the rule. During a user's manipulation of Canadarm2, CLM finds, from the current sequence of executed nodes, the node $NODE_p$ executed prior to $NODE_f$ which caused the user's error. It then attaches an expectation codelet to node $NODE_p$, puts it into the WM to be executed by BN and waits for the user's confirmation to find the cause of the problem. If the cause of the failure is $NODE_p$, CLM copies the action of node $NODE_p$ into the cause of the node $NODE_f$. CLM then copies the information of $NODE_p$ into the left part of the created rule R in CELTS' Causal Memory and makes a direct link between $NODE_f$ and $NODE_p$.

If, however, it turns out that the node $NODE_p$ in the previous step is not the cause of the error, then CLM (**Figure 4.** Step 2) searches for the node $NODE_n$ with the next highest CE value (MaxCE, explained in the previous subsection). It then, attaches an expectation codelet to it, puts it into the WM to be executed by BN and waits for the user confirmation. If the cause of the error is $NODE_n$, CLM copies its action to the $NODE_f$'s cause and all information into the left part of the created rule R in CELTS' CM. Finally to save the traces of what was done to find the cause, 1) CLM creates a sequence of empty nodes similar to what is retrieved as the sequences of executed nodes from CELTS' different memories, 2) assigns $NODE_n$ to its first node and $NODE_f$ to the last node and 3) copies to the sequence created in CM all intermediate nodes between $NODE_n$ and $NODE_f$, and then creates links between them. The nodes $NODE_n$ and $NODE_f$ in this sequence are tagged as the cause and effect of the problem that caused the error.

However, if, in the execution of the node $NODE_n$ in the previous step, the resulting information brought back by the expectation codelet to WM does not meet the expected results, CLM then (**Figure 4**. Step 3) repeatedly searches for the node of the sequence from $NODE_{n-1}$ to $NODE_1$ with the highest CE value but less than the $NODE_n$'s CE value and pursues the same previous processes as explained in steps one and two to find the cause of the error. This process will continue for the remaining nodes retrieved from CELTS' LTM if each attempt fails. If CELTS cannot find any cause, the message "I cannot find the cause of the problem" is shown.

4.4. Using Mined Patterns to Improve CELTS' Behavior

The third part of CELTS' Causal Learning occurs in Step

7 and Step 8 of CELTS' cognitive cycle. It consists of improving CELTS' behavior by making it reuse found rules to predict why users are making mistakes, determine how to best help them, and in some specific cases, to reconstruct the Causal Memory (CM). Finding causes will directly influence the actions that will be taken by CELTS' Behavior Network (BN). CELTS' behavior will improve due to the fact that the more it interacts with users and they confirm the correctness of the found causes for their mistakes or not, the more the estimated CE values for the nodes in the rules get reinforced or weakened. After some interactions between CELTS and the users, CLM may find for instance a chain of interrelated nodes. For instance, in **Figure 2(D)**, the node *V* is usually in relation with node *Y* and node *Y* is in relation with node *Z*, according to user confirmations and the minimum support and confidence defined by the domain expert. For instance, CLM learned after several interactions with users that 60 % of the time "user chose the wrong joints → user makes Canadarm2 pass too close to the ISS". This means that after a while CELTS' CLM is capable of jumping from a start point in the BN to a goal and eliminates unnecessary nodes between them.

However, jumping from one point to a goal point in the BN is not always a good decision as CELTS is a tutor and some intermediate nodes are very important hints to users. To solve this problem, in the first step, we tagged the important nodes in the BN as not to be eliminated. Thus, some experiments go from one point to the other (for instance in **Figure 2**. Dnodes V → Z), CELTS' CLM makes an obligatory passage through intermediate nodes such as node *Y* and eliminates only unnecessary nodes between them. In the second step, to automatically eliminate unnecessary nodes that have not been pre-tagged by a human expert, we used the aforementioned algorithms (previous subsection **Figure 4.** Step 2 and Step 3) for finding causes when the users make an error while interacting with CELTS. This means that to achieve a goal from a start point in the BN, according to CELTS' experiences with users, CLM must decide to preserve important nodes and only eliminate those that are unnecessary in the BN (e.g. **Figure 2(D)** two points *v* and *z*).

Finally, it is worth noting that CELTS' BN is an acyclic graph. The Causal Markov Assumption (CMA) postulates that for any variable *X*, *X* is conditionally independent of all other variables in an acyclic causal graph (except for its own direct and indirect effects) based on its own direct causes. Accordingly, the refined BN produced by CLM could be considered a primitive proposition for the construction of a causal Bayesian network.

