TPLP **23** (5): 1070–1093, 2023.

doi:10.1017/S1471068422000382 First published online 20 October 2022

An Application of a Runtime Epistemic Probabilistic Event Calculus to Decision-making in e-Health Systems

FABIO AURELIO D'ASARO
Department of Human Sciences, Ethos Group, University of Verona, Verona, Italy
(*e-mail:* fabioaurelio.dasaro@univr.it)

LUCA RAGGIOLI, SALIM MALEK, MARCO GRAZIOSO and SILVIA ROSSI
Department of Electrical Engineering and Information Technologies, University of Naples Federico II, Naples, Italy
(*e-mails:* luca.raggioli@manchester.ac.uk, smalek@fbk.eu, marco.grazioso@unina.it, silrossi@unina.it)

submitted 22 June 2021; revised 29 June 2022; accepted 28 September 2022

Abstract

We present and discuss a runtime architecture that integrates sensorial data and classifiers with a logic-based decision-making system in the context of an e-Health system for the rehabilitation of children with neuromotor disorders. In this application, children perform a rehabilitation task in the form of games. The main aim of the system is to derive a set of parameters the child's current level of cognitive and behavioral performance (e.g., engagement, attention, task accuracy) from the available sensors and classifiers (e.g., eye trackers, motion sensors, emotion recognition techniques) and take decisions accordingly. These decisions are typically aimed at improving the child's performance by triggering appropriate re-engagement stimuli when their attention is low, by changing the game or making it more difficult when the child is losing interest in the task as it is too easy. Alongside state-of-the-art techniques for emotion recognition and head pose estimation, we use a runtime variant of a probabilistic and epistemic logic programming dialect of the Event Calculus, known as the Epistemic Probabilistic Event Calculus. In particular, the probabilistic component of this symbolic framework allows for a natural interface with the machine learning techniques. We overview the architecture and its components, and show some of its characteristics through a discussion of a running example and experiments.

KEYWORDS: e-health, logic programming, answer set programming, sensor fusion, motor rehabilitation

1 Introduction

In this paper, we present and discuss an architecture for integrating and reasoning about sensors in the context of the *AVATEA* project (*Advanced Virtual Adaptive Technologies e-hEAlth*). The final goal of the project is to design and implement an integrated system to support the rehabilitation process of children with *Development Coordination*

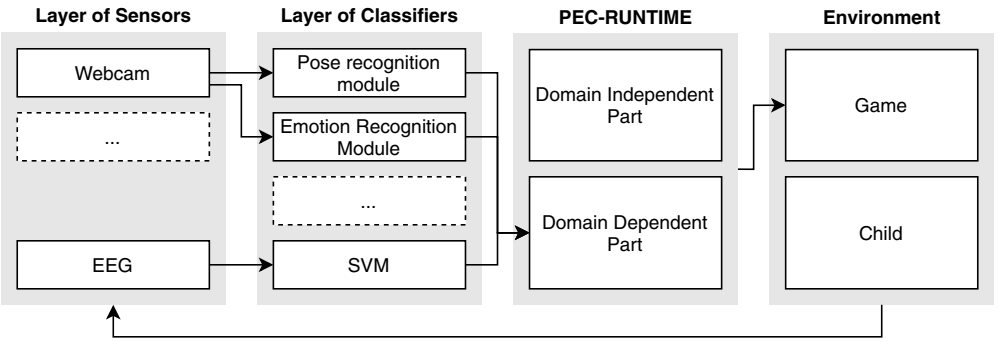


Fig. 1. The architecture of the AVATEA system. Several classifiers are applied to a stream of data from different sensors. Note that a single sensor may produce data that is then fed into two or more classifiers (e.g., in the case of the webcam). Timestamped output from the classifiers is fed into PEC-RUNTIME, the logical core of the architecture. PEC-RUNTIME processes this information together with some domain independent axioms and outputs its decision to the environment (the game, in this case).

Disorders (*DCD*s). The system needs to integrate and control several components, including an adjustable seat, various types of sensors, and an interactive visual interface to perform rehabilitation exercises in the form of games (such games are sometimes called *exergames*, presented by Vernadakis *et al*. 2015). One of the main goals of the project is to automatize the therapeutic task, ideally without making it less effective. To this end, we employ a logic-based AI engine that collects data from the environment and decides what strategy may be implemented to make the video game as challenging as possible while keeping the child engaged and attentive. For example, the engine may decide to emit a sound if the visual attention of the user is detected to be low, or increase the video game's difficulty level if the sensors seem to indicate that the child is bored as s/he is performing the task effortlessly.

Although this architecture has been specifically designed for the rehabilitation of children with neuro-motor disorders, it can be more generally applied to any task requiring a system to take runtime decisions according to a stream of sensorial data. It mainly consists of four modules communicating with each other (see Figure 1). Layers of sensors and classifiers (e.g., a webcam paired with head pose and emotion recognition algorithms) collect and process information about the current state of the environment and the child undergoing the rehabilitation process. This information is fed to a probabilistic logic programming system, a modified version of the *Epistemic Probabilistic Event Calculus* (*EPEC* for short, first introduced by D'Asaro *et al.* D'Asaro *et al*. 2017 as a non-epistemic framework under the name PEC and extended in D'Asaro *et al*. 2020 to the epistemic case). To guarantee good performance at runtime, in this work, we present a novel implementation of EPEC, dubbed *PEC-RUNTIME*, which implements a form of *progression* (Lin and Reiter 1997). It is worth noting here that, unlike EPEC, PEC-RUNTIME has a non-epistemic nature. However, since our application domain needs epistemic actions (namely, getting information from sensors), we use it to approximate the behavior of EPEC based on the correspondence between epistemic and non-epistemic action outlined in Section 2.3. PEC-RUNTIME processes the sensor data and takes decisions

https://doi.org/10.1017/S1471068422000382 Published online by Cambridge University Press

according to a predefined strategy. Finally, the gaming platform actuates these decisions and communicates the new state of the game to the layer of sensors.

EPEC (and consequently PEC-RUNTIME) is particularly well suited for this task as its probabilistic nature facilitates the communication with probabilistic machine learning-based classifiers. As it is a symbolic framework, it can also be used to provide accurate human-understandable reports of the user activity at the end of each therapeutic session. Typical feedback includes information about attention levels of the child (e.g., "The user was highly engaged for 36 s during the session"), justification for the decisions taken (e.g., "Switched to higher difficulty level as the user has been carrying out the exercise correctly for the last 10 s"), as well as a complete report of what happened throughout the therapeutic session (e.g., "The user was not visible at 15:43:23", "Visual stimulus presented at 15:43:24", etc.) and a series of graphs for accurate tracking of user activity.

