Algorithmic Knowledge Profiles for Introspective Monitoring in Artificial Cognitive Agents

Manuel F. Caro, Adán A. Gómez and Juan C. Giraldo

Departamento de Informática Educativa
Universidad de Córdoba – Montería, COL
{manuelcaro; aagomez; jgiraldo} @correo.unicordoba.edu.co

***Abstract*—This paper describes a new approach for the representation of profiles of cognitive functions. The profiles are used in introspective monitoring to keep updated the meta-level about the most relevant attributes of each cognitive function that is executed in the object-level. The profiles are called *algorithmic knowledge profiles*. Declarative meta-knowledge, procedural meta-knowledge and strategic meta-knowledge are represented using the *algorithmic knowledge profile*. The profiles are formally defined and the computational implementation is based on extended tabular trees. The approach is illustrated using an instructional planner agent as an example.**

Keywords—introspective monitoring; cognitive computing; metacognition in computation; algorithmic knowledge profile; extended tabular trees.

I. Introduction

Cognitive Computing is an emerging paradigm of Artificial Intelligence (AI) based on Cognitive Informatics, that implements computational intelligence by autonomous inferences and perceptions mimicking the mechanisms of the brain and natural intelligence [1].

Computational metacognition in Cognitive Computing refers to the capability of Intelligent Systems (IS) to monitor and control their own learning and reasoning processes [2]. Reasoning and learning processes are two higher level cognitive functions of natural intelligence. Metacognition allows IS to gain awareness of knowledge about cognition and control of cognition. According to [3], [4] an IS with metacognitive abilities is composed at least by two cognitive levels named object-level and meta-level. The object-level contains the model that an artificial intelligent agent has for reasoning about the world (i.e. agent's environment) to solve problems [5]. The meta-level is a level of representation of the reasoning of an artificial intelligent agent [6], [7].

Introspective monitoring is a metacognitive mechanism, which is done through information feedback that is gathered at the meta-level from the object level. Introspective monitoring includes mechanisms for detecting reasoning failures at the object-level. The main purpose of monitoring is to provide enough information to make effective decisions in the meta-level control.

Some representative approach of implementation of introspective monitoring has been developped in Computational metacognition. Zilberstein [8] developed an approach for implementation of introspective monitoring maintaining statistical profiles of past metareasoning choices. Singh [9] attempted to implement the full spectrum of introspective monitoring in EM-ONE. Meta-AQUA [4], [10] performs deep understanding and reasons about the mental states. MCL [11]–[13] has cognitive mechanisms to reason about failures in its own reasoning process. However, neither MCL nor Meta-AQUA has an explicit model of self. The systems do not have a model of the contents of their background knowledge.

In the context described, the main objective of this paper is to introduce a new approach for the representation of profiles of cognitive functions. The profiles are called *algorithmic knowledge profiles*. The *algorithmic knowledge profiles* are used for introspective monitoring in cognitive agents. Declarative meta-knowledge, procedural meta-knowledge and strategic meta-knowledge are represented using the *algorithmic knowledge profile*.

The paper is structured as follows. In Section 2, we describe the *algorithmic knowledge profiles*. First, the *Internal representation of an algorithmic knowledge profile* is formalized. Next the computational implementation is described. In Section 3, we present an illustrative example of the implementation of *algorithmic knowledge profiles* in an *instructional planner agent*. Finally, conclusions and future works are presented.

II. ALGORITHMIC KNOWLEDGE PROFILE IN INTROSPECTIVE MONITORING

In this section we describe the internal representation of an *algorithmic knowledge profile,* the components of the self-model and the computational implementation using Extended Tabular-Trees.

A. *Internal representation of an algorithmic knowledge profile*

The main purpose of introspective monitoring is to provide enough information to make effective decisions in the meta-level control. Each cognitive function at the object-level has a performance profile that is continuously updated in the meta-level. The performance profile is used to evaluate the results of each cognitive function.

`ProfileGeneration` is a metacognitive task that allows the creation of profiles that contain relevant information about the reasoning processes that take place at the object-level.