For instance, like the cars' side and front mirrors example given above, after several interactions with users rules are extracted by the algorithms: 1) Forgetting camera adjustment (F) → Choosing Bad joint (B) → collision risk (C); 2) Choosing bad joint (B) ← Forgetting camera adjustment (F) → collision risk (C). If we assume that CMA holds, both structures in our example entail exactly the same conditional and unconditional inde- pendent relationships: In both, F, B and C are dependent and F and C are independent conditional on B [18]. Moreover, an important difference between the BN and Bayesian Networks is that no node in the BN possesses a table of conditional probabilities like nodes of Bayesian networks do. Instead, the information about probabilities is stored in each causal rule as the causal confidence and causal support values which can be interpreted as an estimate of the probability P(Y|X) [29-31].

5. Testing Causal Learning in the New CELTS

To validate CELTS' Causal Learning mechanism (CLM), we integrated it into Canadarm2 simulator, our simulator designed to train astronauts to manipulate Canadarm2 (**Figure 2(A)**). Users were invited to perform arm manipulations using the simulator. In these experiments, users had to move Canadarm2 from one configuration to another in the virtual world while avoiding collisions between Canadarm2 and the space station. This is a complex task, as Canadarm2 has seven joints and the user must 1) choose the best three cameras (from a set of about twelve cameras on the space station) for viewing the environment (since no camera offers a global view of the environment), 2) not move Canadarm2 too close to the ISS, 3) choose the right joint for the arm movements, and 4) adjust parameters of cameras properly. These experiments sought to validate CELTS' ability to find the causes of mistakes made by users. During these experiments, we observed that CELTS was able to find the causes and propose appropriate hints to help users. Some experiments are described next.

Users' Learning Situations

A user learns by practicing Canadarm2 manipulations while receiving hints created initially by an expert and given to the user by CELTS. We performed more than 300 CELTS executions of Canadarm2 in the virtual world including good moves and dangerous moves, such as collisions. During each execution, CELTS chose a tutoring scenario depending on the situation. Our experiments showed that CELTS is often capable of finding the right causes of problems created by users in different situations. In what follows, two different experiments are detailed.

Experiment 1: Approximate Problem

When manipulating Canadarm2, it is important for the users to know the exact distance between Canadarm2 and ISS at all times. This prevents future collisions or collision risks on the ISS. **Figure 2(D)** shows the scenario created by an expert in the CELTS' Behavior Network (BN). This scenario is an intervention by CELTS to help the user while manipulating Canadaram2 to avoid collisions between Canadaram2 and ISS. The user weakly estimated the distance between Canadaram2and ISS because 1) the user chose to move the wrong joint; 2) the user was tired; 3) the user did not remember his course; 4) the user has never passed through this zone.

As we can see in the **Figure 2(D)**, this is a long scenario and each time, to find the cause of mistakes made by the user, CELTS may be required to interact for a long period of time (e.g. asking questions, giving hints, and demonstrating some examples) to find the causes and provide appropriate feedback. The scenario starts when CELTS detects that a user has chosen the wrong joint and is moving Canadarm2 too close to the ISS. CELTS first prompts the following message: Have you ever passed through this zone? 1) If the answer given by the user is yes, CELTS asks the user to verify the name of the joint that she has selected. If the user fails to answer correctly, CELTS proposes a hint in the form of a demonstration or it stops Canaradm2 manipulation. In this case, the user needs to revise the course before starting Canaradm2 manipulation again; 2) if the user's answer is no, CELTS asks her to estimate the distance between Canadarm2 and ISS. If the user fails to answer correctly then the next hint from CELTS asks the user if she is tired or forgot the course about this zone or if he or she needs some help; if the user answers correctly, it means that the user is an expert user and that the situation is not dangerous.

Interacting with various users and according to the users' answers, CELTS found the following rule 1) 60 % of the time "user chose the wrong joints → user makes Canaradm2 pass too close to the ISS"; 2) in 35% of the time "user has never passed through this zone → user manipulates near to the ISS", **Figure 2(D)**; 3) in 5% of the time "user is an expert → user makes Canaradm2 pass too close to the ISS".

It must be noted that the percentage value attributed to the extracted rules varies depending on the users' answers to CELTS' questions.

Experiment 2: Camera Adjustment Problem

As explained above, forgetting to adjust the camera prior to moving Canadarm2 increases collision risk (as depicted in **Figure 2(A)**). During our experiments, we noted that users frequently forgot this step, and moreover, users frequently did not realize that they had neglected this step. This increases the risk of collisions (as depicted in **Figure 2(A)**) in the virtual world. We thus decided to implement this situation as a medium-threat situation in CELTS' BN (e.g. **Figure 2(D)**).