This paper extends (D'Asaro *et al.* 2019) and is organized as follows. In Section 2, we overview the logical part of our architecture, EPEC, and show some of its features through the discussion of a toy example. In Section 3, we discuss how we adapted the original ASP-based inference mechanism of EPEC in order to work at runtime. In Section 4, we briefly describe the layers of sensors and classifiers. In Section 5, we demonstrate our approach in our use case and provide a systematic empirical evaluation. In Section 6 we discuss related work, in particular probabilistic logic programming languages for reasoning about actions and gamification techniques. Some final remarks about directions for further work conclude the paper in Section 7.

2 Overview of the language

In this section, we overview a subset of EPEC[1] that is relevant to our application, and show how it can be used to represent a domain. Similarly to other logical languages for reasoning about actions, EPEC models a domain as a collection of *fluents* and *actions*. Moreover, EPEC represents time explicitly through *instants*. Fluents, actions and instants constitute the principal sorts of EPEC. Actions are further sub-divided into *agent actions* (under the control of the agent being modeled) and *environmental actions* (performed by the environment). Fluents can take value in a set of *values*. Domain-specific theories can be designed using EPEC propositions. As an illustration, we use the following toy domain (inspired by the AVATEA use case) as our running example (but you can see e.g., Acciaro *et al.* 2021 for other use cases):

Scenario 2.1
A child undergoes a simple attention test in which her level of visual attention is measured. The task consists in tracking a moving object on a screen. Therapists agree that when the child is not looking at the object, there is 90% chance this is due to lack of attention, and that when the child is looking at the object there is 100% chance s/he is paying attention to the task. A webcam is used to track her gaze. Readings from the webcam are only partly reliable due to hardware limitations, and are produced every second. These readings are then piped into a specialized classifier which decides whether

[1] The fully fledged framework was introduced in D'Asaro *et al.* (2020), and has some minor differences from the version presented here.

https://doi.org/10.1017/S1471068422000382 Published online by Cambridge University Press

the child is looking at the object on the screen or not. The classifier also outputs a confidence level for its classification. When the child is not looking at the screen, it may be useful to play a sound in order to re-engage the child. Two seconds after the start of the experiment, the child has been detected to be tracking the object on the screen with confidence levels 0.25 and 0.13. What is the level of attention of the child at instant 2? Should the sound be played at time 2?".

We let the set of instants in our timeline be $\{0, 1, 2\}$, whose elements can be naturally interpreted as seconds since the start of the experiment.

2.1 Syntax overview

The only fluent in our example scenario is *Attention*. It is *boolean*, that is, it may take values $\top$ and $\bot$. In EPEC, this is written

$$\mathit{Attention} \textbf{ takes-values } \{\top, \bot\}. \tag{1}$$

In the example, we consider the agent to be an automated system that is responsible for triggering a re-engagement strategy when necessary. On the other hand, the child is considered part of the environment. Therefore, the actions are *PlaySound* and *TrackObject*, where *PlaySound* is the agent action of playing a sound and *TrackObject* is the environmental action which corresponds to the child tracking the object with her eyes. The effects of *not* tracking the object are defined by the following proposition:

$$\neg \mathit{TrackObject} \textbf{ causes-one-of } \{(\{\neg \mathit{Attention}\}, 0.9), (\emptyset, 0.1)\}. \tag{2}$$

Looking at the object is never considered to be due to chance, therefore we also include the following proposition:

$$\mathit{TrackObject} \textbf{ causes-one-of } \{(\{\mathit{Attention}\}, 1)\}. \tag{3}$$

For what regards action *PlaySound* under the control of the agent, we want to formalize the *conditional plan* stating that this action must be performed at instant 2 if the child is believed to be distracted, that is, if the probability of *Attention* has fallen below a predefined threshold, say 0.1. This can be written in EPEC as:

$$\mathit{PlaySound} \textbf{ performed-at } 2 \textbf{ if-believes } (\mathit{Attention}, [0, 0.1]). \tag{4}$$

We consider the child to be initially paying full attention to the task. This translates to:

$$\textbf{initially-one-of } \{(\{\mathit{Attention}\}, 1)\}. \tag{5}$$

Finally, the action *TrackObject* occurs twice at instants 0 and 1 with confidence 0.25 and 0.13 respectively. This translates to the following propositions:

$$\mathit{TrackObject} \textbf{ occurs-at } 0 \textbf{ with-prob } 0.25 \tag{6}$$

$$\mathit{TrackObject} \textbf{ occurs-at } 1 \textbf{ with-prob } 0.13. \tag{7}$$

In EPEC, Proposition (1) is known as a *v-proposition* (*v* for "value"), Propositions (2) and (3) are known as *c-propositions* (*c* for "causes"), Proposition (4) is known as a *p-proposition* (*p* for "performs"), Proposition (5) is known as an *i-proposition* (*i*

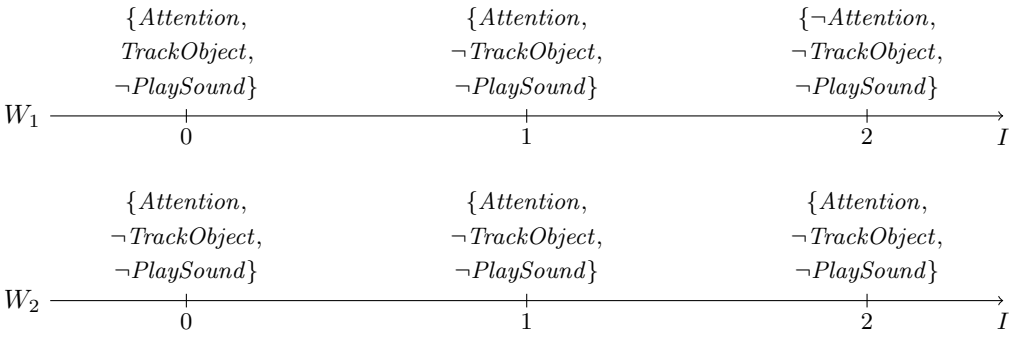
https://doi.org/10.1017/S1471068422000382 Published online by Cambridge University Press

for "initially"), Propositions (6) and (7) are known as *o-propositions* (*o* for occurs). Propositions (4), (6) and (7) are known as the *narrative* part of the domain.

The set of propositions $\{(1), \ldots, (7)\}$ constitutes what in EPEC is called a *domain description* and is usually denoted by $\mathscr{D}$ (with appropriate subscripts and/or supercripts). The narrative part of a domain is denoted by $narr(\mathscr{D})$. A domain description is given meaning using a bespoke semantics.