In formula (1) we can see that the algorithm used by the cognitive function to process the inputs is part of the local state of the function at the metalevel. Local states of this type are called *algorithmic local states* [14], [15]. Therefore, we call *algorithmic knowledge profile* to a profile that holds the local state of a cognitive function in the form of algorithmic local states.

A profile (P) of a cognitive function (χ) consists of a data set λ, a set of algorithms α, and a feeling of confidence ν.

$$P_\chi = <\lambda, \alpha, \nu> \qquad (1)$$

Where λ represents the local state of the system with respect to a cognitive function at object-level. Data set λ consists of the set of values related to the processing and performance of the cognitive function (χ). λ=`{ID, B, E, S, C, IP, OP}`, with:

> *ID* is the identifier of the cognitive function.
>
> *B* is the time stamp of when the cognitive function was started.
>
> *E* is the time stamp of when the cognitive function is finished.
>
> S is the state of the cognitive function, $s \in S$ and `S={active, inactive}`
>
> *C* is the priority level for focus attention $c \in C$ and `C={low, medium, high}`.
>
> *IP* is the set of parameters used as input of the cognitive function.
>
> *OP* is the output of the cognitive function.

The set of algorithms α represents the behavior of a cognitive function, having as input a local state at an instant of time. Given a local state $<\lambda_i, \alpha_i, \nu_i>$ in P_χ, the following possibilities can be found: *i)* α_i=`{}`, *ii)* α_i=`{a₁}`, *iii)* $\alpha_i=\{a_1, \dots, a_n\}$, with a_i is an algorithm.

Adapting Koriat [16], the strategic choice of α is assumed to be guided in part by the agent´s subjective beliefs and subjective feelings. Each cognitive function triggers a feeling of confidence ν associated with the correctness of the decision of use α having as input λ, with $\nu \in N$ and N=`{low, medium, high}`.

B. *How does the agent knows that it knows?*

According to Halpern [14], [15], [17], an agent knows a fact φ and the agent can compute that knows φ. Knows φ means that the agent has an algorithm to decide if it knows φ. The modal operator K_χ represents the explicit knowledge that a function χ at object-level of an agent has about some fact φ.

iff $alg_{\chi}(\varphi) = \alpha$ and $data_{\chi}(\varphi) = \mathcal{L}$, then $K_{\chi}\varphi \in$ {"Yes", "No", "?"}, with $\mathcal{L} \in \lambda$. (2)

In formula (2) $K_{\chi}\,\varphi$ ="Yes" means knowledge algorithm says "Yes" to a formula φ (in a given state $<\lambda_i, \alpha_i, \nu_i>$ in P_{χ}) if the algorithm determines that the agent can compute that knows φ at the state (e. g. $\alpha_i \neq \{\}$ in P_{χ}); "No" if the algorithm determines that the agent cannot compute if know φ at the state (e. g. $\alpha_i = \{\}$ in P_{χ}); and "?" if the algorithm cannot determine whether the agent knows φ (e. g. $data_{\chi}(\varphi) \neq \mathcal{L}$ in P_{χ}).

C. *Self-model*

The meta-level keeps an updated model of the object-level called the "self-model" [2], [3], [5], [7]. This self-model is based on an internal representation of the reasoning processes that occur at the object-level. The self-model stores the data generated by the reasoning processes, along with the rules that were used to address the reasoning. The self-model allows the interchange of information between the meta-level and the object-level.

The self-model enables an intelligent system to reason about its own reasoning and knowledge, since it describes the internal state of the system. The self-model allows the meta-level to have awareness about reasoning processes that are conducted at the object-level [2], [7]. In our proposal, the set of profiles is a component of the self-model. Where each cognitive function and each strategy that is executed at the object-level has an associated profile in the self-model.

Self-model (S_m) consists of the set of elements that store information about the reasoning process at object-level. S_m = {D, T, M, J}, with:

D is the set of computational data generated by cognitive and metacognitive task.

T is the set of traces generated by reasoning and metacognitive tasks. *T* is composed of a set of pattern containing a set of goals, rules and actions performed according to an agent's mental state.

M is the set of performance profiles used to evaluate the results of each cognitive function or strategy.