When a user forgot to perform camera adjustments, CELTS had to make a decision; it could either 1) give a direct solution such as "You must stop the arm immediately" or 2) give a brief hint such as "I think this movement may cause some problems. Am I wrong or right?" or 3) give a proposition such as "Stop moving Canadarm2 and revise your lessons". Through interactions with different users, CELTS recorded sequences of events, each of them carrying emotional valences (see [2] for more details).

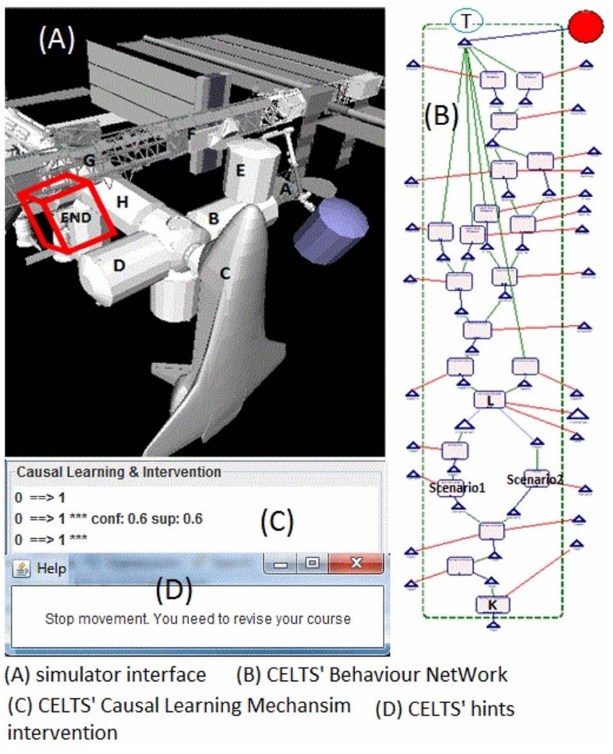
From the interactions that occurred between CELTS and users to solve camera adjustment, CLM drew the following conclusions: 1) 60% of the time, "the user is tired → the user performs a camera adjustment error"; 2) 30% of the time, "the user has forgotten this lesson → the user performs a camera adjustment error" and 3) 10% of the time, "the user lacks motivation → the user is inactive".

After some trials, CELTS' CLM is capable of inducing (by jumping from one point to another point in the BN, **Figure 2(D)**) the source of the users' mistakes and proposing a solution for them in the virtual world. However, given that CELTS is a tutor and must interact with the user, jumping from the start point to the end of the scenario (**Figure 2(D)**, $V \rightarrow Z$) causes the elimination of some important steps in the BN. To prevent this, as mentioned before, we tagged the important nodes in the BN as not to be eliminated. Thus, after some experiments, to go from $V \rightarrow Z$, CLM obligatorily passed through intermediate nodes such as node Y in **Figure 2(D)** We call this process as CELTS' partial Procedural Learning (Step 8 of CELTS' cognitive cycle).

Experiment 3: Complex Situation

To evaluate the extent of CELTS' capabilities when equipped with CLM, we decided to examine a very complex path in the virtual world. We considered an exercise between two ISS modules, JEMEF01 (labeled and is referred as A) and MPLM02 (labeled by red cube and is referred to as END) in the virtual world (as shown in **Figure 5(A)**) in which users' mistakes while moving Canadarm2 from configuration A to END are very likely.

As shown in **Figure 5(A)**, Canadarm2 is very close to configuration A. Thus, the exercise starts near to the module A and finishes at module END. In the first step of this experiment, the user has handled the collision risk problem with the configuration A. In the second step, the user faces at least four paths, from configuration A to END (**Figure 5(A)**). Importantly, the expert system has


(A)
END
H
B
D
C
E
G
F
A
(B)
T
L
Scenario1
Scenario2
K
(C)
(D)
(A) simulator interface
(B) CELTS' Behaviour NetWork
(C) CELTS' Causal Learning Mechansim
(D) CELTS' hints intervention

```
Causal Learning & Intervention
0 ==> 1
0 ==> 1 *** conf: 0.6 sup: 0.6
0 ==> 1 ***
```

Help

Stop movement. You need to revise your course

Figure 5. Simulation of the International Space Station (ISS).

conceived only three scenarios in the BN regarding only three paths with their corresponding obstacles to be avoided: P_1 (AECDH), P_2 (AEBCDH) and P_3 (AEFGHD) (**Figure 5(A)**).