2.2 Semantics overview

The semantics of EPEC enumerates all the possible evolutions (*worlds* in the terminology of EPEC) of the environment being modeled, starting from the initial state. It then applies a series of standard reasoning about actions principles (e.g., persistence of fluents, closed world assumption for actions) to filter out those evolutions of the environment that do not make intuitive sense with respect to the given domain. The remaining worlds, dubbed *well-behaved worlds* to indicate that they are meaningful with respect to the given domain description, are assigned a probability value. Some well-behaved worlds for our example domain can be depicted as follows:



World W_1 represents a possible evolution of the environment starting from a state in which the child is initially (at instant 0) fully attentive and tracking the object with her eyes. At instant 1, the child is still paying attention to the task, but he/she is no longer tracking the object. At instant 2, the child is distracted and again not tracking the object. The sound is never played. EPEC's semantics assigns a probability of 0.19575 to this world, as this results from the product of 0.25 (due to proposition (6) and *TrackObject* occurring at 0 in W_1), $1 - 0.13$ (due to proposition (7) and *TrackObject* not occurring at 1 in W_1) and 0.9 (due to proposition 2 and since not tracking the object at instant 1 caused *Attention* not to hold at instant 2). The intuitive meaning of world W_2 and its probability can be figured similarly. In W_2 the child is always attentive but never tracking the object, the sound is never played, and its probability is 0.006525.

There are 8 well-behaved worlds with respect to our example domain, and they are such that their probabilities sum up to 1. Formal results about the semantics of EPEC guarantee that this is always the case for any domain description and the corresponding set of well-behaved worlds.

We now want to address the question "*What is the level of attention of the child at instant 2?*". EPEC first needs to *query* the domain description about the probability

https://doi.org/10.1017/S1471068422000382 Published online by Cambridge University Press

of fluent *Attention* at instant 2. In EPEC, queries are expressed in the form of an *i-formula* (i.e., a *formula* with instants attached). In our case, the query has the form $Q_1 = [Attention]@2$, but more complicated queries are also possible, for example $Q_2 = ([\neg Attention]@1 \vee [\neg TrackObject]@1) \wedge [Attention]@2$. We say that i-formula Q_1 *has instant* 2 (as this is the only instant appearing in the formula) and that the i-formula Q_2 has instants > 0 (as all the instants appearing in the formula are strictly greater than 0). Answering query $[Attention]@2$ amounts to summing the probabilities of those well-behaved worlds in which the i-formula is true, that is, such that *Attention* is true at instant 2. This results in 0.158275, and can also be written as:

$$\mathscr{D} \models [Attention]@2 \textbf{ holds-with-prob } 0.158275.$$

Note that this value falls outside the interval $[0, 0.1]$ in the precondition of Proposition (4). Therefore, EPEC deduces that the answer to our last question in Scenario 2.1 ("*Should the sound be played at time* 2*?*") is *no*. Then, Proposition (4) does not *fire* and the action *PlaySound*, under the control of the agent, is not played in any of the well-behaved worlds.

2.3 Simulating epistemic actions via c-propositions and o-propositions

The reader may have noticed that our target application domain mainly deals with sensors. In its original formulation (D'Asaro *et al*. 2020), epistemic modeling is reserved to an additional, type of propositions, called *s-propositions* (*s* for "senses"), of the following form:

$$S \textbf{ senses } F \textbf{ with-accuracies } M \tag{8}$$

for an action S, a fluent F and some matrix M representing the accuracy of A when sensing F. Given their role within EPEC, we would need s-propositions to model sensors in our domain. However, s-propositions add a layer of complexity that in this work we would like to avoid due to the necessity of taking decisions at runtime. In this section, we show a possible translation procedure of s-propositions into pairs composed by a c-proposition and an o-proposition. This aims to demonstrate that one can make "improper" use of c-propositions and o-propositions (that are typically used for effectors) as a means to model epistemic aspects of a domain. Whether a sound and complete translation is available remains however a subject for future work.

To illustrate, consider a simple domain description in which an agent (imperfectly) tests twice for *Flu*:

$$Flu \textbf{ takes-values } \{\top, \bot\} \tag{9}$$

$$\textbf{initially-one-of } \{(\{Flu\}, 0.7), (\{\neg Flu\}, 0.3)\} \tag{10}$$

$$Test \textbf{ senses } Flu \textbf{ with-accuracies } \begin{pmatrix} 0.8 & 0.2 \\ 0.4 & 0.6 \end{pmatrix} \tag{11}$$

$$Test \textbf{ performed-at } 0 \tag{12}$$

$$Test \textbf{ performed-at } 1 \tag{13}$$

which means that the *Test* has the effect of producing knowledge about whether the agent has *Flu* with associated confusion matrix $\begin{pmatrix} 0.8 & 0.2 \\ 0.4 & 0.6 \end{pmatrix}$, where 0.8 is the true positives rate

https://doi.org/10.1017/S1471068422000382 Published online by Cambridge University Press

and 0.6 is the true negatives rate. The domain description above entails the following *b-proposition* (*b* for "believes"):

$$\textbf{at } 2 \textbf{ believes } [Flu]@0 \textbf{ with-probs} \tag{14}$$

$$\{(\langle\{Test, ((Test, Flu), false)\}@0, \{Test, ((Test, Flu), false)\}@1\rangle, 0.136, 0.206), \tag{15}$$

$$(\langle\{Test, ((Test, Flu), false)\}@0, \{Test, ((Test, Flu), true)\}@1\rangle, 0.184, 0.609), \tag{16}$$

$$(\langle\{Test, ((Test, Flu), true)\}@0, \{Test, ((Test, Flu), false)\}@1\rangle, 0.184, 0.609), \tag{17}$$

$$(\langle\{Test, ((Test, Flu), true)\}@0, \{Test, ((Test, Flu), true)\}@1\rangle, 0.496, 0.903)\}. \tag{18}$$

This proposition may be interpreted as saying that an agent sitting at instant 2, having tested for *Flu* twice, is in one of the four following knowledge states:

- The test came up negative twice (as in the case of (15)). Given the initial knowledge about the distribution of *Flu*, this is likely to happen with probability 0.136. In this case, the agent will believe that the probability of $[Flu]@0$ decreased from the initial value 0.7 to 0.206.
- The test came up negative at instant 0 and positive at instant 1 (as in the case of (16)). This is likely to happen with probability 0.184, and in this case the agent will believe that the probability of $[Flu]@0$ is 0.608.
- The test came up positive at instant 0 and negative at instant 1 (as in the case of (17)). This is likely to happen with probability 0.184, and in this case the agent will believe that the probability of $[Flu]@0$ is 0.608.
- The test came up positive twice (as inthe case of (18)). This is likely to happen with probability 0.496, and in this case the agent will believe that the probability of $[Flu]@0$ is 0.903.