Metacognitive judgments (*J*) represent assessments performed in the meta-level about events that occur in object-level. Metacognitive judgments are triggered when knowledge is acquired [10], in our approach; this is referred when the *algorithmic knowledge profile* is updated.

D. *Computational implementation of algorithmic knowledge profiles*

The implementation is based on Extended Tabular-Trees (XTT) [18], [19]. XTT aims at combining some approaches such as decision-tables and decision-trees by building a special hierarchy of Object-Attribute-Tables [19].

$A_i(\chi) = t_i$, where A_i is an attribute, χ is some cognitive function and t_i is the (current) value of attribute A_i for cognitive function χ. Each row in the table represents a rule.

According Liegeza [18] [19], a XTT takes the form presented in Tab. 1.

TABLE I. XTT REPRESENTATION

	Info	**Pre-condition**	**Algorithm**	**Assert**	**Control**
I	Ψ	$A_1 \ldots A_n$	$H_1 \ldots H_n$	$C_1 \ldots C_n$	N E
	ψ_1	$t_1 \ldots t_n$	$h_1 \ldots h_n$	$c_1 \ldots c_n$	n e
	…	…	…	…	…
	ψ_n	$t_1 \ldots t_n$	$h_1 \ldots h_n$	$c_1 \ldots c_n$	n e

The columns in the table have the following meanings:

I is the rule identifer of the rule.

Ψ is the context common for all the rules.

A_1 ... A_n are the preconditions attributes.

H_1 ... H_n are the algorithms used by the cognitive function.

C_1 ... C_n are the assert. Assert is the process of adding new facts to the knowledge base.

Control part defines the (N) next rule to be executed (if present) or the (E) else rule to be executed in case of failure.

Based on Nalepa and Liegeza [19], formally a rule is define as follows:

$$\begin{aligned}
&\textit{rule(i)}:\ \psi \wedge \\
&\quad A_1 \in t_1 \wedge A_2 \in t_2 \wedge \ldots \wedge A_n \in t_n \\
&\quad \rightarrow \\
&\quad \textit{assert}(C_1 = c_1, C_2 = c_2, \ldots, C_n = c_n) \\
&\quad \textit{strategy}\ (H_1 = h_1, H_2 = h_2, \ldots, H_n = h_n) \\
&\quad \textit{next}\ (j), \textit{else}\ (k). \qquad (3)
\end{aligned}$$

The implementation of the profiles based on XTT allows the representation of different kinds of meta-knowledge (knowledge at meta-level).

- A = {A_1, A_2, ... , A_n}, with $A_1 \in t_1 \wedge \ldots \wedge A_n \in t_n$. The set of attributes of interest of a cognitive function at object-level represents the declarative meta-knowledge.

- H = {H_1, H_2, ... , H_n}, with ($H_1=h_1$, $H_2=h_2,\ldots, H_n=h_n$). The set of algorithms associated to some state of a cognitive function represents the procedural meta-knowledge.
- *rule(i)* represents the strategic meta-knowledge.

- The *Assert* statement means the inclusion of new knowledge in the knowledge base, which is equivalent to learning new facts of any kind of knowledge

III. ILLUSTRATIVE EXAMPLE

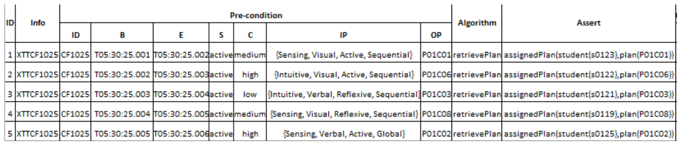
In this section we provide an example of a cognitive agent that has to act on their knowledge.

A. *Instructional planner*

Consider an *instructional planner agent* of a *tutor module* in an Intelligent Tutoring System. The *instructional planner agent* generates instructional plans, given the current state of the student, and monitoring the execution of the content plan in order to determine when to re-plan, or generate a new plan[20]–[24].

In our example, the *instructional planner agent* receives as input the student's learning style profile and generates as an output an instructional plan. To illustrate the example we use the following notation: Let η a *learning style profile* of a student and π an *instructional plan* for a student.

Fig. 1 shows a representation of algorithmic knowledge profile of the cognitive function *instructional planner*.