Whichever paths are chosen by the users, obstacles *A*, *E*, *B*, *C*, *D*, *H*, *G*, and *F* have to be avoided in the virtual world to prevent any collision. Therefore, the nodes in the BN corresponding to those obstacles in the virtual world are marked as "Not to be eliminated", by the domain expert.

The domain expert marks Configurations *A*, *C* and *D* as very important for the paths P_1 and P_2. Thus, in configuration *C*, in order to go through them without causing any collision risk, the user must first rotate camera8 60 degrees horizontally (**Figure 5**) and then choose the specific joint EP and then joint SP (**Figure 5**). In configuretion *D*, the user must first adjust camera6 in order to have a good view of obstacles *G* and *H* before performing any movements. In path P_3, the user must respect the following steps to prevent collision risk while manipulating Canarm2 from the configuration *A* to the END. First, in configuration *E*, camera2 must be turned 30 degrees, and the manipulation must then be continued using joint *SR*. Then, in configuration *F*, the joint *SY* must be selected and rotated 90 degrees to prevent any collision with ISS. In configuration *G*, the obstacle *H* must be avoided by rotating Canadarm2 60 degrees.

It must be noted that in "Part One" of this experiment, some specific nodes in CELTS' BN (**Figure 5**) are marked as "not to be eliminated," since our purpose here is to examine CELTS' capacity to find the best scenario among different solutions given by the expert. And in "Part two" of this experiment, nodes in CELTS' BN can be eliminated. Thus, CELTS must find: 1) the best Scenario; 2) the cause of users' mistakes and eliminate unnecessary nodes between points L and T in the BN (**Figure 5**).

Thus, after a number of interactions with different users, we expect CELTS to propose the most self-satisfying paths from configuration *K* to *L* and eliminate unnecessary nodes between points *L* to *T*. The experiment is divided into two parts:

Part One

When the user (**Figure 5(A)**) begins a manipulation and makes a mistake, the precondition of BN nodes activates and waits for the relevant information to fire corresponding nodes and demonstrate a message to the user. For instance, the BN node *K* activates when Canadarm2 approaches configuration A in the virtual world. To help users handle the collision risk problem with configuration *A*, the domain expert conceived two paths in CELTS' BN (from points *K* to *L* in **Figure 5(B)**) that correspond to this situation in the virtual world. After interacting with users at point L, at the end of scenario1 and scenario2, CELTS asks an evaluation question to be sure that the hints or questions given to the users were useful and that users are aware of the collision risk in the virtual world.

It must be noted that due to the imminent collision risk, users' incorrect answers to CELTS' inquiries will activate the short route and trigger direct emotional interventions as explained in CELTS' cognitive cycles (see [2] for more details).

As explained above, during the collision risk, CELTS has here two choices to help users handle the situation. It can give a direct solution to the users (scenario2, **Figure 5(B)**) or start by providing hints to help them handle the situation by themselves (scenario1, **Figure 5(B)**).

After many executions, CELTS extracted corresponding frequent event sequences for the first part of this experiment (**Figure 5(B)** from node *K* to *L* in the BN), with a minimum support (minsup) higher than 0.45. Using the information extracted from this experiment, CELTS proposed scenario1 to help users prevent collision risk in the virtual world (**Figure 5(B)**), because it contains a positive emotional valence as opposed to scenario2.

Part Two

In the second part of the experiment, after CELTS learned to choose the best scenario to help users prevent collision risk with configuration *A* (**Figure 5(A)**), users were asked to continue their manipulation and move Canadarm2 to the configuration END. CLM learned how to help users when they choose paths P_1, P_2, P_3 to move Canadarm2 from configuration *A* to END based on the domain experts' hints and questions in the BN during the 300 random executions mentioned at the onset of this section.

Here are the details. The extracted information from the second part of our experiment is 1) 50% of the time, "the user is tired → the user forgot to adjust camera8"; 2) 40% of the time, "the user is tired → the user performs a bad manipulation of Canadarm2"; 3) the remaineder10 % rules found by CLM are that "the user is tired → user must revise the course", "the user is tired → user did not make a collision risk", and "the user is tired → user wants to continue Canadarm2 manipulation". Note, however, that the third rule found by CLM is not always true.

Other extracted information demonstrated that 1) 50% of the time, "the user forgot to adjust cameras → the user had a bad view of ISS' configurations and Canadarm2; 2) 20% of the time, the "the user had a bad view of ISS" configurations and Canadarm2 → the user caused a collision risk near obstacles *C*, *D*, *E*, and *F*; 3) 20% of the time "the user forgot to adjust the cameras → the user manipulates very near to obstacles *G* and *H*"; 4) the remainder 10% rules found by CLM are that "the user forgot to adjust cameras → the user must review the lesson", "the user forgot to adjust cameras → the user adjusted camera8", and "the user was not tired → the user forgot to answer questions".