This syntax shows the epistemic core of EPEC, which allows one to reasoning about *past, present and future*: for example, note that in (14) we reason about the perspective of an agent at instant 2 who is reasoning about whether it *had Flu* at instant 0 in the past. This powerful syntax comes at expenses of computational efficiency, and therefore in this work we adopt a number of simplifications due to the fact that we only need to reason about the present. The first of such simplifications is that we drop s-propositions and simulate them via o-propositions and c-propositions, in a way that we now aim to make clearer.

Assume that the agent modeled in the domain description above tests positive twice. Then, we may note that substituting the s-proposition (11) and the two p-propositions (12) and (13) with

$$Test \textbf{ causes } \{(\{Flu\}, 1)\} \tag{19}$$

$$Test \textbf{ occurs-at } 0 \textbf{ with-prob } 0.412 \tag{20}$$

$$Test \textbf{ occurs-at } 1 \textbf{ with-prob } 0.452 \tag{21}$$

yields

$$\mathscr{D} \models [Flu]@2 \textbf{ holds-with-prob } 0.903 \tag{22}$$

which matches with the result in (18), and therefore we may say that *simulates* it. Although a formal characterization of the translation from s-proposition to c-propositions and o-propositions is beyond the scope of this work, note that an s-proposition for a

https://doi.org/10.1017/S1471068422000382 Published online by Cambridge University Press

boolean fluent F of the form

$$S \textbf{ senses } F \textbf{ with-accuracies } \begin{pmatrix} a & 1-a \\ b & 1-b \end{pmatrix} \tag{23}$$

is simulated either by a c-proposition

$$S^{+} \textbf{ causes-one-of } \{(\{F\}, 1)\} \tag{24}$$

and an o-proposition

$$S^{+} \textbf{ occurs-at } I \textbf{ with-prob } \frac{p(a-b)}{p(a-b)+b} \tag{25}$$

if S is performed at I and produces a positive result, or by a c-proposition

$$S^{-} \textbf{ causes-one-of } \{(\{\neg F\}, 1)\} \tag{26}$$

and an o-proposition

$$S^{-} \textbf{ occurs-at } I \textbf{ with-prob } \frac{(p-1)(a-b)}{p(a-b)+b-1} \tag{27}$$

if S is performed at I and produces a negative result, where p is the probability such that

$$\mathscr{D} \mid\!\models [F]@I \textbf{ with-prob } p$$

in the domain $\mathscr{D}$ under consideration. This informally shows that it is possible to meaningfully interpret c-propositions and o-propositions as epistemic actions in place of s-propositions, and justifies their employment to perform sensing.

3 A runtime adaptation of EPEC

The subset of EPEC introduced in Section 2 is sufficient to handle domains that require limited epistemic functionalities. For this special class of domains, one can use the non-epistemic framework to *approximate* the behavior of EPEC and efficiently compute probabilities of queries at runtime, at the expense of the ability to reason about the past. These assumptions are not restrictive for our use case, as we only need to reason about the present state of the world and possibly react at runtime. In this section, we provide details about an implementation, dubbed *PEC-RUNTIME*, that exploits these restrictions to provide a fast reasoning system that overcomes the limitations of other implementations of EPEC in terms of computation time. PEC-RUNTIME is publicly available.[2]

PEC-RUNTIME is based on *PEC-ASP*, an Answer Set Programming implementation of the non-epistemic fragment of EPEC that was presented in D'Asaro *et al.* (2017) and based on ASP grounder and solver clingo (Gebser *et al.* 2014). PEC-ASP and PEC-RUNTIME share the same syntax and domain-independent part of the implementation. In addition, PEC-RUNTIME exploits clingo's integration with Python to control the grounding and solving process.

[2] https://gitlab.com/fdasaro/pec-runtime.

https://doi.org/10.1017/S1471068422000382 Published online by Cambridge University Press

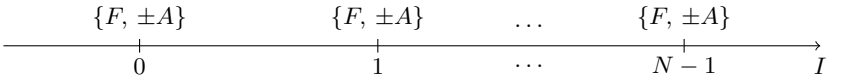
One of the main problems of PEC-ASP is that it is an *exact* inference mechanism, which needs to enumerate all the well-behaved worlds that satisfy a given query. The number of worlds typically grows exponentially with the size of the narrative. As an illustration, consider the simple domain consisting of one fluent F, one action A, set of instants $\{0, 1, \ldots, N-1\}$ for some $N > 0$, the i-proposition

$$\textbf{initially-one-of}\ \{(F, 1)\} \tag{28}$$

and N o-propositions

$$A\ \textbf{performed-at}\ I\ \textbf{with-prob}\ 0.5 \tag{29}$$

one for each I in the set of instants. This simple domain description has well-behaved worlds of the following form:



where $\pm A$ is either A or its negation $\neg A$. Clearly, there are 2^N such worlds. Given that F is always true in all of them, PEC-ASP needs to enumerate all 2^N well-behaved worlds even to answer a simple query such as $[F]@N-1$. A strategy to tackle this problem is to sample a number $M \ll 2^N$ of well-behaved worlds and approximate the probability of the query. This is the approach adopted by *PEC-ANGLICAN*, a probabilistic programming implementation of PEC.[3] However, this technique also suffers from scalability and precision issues when dealing with large narratives as we describe in Section 5.1, Experiment (a), and is mostly suited for offline tasks.

The approach we adopt here relies on a form of *progression* (Lin and Reiter 1997). Given a domain description $\mathscr{D}$ and an instant I, knowledge about what happened before instant I can be appropriately recompiled and mapped to a new domain description $\mathscr{D}_{\geq I}$ that does not include any narrative knowledge about instants $< I$ but agrees with the original domain $\mathscr{D}$ on all queries about instants $\geq I$. This requires *exhaustively* querying $\mathscr{D}$ about the state of the environment at I, that is, if $F_1, \ldots, F_n$ are all the (boolean) fluents in the language, one must perform queries $[\pm F_1 \wedge \cdots \wedge \pm F_n]@I$ where $\pm F_i$ is either F_i or its negation $\neg F_i$. These queries must be then recompiled into an appropriate i-proposition.