ID	Info	Pre-condition							Algorithm	Assert
		ID	B	E	S	C	IP	OP		
1	XTTCF1025	CF1025	T05:30:25.001	T05:30:25.002	active	medium	{Sensing, Visual, Active, Sequential}	P01C01	retrievePlan	assignedPlan(student(s0123),plan(P01C01))
2	XTTCF1025	CF1025	T05:30:25.002	T05:30:25.003	active	high	{Intuitive, Visual, Active, Sequential}	P01C06	retrievePlan	assignedPlan(student(s0122),plan(P01C06))
3	XTTCF1025	CF1025	T05:30:25.003	T05:30:25.004	active	low	{Intuitive, Verbal, Reflexive, Sequential}	P01C03	retrievePlan	assignedPlan(student(s0121),plan(P01C03))
4	XTTCF1025	CF1025	T05:30:25.004	T05:30:25.005	active	medium	{Sensing, Visual, Reflexive, Sequential}	P01C08	retrievePlan	assignedPlan(student(s0119),plan(P01C08))
5	XTTCF1025	CF1025	T05:30:25.005	T05:30:25.006	active	high	{Sensing, Verbal, Active, Global}	P01C02	retrievePlan	assignedPlan(student(s0125),plan(P01C02))

Figure 1. Algorithmic knowledge profile representation

Fig 1 shows every time the cognitive function identified with `ID` CF1025 *(instructional planner*). Function CF1025 has been executed at object level with a start time `B`, a end time `E`, a state `S` "active" and a priority level `C`. Similarly, Fig. 1 show `IP` as a set of parameters used as input of the cognitive function (in this case, *learning style profile* of the student) and `OP` as a output of the cognitive function (in this case, the *instructional plan* that has been retrieved from the knowledge base). Finally, Fig. 1 shows in the `ASSERT` field, how the instructional plan is assigned to the student and keeping it in the knowledge base.

The *instructional planner agent* ***i*** has two algorithms α_i={a$_1$, a$_2$} to generate a *instructional plan* $\boldsymbol{\pi}$. With:

> a$_1$= *retrievePlan(η, π),* if the agent finds in the knowledge base (KB) a π according to η (e. g. $\exists\pi \in$ KB: $\pi \vDash \eta$), then π is retrieved.
>
> a$_2$= *buildPlan(η, π),* if the agent cannot finds in KB a π according to η, then π is built according to η.

Different kinds of meta-knowledge (knowledge at meta-level) can be seen in Fig 1:

- Declarative meta-knowledge is represented by set of attributes $A_i = \{A_{i1}, A_{i2}, \dots, A_{in}\}$, with A_1 = {`("ID": CF1025), ("B": T05:30:25.001), ("E": T05:30:25.002), ("S": active), ("C": medium), ("IP": {Sensing, Visual, Active, Sequential}), ("OP": P01C01)`}. Each pair *(attribute, value)* represents a new meta-level belief about a cognitive function.

- Procedural meta-knowledge is represented by set of algorithms $H_i = \{H_{i1}, H_{i2}, \dots, H_{in}\}$, which, for a context A_1 = { `("ID": CF1025), ("B": T05:30:25.001), ("E": T05:30:25.002), ("S": active), ("C": medium), ("IP": {Sensing, Visual, Active, Sequential}), ("OP": P01C01)` } we have to run the algorithm *retrievePlan.* Thus, H_1 = { *retrievePlan* }.

- The following rule represents the strategic meta-knowledge contained in the first row of the algorithmic profile of the function:

```
rule(1):
        XTTCF1025 ∧
        ("ID": CF1025) ∧
        ("B":T05:30:25.001)∧
        ("E":T05:30:25.002) ∧
        ("S": active) ∧
        ("C": medium) ∧
        ("IP": {Sensing,
                Visual,
                Active,
                Sequential}) ∧
        ("OP": P01C01)
→
Assert(assignedPlan( student(s0123),
                        plan(P01C01)))
Strategy (retrievePlan)
```

In this example, `Assert` is represented by the assignment of plan `P01C01` to student `s0123` through the following sentence:

```
assignedPlan(student(s0123), plan(P01C01))
```

Let φ a fact about a cognitive function at object-level of *instructional planner agent* ***i***. With *φ=learningStyleProfile(η),* where η is the set of input of the function and π is the output of the function. For this example, $K_\chi\varphi$ represents a question about the knowledge of the *instructional planner agent:*

Given η, do you know how to generate π?