The extracted rules in this experiment demonstrate that if a user forgets to adjust the cameras in the virtual world, he or she will have a bad view of the virtual world and this will increase collision risk.

The extracted rules could be interpreted such that the probability of the user forgetting to adjust the cameras is independent of the probability of a collision with ISS' configurations, provided that the user has poor visibility in the virtual world. The extracted rules could also be interpreted such that the probability of having a poor view of ISS' configurations is independent of the probability of causing collisions in the virtual world provided that the user has forgotten to adjust the camera.

The percentage values CELTS attributed to the various possible causes are true most of the time, although they must be verified by a domain expert before use. These experiments demonstrated that CELTS is capable of choosing the best scenario for a given situation, selecting that which has received the highest positive emotional valence during its interactions with the users. It is furthermore capable of eliminating unnecessary nodes in the BN.

The text has referred up to now to paths 1 through 3 in the explanation of how to go from configuration *A* to END procedures. However, there exists a path P4 which could be considered as a shortcut.

The relevant obstacles to be avoided for this path are: *A*, *E*, and *D*. Ideally, CELTS would eventually ask users if they have some information about the obstacles they will encounter. However, CELTS cannot ask these questions when users choose path P_4 prior to starting Canadarm2 manipulation, since the domain expert has not conceived relevant scenarios for this path (P_4) in CELTS' BN. In this case, CELTS' CLM automatically connects to the CanadarmTutor database [27]. The database contains different paths that users such as experts and novices have previously performed to move Canadarm2 on the ISS. Searching all the information about paths, CELTS' CLM, has the capacity of giving primitive hints to users when they encounter obstacles *E*, *D* and *H* in path P_4.

One of our future goals would be to equip CELTS with the capacity of asking users about obstacles they might encounter in this path, before the manipulation starts.

6. CELTS' Performance after the Implementation of Causal Learning

The rule mining algorithm explained in this paper was tested in different projects. We first explain how the proposed algorithm improved CELTS performance. Second we briefly explain the use of the proposed algorithm in another e-learning project. Third, we discuss the performance of the proposed algorithm with real data such as biological database.

1) We performed a second experiment with CELTS' causal learning mechanism, but this time to observe how our rule algorithm behaves when the number of recorded sequences increases. The experiment was done on a 3.6 GHz Pentium 4 computer running Windows XP, and consisted of performing more than 250 CELTS executions for various situations (e.g., scenario 1 and scenario 2 in **Figure 5(B)**). In this situation, CELTS conducts a dialogue with the user that includes from four to 20 messages or questions depending on what the user answers and the choices CELTS makes. During each trial, we randomly answered the questions asked by CELTS, and took various measures during CELTS' learning phase. Each recorded sequence contained approximately 30 broadcasts. **Figure 6** presents the results of the experiment. For all graphs, the X axis represents the executions from 1 to 250. The *Y* axis denotes execution times in graph A, and rule counts in graph B-D. The first graph

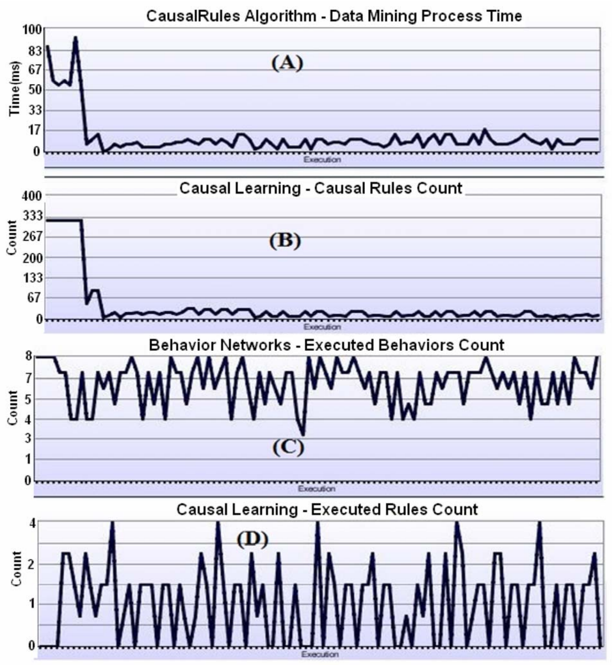


Figure 6. Data mining algorithms' performance.