We elaborate this procedure using our Scenario 2.1. Consider the domain description $\mathscr{D}$ and instant 1. Since this domain has only one fluent, exhaustively querying $\mathscr{D}$ at 1 means finding the probabilities P_1 and P_2 such that

$$\mathscr{D} \mid\!\models [\mathit{Attention}]@1\ \textbf{holds-with-prob}\ P_1 \tag{30}$$

$$\mathscr{D} \mid\!\models [\neg \mathit{Attention}]@1\ \textbf{holds-with-prob}\ P_2. \tag{31}$$

These values are $P_1 = 0.325$ and $P_2 = 1 - P_1 = 0.675$ (as it can be calculated, e.g., by using PEC-ASP) and can be recompiled into the following i-proposition:

$$\textbf{initially-one-of}\ \{(\{\mathit{Attention}\}, 0.325), (\{\neg \mathit{Attention}\}, 0.675)\} \tag{32}$$

[3] https://github.com/dasaro/pec-anglican.

https://doi.org/10.1017/S1471068422000382 Published online by Cambridge University Press

The domain description $\mathscr{D}_{\geq 1}$ consists of propositions (32), (1), (2), (3), (4) and (7). Since the idea is that of discarding all knowledge about instants before 1, $\mathscr{D}_{\geq 1}$ has corresponding domain language with set of instants $\{1, 2\}$ (i.e., instant 0 was removed from the language). Domain descriptions $\mathscr{D}$ and $\mathscr{D}_{\geq I}$ agree on all queries with instants $\geq I$: for example, it is possible to show that $M_{\mathscr{D}_{\geq 1}}([Attention]@2) = 0.158275$, and we have already calculated it is also the case that $M_{\mathscr{D}}([Attention]@2) = 0.158275$. Since domain description $\mathscr{D}_{\geq 1}$ has fewer o-propositions than $\mathscr{D}$, it also has fewer well-behaved worlds (namely 5). In turn this intuitively means that, if a query only contains instants ≥ 1, it is computationally more convenient to query $\mathscr{D}_{\geq 1}$ rather than $\mathscr{D}$.

Repeating the process on $\mathscr{D}$ and instant 2 produces the i-proposition

$$\textbf{initially-one-of}\ \{(\{Attention\}, 0.158275), (\{\neg Attention\}, 0.841725)\} \tag{33}$$

and the domain description $\mathscr{D}_{\geq 2}$ consisting of propositions (33), (1), (2), (3) and (4). This new domain description has only 2 well-behaved worlds. Clearly, $M_{\mathscr{D}_{\geq 2}}([Attention]@2) = 0.158275$ as this is encoded directly in the i-proposition, and this also equals $M_{\mathscr{D}}([Attention]@2)$ and $M_{\mathscr{D}_{\geq 1}}([Attention]@2)$ as we have already calculated.

Note that the domain description $\mathscr{D}_{\geq 2}$ trivializes the task of deciding whether Proposition (4) fires or not. In fact, it suffices to check whether its epistemic precondition is *satisfied* or not by the i-proposition in $\mathscr{D}_{\geq I}$. In this case, the epistemic precondition requires *Attention* to be in the interval $[0, 0.1]$, which is not the case as it can be immediately seen by looking at the i-proposition (33) and considering that $0.158275 \notin [0, 0.1]$.

As we show below, we can turn this intuition into an efficient procedure for updating a domain description as new events are received and reason about this smaller domain, instead of augmenting it with new propositions and reason about the full augmented domain. The pseudo code of our PEC-RUNTIME procedure is as follows:

Algorithm 1: *PEC-RUNTIME*(*Domain Description* $\mathscr{D}$, *Classifiers* C)

```
I = 0;
while session is not over do
    currentIProp ← constructIProp(exhaustivelyQuery(𝒟, I));
    𝒟 ← 𝒟 \ iprop(𝒟) \ narr(𝒟) ∪ {currentIProp};
    for p-proposition p in 𝒟 do
        if p is satisfied in iprop(𝒟) then
            Execute p's postconditions;
        end
    end
    currentNarrative ← generateEventsFrom(C, I);
    𝒟 ← 𝒟 ∪ currentNarrative;
    outputNarrative ← outputNarrative ∪ currentNarrative;
    I ← I + 1;
end
return outputNarrative
```

The correctness of this algorithm is guaranteed by the following proposition.

https://doi.org/10.1017/S1471068422000382 Published online by Cambridge University Press

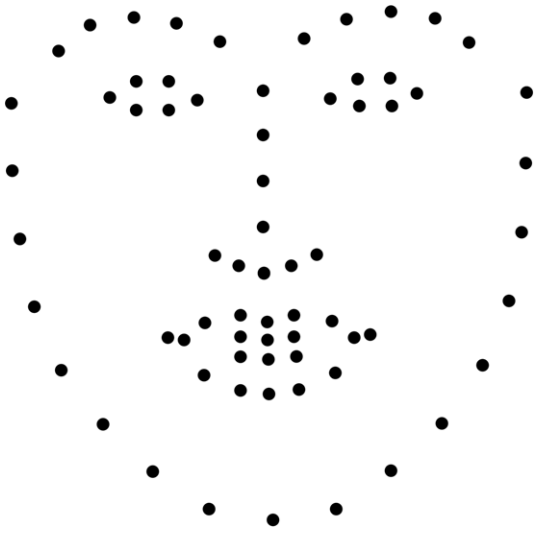


Fig. 2. Facial landmarks detected by the Head Pose Recognition module.

Proposition 3.1
Let $\mathscr{D}$ be any domain description such that $\mathscr{D} = \mathscr{D}_{\leq 0}$. Then, $M_{\mathscr{D}_{>0}}(\varphi) = M_{\mathscr{D}}(\varphi)$ for any i-formula φ having instants > 0.

Proof
See Appendix B □

4 Sensors and classifiers

In this section, we briefly overview the layers of sensors and classifiers we currently use to detect user activity that is relevant to the rehabilitation task.

At the present stage, our main source of information is a 2D-webcam placed in front of the children performing the rehabilitation task. The camera stream is mainly responsible for the evaluation of the current pose of head and shoulders of the child, and to estimate his/her emotional response to the interaction. Its output is processed by three modules:

Head Pose Recognition Module: We use the library *Head Pose Estimation*[4] to detect the facial landmarks (see Figure 2). A box containing the face and its surrounding area is selected, resized, and normalized in order to use it as input for the facial-landmarks detector, which is based on a CNN architecture. The facial-landmarks detection step is carried out by a custom trained facial landmark detector based on TensorFlow and trained on the iBUG datasets.[5] It outputs 68 facial landmarks (2D points) as in Figure 2. In turn, these landmarks are used to estimate the direction of gaze and quality of head posture (Malek and Rossi 2021).