To answer this question, the *instructional planner agent* (***i***) runs an algorithm $X_i\varphi$ based in Halpern [14], where:

```
Algorithm $X_i\ \varphi$
Begin
        read $\varphi$
        case of
           if asked "does hold $\varphi$" $\wedge$ $K_\chi\varphi$ do say "Yes"
           if asked "does hold $\varphi$" $\wedge$ $K_\chi\neg\varphi$ do say "No"
           if asked "does hold $\varphi$" $\wedge$ $\neg K_\chi\varphi$ $\wedge$ $\neg K_\chi\neg\varphi$ do say "?"
        end case
end
```

The algorithm $X_i\varphi$ reads the *algorithmic knowledge profile* of the cognitive functions and tests equation (2) for each rule (e. g. for each profile row), using as inputs φ and λ_i to compute the response. The answer of the question can be: (*i*) "Yes" if it finds an algorithm in the *algorithmic knowledge profile* of the function that allows generating the π; (*ii*) "No" if it cannot find an algorithm in the profile of the function that allows generating the π; and (*iii*) "?" if the agent does not recognize φ.

We can see that the example of the *instructional planner agent* illustrates our proposal of the use of *algorithmic knowledge profiles* for introspective monitoring of knowledge an agent has about its own knowledge.

IV. Conclusions

In this article we have presented a new approach for the representation of profiles of cognitive functions. The profiles are called *algorithmic knowledge profiles*. The *algorithmic knowledge profiles* are used for introspective monitoring in cognitive agents. The computational implementation of the profiles was done using extended tabular trees (XTT).

The profiles allow to keep updated to the meta-level about the most relevant attributes of each cognitive function that is executed in the object-level

From the profiles different kinds of meta-knowledge can be represented. The declarative meta-knowledge of the process of reasoning is represented as $A_i(\chi)$. The procedural meta-knowledge is represented as the set of algorithms associated with the mental states. The strategic meta-knowledge is obtained from the rules that make up each row of the XTT.

The proposed example of the *instructional planner agent* illustrated our proposal of the use of *algorithmic knowledge profiles* for introspective monitoring of knowledge an agent has about his own knowledge. As future work, we plan the implementation of the introspective monitoring mechanisms that allow the reading of the *algorithmic knowledge profiles*.

Acknowledgment

The authors thank the Department of Educational Informatics of University of Córdoba, for the support and advice given throughout this work.

References

[1] Y. Wang, G. Baciu, Y. Yao, W. Kinsner, K. Chan, B. Zhang, S. Hameroff, N. Zhong, C.-R. Hunag, B. Goertzel, D. Miao, K. Sugawara, G. Wang, J. You, D. Zhang, and H. Zhu, "Perspectives on Cognitive Informatics and Cognitive Computing," *Int. J. Cogn. Informatics Nat. Intell.*, vol. 4, no. 1, pp. 1–29, 2010.

[2] M. Cox and A. Raja, "Metareasoning: An introduction," in *In Metareasoning: Thinking about thinking*, Cox and Raja, Ed. Cambridge. MA: MIT, 2012, pp. 1–23.

[3] J. Flavell and H. Wellman, "Metamemory," in *Perspectives on the Development of Memory and Cognition (pp. 3-33). Hillsdale, NJ:*, & J. W. H. (Eds. . R. V. Kail, Jr., Ed. Lawrence Erlbaum Associates, 1977.

[4] M. Cox and A. Ram, "Introspective multistrategy learning: On the construction of learning strategies," *Artif. Intell.*, vol. 112, no. 1, pp. 1–55, Aug. 1999.

[5] M. Caro, D. Josyula, M. Cox, and J. Jiménez, "Design and validation of a metamodel for metacognition support in artificial intelligent systems," *Biol. Inspired Cogn. Archit.*, vol. 9, pp. 82–104, 2014.