(*A*) shows the time for mining rules which was generally short (less than 10 s) and after some executions remained low and stabilized at around 4 rules during the last executions. In our context, this performance was very satisfying. However, the performance of the rule mining algorithm could still be improved as we have not yet fully optimized all of its processes and data structures. In particular, in future works we will consider modifying the algorithm to perform incremental mining of rules. The second graph (*B*) shows the number of causal rules found after each CELTS execution. This would improve performance, as it would not be necessary to recalculate from scratch the set of patterns for each new added sequence. The third graph (*C*) shows the average number of behaviors executed (nodes in the BN) for each CELTS execution without causal learning. It ranges from 4 to 8 behavior broadcasts. The fourth graph (*D*) depicts, after the implementation of causal learning, the number of rules used by CELTS at each execution. Each executed rule means that CELTS skip some unnecessary intermediate steps in the BN. The average number of executed rules for each interaction ranged from 0 to 4 rules. This means that CELTS generally used fewer nodes to perform the same task after the implementation of causal learning.

2) A second use of the causal rule mining algorithm explained in this paper is in an intelligent agent [32]. The project is aimed at discovering patterns while observing humans performing specific procedural tasks (see [32] for more details). Once, the agent learned to perform the task it can reuse rules and other kinds of patterns found to perform the task by itself or teach the task. The mechanism implemented in this agent is different from the causal learning mechanism in CELTS, in that it is designed to learn a task instead of finding causes, and it is not based on cognitive theories.

3) At this point we discuss very briefly the contributions of our algorithm to the field of data mining. The first issue is time complexity. Without going into technical details, the most costly part of the rule mining algorithm used in this paper is the the Apriori algorithm. The Apriori algorithm time complexity was shown by Hegland [33] to be O(d_2,*n*) where *d* is the number of different items and n is the number of transactions in the database. The elimination of association rules that do not respect the temporal ordering to obtain causal rules is performed in linear time with respects to the number of sequences that contain the antecedent of each rule, the size of each sequence, and the total number of rules [28]. However, it must be noted that, regarding performance, we have also recently designed an alternative algorithm that is faster for discovering causal rules and is not based on Apriori [32]. Integrating it into CELTS would further improve its performance.

To test the performance of our algorithm for discovering causal rules from the data mining point of view, we have also performed several performance studies that have been published in a conference paper on data mining [28]. These studies were carried out on several large real-life datasets such as click-streams data from websites and biological sequences of proteins. In these studies, the data mining procedure has shown good performance for datasets having up to 70,000 sequences even under very low confidence and support thresholds (for example, the algorithm terminated in less than 250 seconds for minSup = 0.05 and minSeqConf = 0.3 with 70,000 sequences on one dataset), which demonstrates its efficiency for much larger amounts of data than what is recorded in CELTS.

7. Comparison between Different Architectures' Learning Capabilities

Now we compare CELTS' learning capabilities with three popular architectures: LIDA [34], ACT-R [35] and CLARION [16] (**Table 1**).

While LIDA is not equipped with Causal Learning, CLARION is equipped with supervised Causal Learning. However, at this point, there is no computational model for causal learning proposed in CLARION. CELTS' Causal Learning Mechanism occurs in an unsupervised fashion and through a type of reinforcement learning, for it partially depends on the temporal occurrence of the

Table 1. Comparison between LIDA, ACT-R, CLARION and CELTS (— = the architecture is not equipped with this specific learning; *X* = the learning mechanism is implemented).

	LIDA (Franklin, 2006)	ACT-R (Anderson, 2004)	CLARION (Sun, 2006)	CELTS (2010)
Explicit Perceptual Learning	*X*	—	*X*	—
Episodic Learning	*X*	*X*	—	*X*
Explicit Procedural Learning	*X*	*X*	*X*	*X*
Implicit Procedural Learning	—	*X*	*X*	*X*
Emotional Learning help other types of learning	—	—	—	*X*
Bottom-up Supervised Learning	*X*	—	*X*	*X*
Supervised Causal Learning	—	—	*X*	*X*
Unsupervised Causal Learning	—	*X*	—	*X*

events and the users' confirmation. As is the case with LIDA's architecture, CELTS' bottom-up learning is implemented for all types of learning such as learning of Emotional, Episodic, Procedural and Causal learning. CELTS is not equipped with Attention Learning. One of our interests is to find a way to integrate it into the architecture.

8. Conclusions

In this paper, using a combination of sequential pattern mining and association rule algorithms, we explained how to integrate causal learning in an Emotional Learning Tutoring System (CELTS). In CELTS, procedural learning is dependent on the causal learning which is further dependent upon the episodic learning. All learning is influenced by emotions.