[4] https://github.com/yinguobing/head-pose-estimation
[5] https://ibug.doc.ic.ac.uk/.

https://doi.org/10.1017/S1471068422000382 Published online by Cambridge University Press

Shoulders Alignment Module: We use *PoseNet* (Papandreou *et al*. 2018) to extract the skeleton joints coordinates from the camera frames. Then we compare these coordinates to the horizontal plane. We use this estimation to determine whether the child is assuming a correct posture while performing the rehabilitation.

Emotion Recognition Module: This module estimates the emotional response of the child, and outputs a *valence* (Rescigno *et al*. 2020). This value represents how good - or how bad - the emotion associated with the event or the situation was, and ranges from -1 to $+1$. To this end we use the *AlexNet* model (Krizhevsky *et al*. 2012) trained on the *Affectnet* Dataset (Mollahosseini *et al*. 2017).

In future extensions of this work we intend to combine data gathered with additional sensory sources. Specifically, we will employ pillows equipped with pressure sensors, mounted on the seat-back of the chair, to complement the evaluation of the Shoulders Alignment Module and better evaluate whether the child is sitting correctly. We also plan on complementing the Head Pose and Emotion Recognition Modules using an Eye Tracker and EEG respectively.

5 Examples and discussion

In this section we demonstrate our approach by means of examples and experimental results. Note that Experiment (a) was run on a Mid-2010 Apple MacBook Core 2 Duo 2.4 GHz with 8 GB of RAM, while Experiments (b), (c), (d), (e) and (f) were run on an Apple Macbook Pro 2020 M1 with 16 GB of RAM.

5.1 Scalability

Experiment (a). To demonstrate that PEC-RUNTIME scales favorably compared with PEC-ASP and PEC-ANGLICAN we tested the three implementations on a simple domain description in which an action repeatedly occurs and causes the probability of a fluent to slowly decay:

$$F \textbf{ takes-values } \{\top, \bot\}$$
$$\textbf{initially-one-of } \{(\{F\}, 1)\}$$
$$A \textbf{ causes-one-of } \{(\{\neg F\}, 0.2), (\emptyset, 0.8)\}$$
$$\forall I,\ A \textbf{ performed-at } I \textbf{ with-prob } 0.5$$

where the set of instants is $\{0, 1, \ldots, 15\}$. The probability of $\neg F$ as calculated by the three frameworks is shown in Figure 3. Table 1 summarizes the performances of PEC-RUNTIME, PEC-ASP and PEC-ANGLICAN.

As it can be seen from these results, PEC-RUNTIME outperforms both PEC-ASP and PEC-ANGLICAN on all queries. In terms of performance, PEC-ANGLICAN is close to PEC-RUNTIME when only 100 well-behaved worlds are sampled. However, sampling 100 worlds causes precision issues as it is clear from Figure 3.

Experiment (b). In this experiment we consider the effect of varying the number of actions in the domain description. To isolate the effect of actions from that of other side-effects we consider domain descriptions of the simplest possible form:

https://doi.org/10.1017/S1471068422000382 Published online by Cambridge University Press

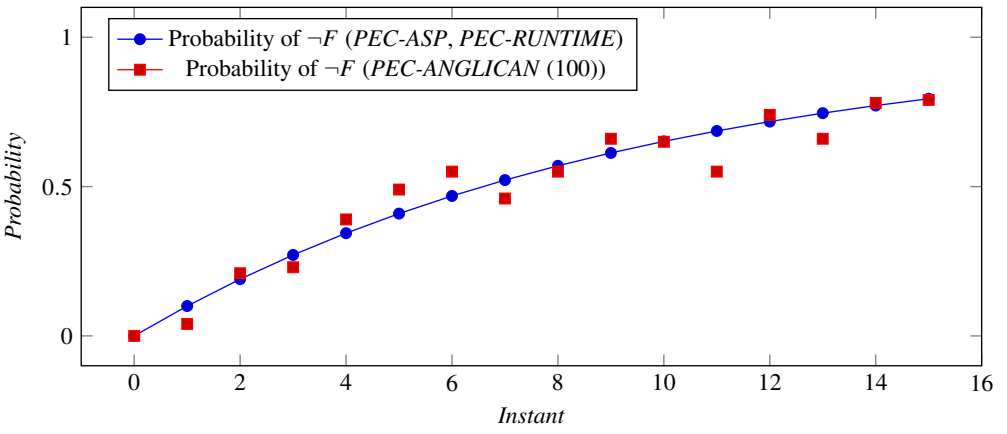


Fig. 3. Probability of F as a function of time for the example discussed in Section 5.1, Experiment (a).

$$F \textbf{ takes-values } \{\top, \bot\}$$
$$\textbf{initially-one-of } \{(\{F\}, 1)\}$$
$$\forall I,\ A_1 \textbf{ performed-at } I \textbf{ with-prob } 0.5$$
$$\vdots$$
$$\forall I,\ A_n \textbf{ performed-at } I \textbf{ with-prob } 0.5$$

where $\mathscr{I} = \{1, \ldots, 15\}$ and we consider different values of n ranging from 0 to 15. We evaluate the computation time for query $[F]@I$ at all instants. Results are plotted in Figure 4 and confirm the intuition that, at an instant I, computation time scales exponentially with the number of action occurrences at that instant.

Experiment (c). To empirically evaluate the effect of the number of fluents in the domain, we performed a similar experiment with domain descriptions of the following forms:

$$F_1 \textbf{ takes-values } \{\top, \bot\}$$
$$\vdots$$
$$F_n \textbf{ takes-values } \{\top, \bot\}$$
$$\textbf{initially-one-of } \{(\{F_1, \ldots, F_n\}, 1)\}$$

where $\mathscr{I} = \{1, \ldots, 15\}$ and n ranges from 1 to 15. Results are shown in Figure 4, and confirm the intuition that computational time scales exponentially as a function of the number of fluents in the domain.