[6] M. Cox, "An Explicit Representation of Reasoning Failures," in *Case-Based Reasoning Research and Development*, 1997, pp. 211–222.

[7] R. Sun, X. Zhang, and R. Mathews, "Modeling meta-cognition in a cognitive architecture," *Cogn. Syst. Res.*, vol. 7, no. 4, pp. 327–338, 2006.

[8] S. Zilberstein and S. Russell, "Optimal composition of real-time systems," *Artif. Intell.*, vol. 82, no. 1–2, pp. 181–213, 1996.

[9] P. Singh, "EM-ONE : An Architecture for Reflective Commonsense Thinking," *Diss. Dr. Philos. Comput. Sci. Eng. - Massachusetts Inst. Technol.*, 2005.

[10] M. Cox, "Perpetual Self-Aware Cognitive Agents," *AI Magazine*, no. 2002, pp. 32–51, 2007.

[11] M. Schmill, D. Josyula, M. Anderson, S. Wilson, T. Oates, D. Perlis, D. Wright, and S. Fults, "Ontologies for Reasoning about Failures in AI Systems," in *in Proceedings from the Workshop on Metareasoning in Agent Based Systems at the Sixth International Joint Conference on Autonomous Agents and Multiagent Sytems*, 2007.

[12] M. Schmill, M. Anderson, S. Fults, D. Josyula, T. Oates, D. Perlis, H. Shahri, S. Wilson, and D. Wright, "The Metacognitive Loop and Reasoning about Anomalies," in *Metareasoning: Thinking about thinking*, M. Cox and A. Raja, Eds. Cambridge, MA: The MIT Press, 2011, pp. 183–198.

[13] M. Anderson, T. Oates, W. Chong, and D. Perlis, "The metacognitive loop I: Enhancing reinforcement learning with metacognitive monitoring and control for improved perturbation tolerance," *J. Exp. Theor. Artif. Intell.*, vol. 18, no. 3, pp. 387–411, Sep. 2006.

[14] J. Y. Halpern, "Reasoning about knowledge," *Comput. Math. with Appl.*, vol. 31, no. 2, p. 130, 1996.

[15] J. Y. Halpern and R. Pucella, "Probabilistic Algorithmic Knowledge," 2005.

[16] A. Koriat, "The feeling of knowing: some metatheoretical implications for consciousness and control," *Conscious Cogn*, vol. 9, no. 2 Pt 1, pp. 149–171, 2000.

[17] J. Halpern, Y. Moses, and M. Vardi, "Algorithmic knowledge," ... *Asp. Reason. About Knowl.*, pp. 255–266, 1994.

[18] A. Ligęza and G. J. Nalepa, "Granular logic with variables for implementation of extended tabular trees," in *Proceedings of the*

21th International Florida Artificial Intelligence Research Society Conference, FLAIRS-21, 2008.

[19] G. J. Nalepa and A. Ligęza, "A graphical tabular model for rule-based logic programming and verification," *Systems Science*, vol. 31, no. 2. pp. 89–95, 2005.

[20] J. Kim and Y. Shinn, "An Instructional Strategy Selection Model Based on Agent and Ontology for an Intelligent Tutoring System," 2010.

[21] T. Li and N. Xiong, "An Intelligence Tutoring System Based on Knowledge Grid," *2009 First Int. Work. Educ. Technol. Comput. Sci.*, pp. 998–1002, 2009.

[22] M. Caro, D. Josyula, and J. Jiménez, "Multi-level pedagogical model for the personalization of pedagogical strategies in Intelligent Tutoring Systems," *Dyna*, vol. 82(194), pp. 185–193, 2015.

[23] C. Woo, M. Evens, J. Michael, and A. Rovick, "Dynamic instructional planning for an intelligent physiology tutoring system," in *Computer-Based Medical Systems - Proceedings of the Fourth Annual IEEE Symposium*, 1991, pp. 226–233.

[24] P. Karampiperis and D. Sampson, "Adaptive instructional planning using ontologies," in *Proceedings - IEEE International Conference on Advanced Learning Technologies, ICALT 2004*, 2004, pp. 126–130.