As in the case of humans, the episodic and causal memories in CELTS mutually influence each other during interactions with learners. For instance, if the causes found by CELTS turn out to be false, it influences the support of the causal rules which in turn influences episodic memory—the increase or decrease of the events supports.

To our knowledge, researchers in artificial intelligence have, up to now, used Bayesian methods to study causal reasoning and causal learning models for cognitive agents. However, the Bayesian approach is not applicable when the agent faces large amounts of data. Another important issue with Bayesian Networks is that they generally require domain experts to specify conditional probabilities by hand, which is often a difficult and time-consuming task. For CELTS, we chose to use data mining algorithms instead of Bayesian Networks because we wanted to create a completely automatic approach that could learn causal knowledge incrementally. The combination of techniques used (sequential pattern mining and association rule mining) is original for proposing a causal learning model for cognitive agents.

However, the causal learning algorithms used in this study are not incremental. Therefore, for each CELTS execution, the algorithms must read the whole database. Another limitation in our work is that given the observed data and the confidence and support calculated by CELTS' CLM, the question remains as to how one could produce the probability distribution as it exists in Bayesian Networks.

9. Acknowledgements

The authors thank the Fonds Québécois de la Recherchesur la Nature et les Technologies for funding this research.

10. References

[1] D. Dubois, P. Poirier and R. Nkambou, "What Does Consciousness Bring to CTS?" Springer, Berlin, 2007, pp. 803-806.

[2] U. Faghihi, P. Poirier, P. Fournier-Viger and R. Nkambou, "Human-Like Learning in a Conscious Agent," *Journal of Experimental & Theoretical Artificial Intelligence*, in Press.

[3] U. Faghihi, P. Poirier, D. Dubois and M. Gaha, "A New Emotional Architecture for Cognitive Tutoring Agents," *Proceedings FLAIRS Conference*, Coconut Grove, 2008, pp. 445-446.

[4] U. Faghihi, P. Poirier, D. Dubois, M. Gaha and R. Nkambou, "How Emotional Mechanism Learn and Helps Other Types of Learning in a Cognitive Agent," *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology* (*WI-IAT* 2008), IEEE Computer Society Press, Washington, 2008.

[5] D. Purves, E. Brannon, R. Cabeza, S. A. Huettel, K. LaBar, M. Platt and M. Woldorff, "Principles of Cognitive Neuroscience," Sinauer Associates, Sunderland, 2008.

[6] C. B. Martin and M. Deutscher, "Remembering," *Philosophical Review*, Vol. 75, No. 2, 1966, pp. 161-196. doi:10.2307/2183082

[7] S. Shoemaker, "Persons and Their Pasts," *American Philosophical Quarterly*, Vol. 7, No. 4, 1970, pp. 269-285.

[8] J. Perner, "Memory and Theory of Mind," In: E. Tulving

and F. I. M. Craik, Eds., *The Oxford Handbook of Memory*, Oxford University Press, Oxford, 2000, pp. 297-312.

[9] S. Bernecker, "The Metaphysics of Memory," Springer, Berlin, 2008. doi:10.1007/978-1-4020-8220-7

[10] A. Maldonado, A. Catena, J. C. Perales and A. Cándido, "Cognitive Biases in Human Causal Learning," 2007.

[11] P. Maes, "How to Do the Right Thing," *Connection Science*, Vol. 1, No. 3, 1989, pp. 291-323. doi:10.1080/09540098908915643

[12] D. A. Lagnado, M. R. Waldmann, Y. Hagmayer and S. A. Sloman, "Beyond Covariation: Cues to Causal Structure," In: A. S. Gopnik and L. Schultz, Eds., *Causal Learning*: *Psychology*, *Philosophy*, *and Computation*, Oxford University Press, Oxford, 2007, pp. 154-172.

[13] A. S. Gopnik and L. Schulz, "Causal Learning: Psychology, Philosophy and Computation," Oxford University Press, Oxford, 2007.

[14] J. R. Anderson, "Rules of the Mind: Mahwah," Lawrence Erlbaum Associates, New York, 1993.

[15] W. Schoppek, "Stochastic Independence between Recognition and Completion of Spatial Patterns as a Function of Causal Interpretation," *Proceedings of the* 24*th Annual Conference of the Cognitive Science Society*, Lawrence Earlbaum Associates, Mahwah, 2002, pp. 804-809.

[16] R. Sun, "The CLARION Cognitive Architecture: Extending Cognitive Modeling to Social Simulation Cognition and Multi-Agent Interaction," Cambridge University Press, New York, 2006.