Experiment (d). This experiment combines (b) and (c) in that it shows how computation time scales with both fluents and actions. The considered domain descriptions are:

$$F_1 \textbf{ takes-values } \{\top, \bot\}$$
$$\vdots$$
$$F_n \textbf{ takes-values } \{\top, \bot\}$$

https://doi.org/10.1017/S1471068422000382 Published online by Cambridge University Press

Table 1. *Time (in seconds) to execute the query* $[\neg F]@I$ *in the example discussed in Section 5.1. The numbers in bracket in the case of PEC-ANGLICAN refer to the number of sampled well-behaved worlds used to approximate the result of the query. For every implementation, reported times include grounding and processing of the domain description*

Instant I	PEC-ASP	PEC-ANG (100)	PEC-ANG (1000)	PEC-ANG (10000)	PEC-RUNTIME
0	0.01	1.55	4.52	29.29	0.07
1	1260.24	1.58	3.23	29.02	0.19
2	2142.81	1.64	4.87	31.54	0.18
3	2557.31	1.58	5.46	33.04	0.17
4	2708.37	1.61	3.22	28.66	0.17
5	2911.29	1.61	3.69	32.92	0.22
6	3086.32	1.79	3.58	35.62	0.13
7	3147.04	1.56	3.70	26.98	0.22
8	3208.48	1.55	5.18	33.00	0.14
9	3251.10	1.53	3.33	32.79	0.19
10	3298.81	1.53	5.41	32.95	0.18
11	3316.88	1.60	4.66	28.72	0.13
12	3311.27	1.57	3.88	27.99	0.21
13	3338.23	1.55	1.50	27.62	0.18
14	3440.68	1.56	4.05	23.97	0.15
15	3368.67	1.80	4.65	24.85	0.13
Average:	2771.72	1.60	4.06	29.93	0.17

$$\textbf{initially-one-of}\ \{(\{F_1,\ldots,F_n\},1)\}$$
$$\forall I,\ A_1\ \textbf{performed-at}\ I\ \textbf{with-prob}\ 0.5$$
$$\vdots$$
$$\forall I,\ A_m\ \textbf{performed-at}\ I\ \textbf{with-prob}\ 0.5$$

where $\mathscr{I} = \{1,\ldots,15\}$, the number of fluents varies from 1 to 15 and the number of actions varies from 0 to 15. Result are shown in Figure 5 and extend those of Figure 4.

Experiment (e). In this experiment we empirically evaluate the effect of initial conditions, and show that computation time scales linearly with their number. To this aim, we consider the following domain description:

$$F_1\ \textbf{takes-values}\ \{\top,\bot\}$$
$$\vdots$$
$$F_{10}\ \textbf{takes-values}\ \{\top,\bot\}$$
$$\textbf{initially-one-of}\ \{(\{F_1,\ldots,F_{10}\},1/n),\ldots,(\{\neg F_1,\ldots,F_{10}\},1/n),$$
$$(\{\neg F_1,\ldots,\neg F_{10}\},1/n)\}$$

where $\mathscr{I} = \{1,\ldots,15\}$ and n ranges from 1 to 2^{10}, that is, the maximum number of possible initial conditions with 10 fluents. Note that n is also the number of well-behaved worlds considered by PEC-RUNTIME at each progression step. Results are shown in

https://doi.org/10.1017/S1471068422000382 Published online by Cambridge University Press

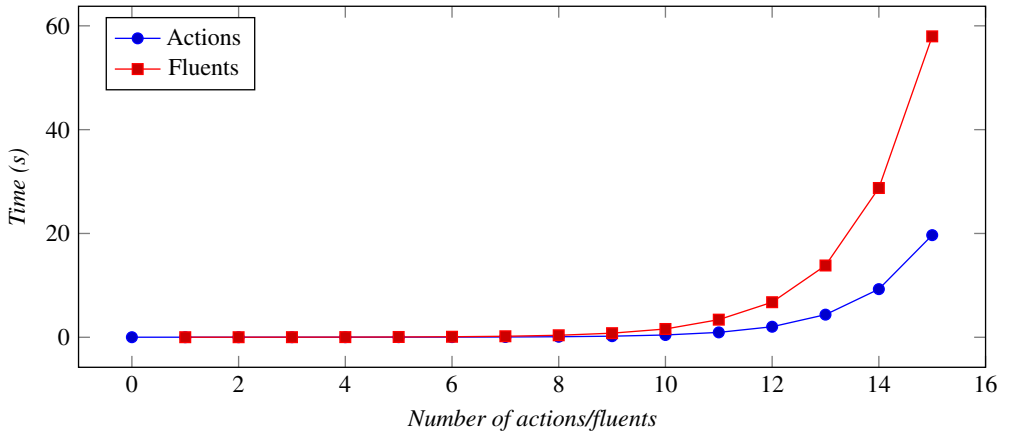


Fig. 4. Time (in seconds) to query the domain in Section 5.1, Experiments (b) and (c), expressed as a function of the number of actions and fluents. The results show averages over 15 runs – however, standard error was not plotted as it it significantly small (< 0.05 at all data points).

Figure 6 and confirm the intuitive hypothesis that computational time scales linearly with the number of initial conditions/well-behaved worlds.

Experiment (f). This experiment aims to study how PEC-RUNTIME behaves in a more realistic domain where every sensor has a fixed probability p of producing a reading at every instant. We consider the following domain descriptions:

$$F_1 \textbf{ takes-values } \{\top, \bot\}$$
$$F_2 \textbf{ takes-values } \{\top, \bot\}$$
$$F_3 \textbf{ takes-values } \{\top, \bot\}$$
$$F_4 \textbf{ takes-values } \{\top, \bot\}$$
$$F_5 \textbf{ takes-values } \{\top, \bot\}$$
$$\textbf{initially-one-of } \{(\{\neg F_1, \neg F_2, \neg F_3, \neg F_4, \neg F_5\}, 1)\}$$
$$\neg A_1 \wedge \neg A_2 \wedge \neg A_3 \wedge \neg A_4 \wedge A_5 \textbf{ causes-one-of } \{(\{\neg F_1, \neg F_2, \neg F_3, \neg F_4, F_5\}, 4/5), (\emptyset, 1/5)\}$$
$$\vdots$$
$$A_1 \wedge A_2 \wedge A_3 \wedge A_4 \wedge \neg A_5 \textbf{ causes-one-of } \{(\{F_1, F_2, F_3, F_4, \neg F_5\}, 4/5), (\emptyset, 1/5)\}$$
$$A_1 \wedge A_2 \wedge A_3 \wedge A_4 \wedge A_5 \textbf{ causes-one-of } \{(\{F_1, F_2, F_3, F_4, F_5\}, 4/5), (\emptyset, 1/5)\}$$

and, for every instant I and action A, the proposition

$$A_i \textbf{ performed-at } I \textbf{ with-prob } 0.5$$

has a some fixed probability p of being included in the domain description, where p varies in the set $\{0.1, 0.2, \ldots, 1\}$. Results are shown in Figure 7.

https://doi.org/10.1017/S1471068422000382 Published online by Cambridge University Press