[17] S. Hélie, "Modélisation de L'apprentissage Ascendant des Connaissances Explicites dans une Architecture Cognitive Hybride," PHD, DIC, UQAM, Montréal, 2007.

[18] A. Gopnik, C. Glymour, D. M. Sobel, L. E. Schulz, T. Kushnir and D. Danks, "A Theory of Causal Learning in Children: Causal Maps and Bayes Nets," *Psychological Review*, Vol. 111, No. 1, 2004, pp. 3-32. doi:10.1037/0033-295X.111.1.3

[19] M. Braun, W. Rosenstiel and K.-D. Schubert, "Comparison of Bayesian Networks and Data Mining for Coverage Directed Verification Category Simulation-Based Verification," High-Level Design Validation and Test Workshop, *Eighth IEEE International*, 2003, pp. 91-95.

[20] S. Franklin and F. G. Patterson Jr, "The LIDA Architecture: Adding New Modes of Learning to an Intelligent, Autonomous, Software Agent," *Integrated Design and Process Technology*, 2006.

[21] D. Hofstadter, R and M. Mitchell, "The Copycat Project: A Model of Mental Fluidity and Analogy-Making," In: K. J. Holyoak and J. A. Barnden, Eds., *Advances in Connectionist and Neural Computation Theory*, Ablex, Norwood, 1994.

[22] D. Dubois, "Réalisation d'un Agent Doté D'une Conscience Artificielle: Application à un Système Tutorel Intelligent," Université du Québec à Montréal, Montréal, 2007.

[23] U. Faghihi, P. Fournier-Viger, R. Nkambou, P. Poirier and A. Mayers, "How Emotional Mechanism Helps Episodic Learning in a Cognitive Agent," *Proceedings of the* 2009 *IEEE Symposium on Intelligent Agents*, Nashville, 2009, pp. 23-30. doi:10.1109/IA.2009.4927496

[24] R. Agrawal, T. Imielminski and A. Swami, "Mining Association Rules between Sets of Items in Large Databases," *SIGMOD Conference*, 1993, pp. 207-216.

[25] L. R. Squire and E. R. Kandel, "Memory: From Mind to Molecules," W. H. Freeman, San Francisco, 2000.

[26] V. Goel and R. J. Dolan, "Differential Involvement of Left Prefrontal Cortex in Inductive and Deductive Reasoning," *Cognition*, Vol. 93, No. 3, 2004, pp. 109-121. doi:10.1016/j.cognition.2004.03.001

[27] R. Nkambou, K. Belghith and F. Kabanza, "An Approach to Intelligent Training on a Robotic Simulator Using an Innovative Path-Planner," *Proceeding of the 8th International Conference on Intelligent Tutoring Systems* (*ITS*), LNCS, 2006, pp. 645-654.

[28] P. Fournier-Viger, U. Faghihi, R. Nkambou and E. M. Nguifo, "CMRULES: An Efficient Algorithm for Mining Sequential Rules Common to Several Sequences," FLAIRS Conference, 2010.

[29] J. Hipp, U. Güntzer and G. Nakhaeizadeh, "Data Mining of Association Rules and the Process of Knowledge Discovery in Databases," *Industrial Conference on Data Mining*, 2002, pp. 15-36.

[30] J. Deogun and L. Jiang, "Prediction Mining—An Approach to Mining Association Rules for Prediction," *Proceedings of RSFDGRC* 2005 *Conference, Springer-Verlag*, Berlin, 2005, pp. 98-108.

[31] L. Li and J. S. Deogun, "Discovering Partial Periodic Sequential Association Rules with Time Lag in Multiple Sequences for Prediction," *Proceedings of ISMIS* 2005, Springer, Berlin, LNCS 3488, 2005, pp. 332-341.

[32] P. Fournier-Viger, "Knowledge Discovery in Problem-Solving Learning Activities," Ph.D. Thesis, University of Quebec in Montreal, 2010.

[33] M. Hegland, "The Apriori Algorithm—A Tutorial. Mathematics and Computation," *Imaging Science and Information Processing*, Vol. 11, 2007, pp. 209-262. doi:10.1142/9789812709066_0006

[34] S. Franklin, "A Cognitive Theory of Everything: The LIDA Technology as an Artificial General Intelligence," Artificial General Intelligence Research Institute (AGIRI) 2006.

[35] J. R. Anderson, D. Bothell, M. D. Byrne, S. Douglass, C. lebiere and Y. Qin, "An Integrated Theory of the Mind," *Psychological Review*, Vol. 111, No. 4, 2004, pp. 1036-1060. doi:10.1037/0033-295X.111.4.1036