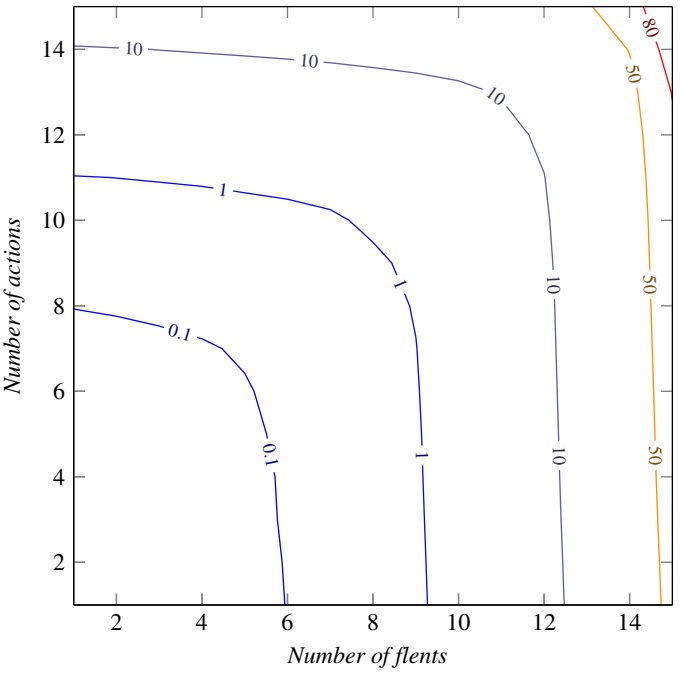


Fig. 5. Contour plot showing time (in seconds) to query the domain in Section 5.1, Experiment (d), expressed as a function of the number of actions and fluents.

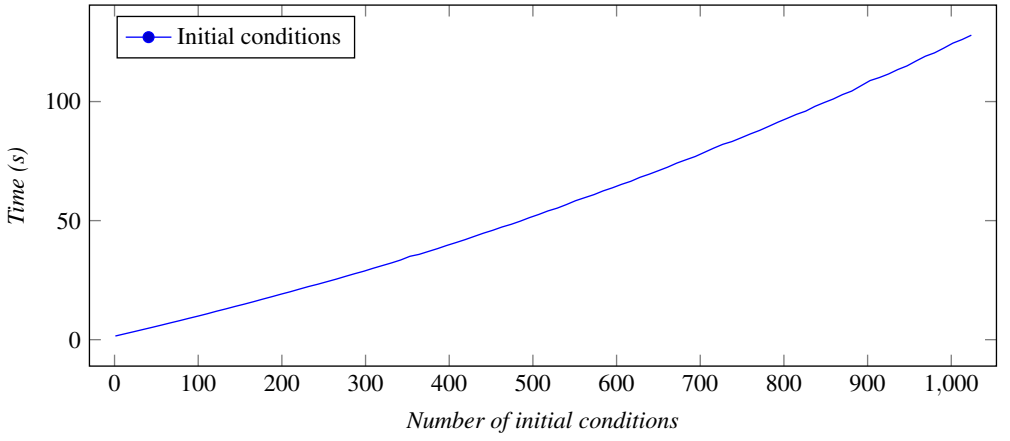


Fig. 6. Time (in seconds) to query the domain in Section 5.1, Experiment (e), expressed as a function of the number of initial conditions.

5.2 Effect of noise

One of the characteristics of EPEC (and its dialects) that makes it highly suitable as an interface with machine learning algorithms is that it supports events annotated with probabilities. If a classifier provides a confidence score for its output, it is possible to feed such score into EPEC in the form of a probability. This may be useful to account for

https://doi.org/10.1017/S1471068422000382 Published online by Cambridge University Press

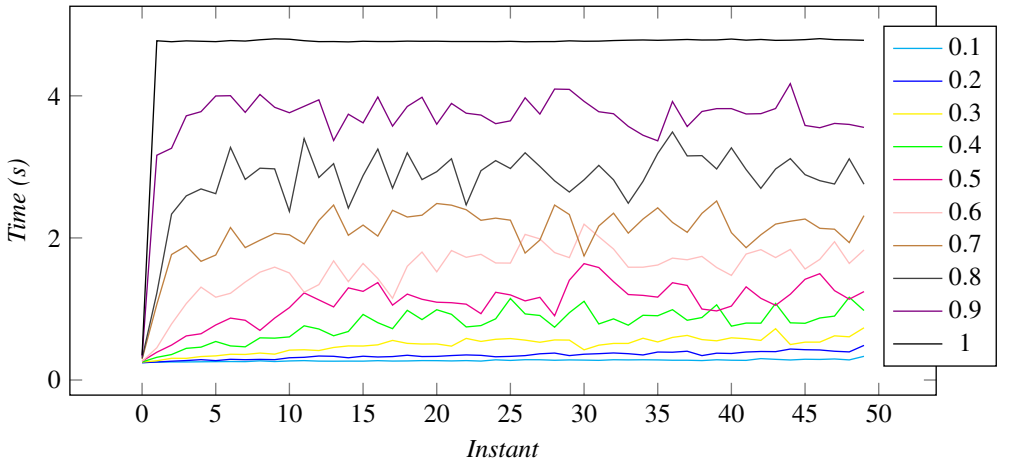


Fig. 7. Time (in seconds) to query the example discussed in Section 5.1, Experiment (f). The results show averages over 30 runs.

possible artifacts and flickering of a given classifier. For instance, consider a classifier C that alternatively produces readings *true* and *false* for some characteristic F. This may be represented in EPEC as the following stream of events:

$$\begin{gathered}
Ctrue \textbf{ occurs-at } 0 \textbf{ with-prob } P_{true} \\
Cfalse \textbf{ occurs-at } 1 \textbf{ with-prob } P_{false} \\
Ctrue \textbf{ occurs-at } 2 \textbf{ with-prob } P_{true} \\
Cfalse \textbf{ occurs-at } 3 \textbf{ with-prob } P_{false} \\
\vdots \\
Ctrue \textbf{ occurs-at } 18 \textbf{ with-prob } P_{true} \\
Cfalse \textbf{ occurs-at } 19 \textbf{ with-prob } P_{false}
\end{gathered}$$

where *Ctrue* (resp. *Cfalse*) is an event that causes the value of fluent F to be $\top$ (resp. $\bot$), that is:

$$\begin{gathered}
Ctrue \textbf{ causes-one-of } \{(\{F\}, 1)\} \\
Cfalse \textbf{ causes-one-of } \{(\{\neg F\}, 1)\}
\end{gathered}$$

and the truth value of f is initially unknown, that is:

$$\textbf{initially-one-of } \{(\{F\}, 0.5), (\{\neg F\}, 0.5)\}$$

We analyze the behavior of EPEC in different scenarios (corresponding to different values for P_{true} and P_{false}):

Scenario (a). The classifier C produces *true* with high confidence and *false* with low confidence, for example, $P_{true} = 0.99$ and $P_{false} = 0.01$. This domain description entails "$[F]$@20 **holds-with-prob** P" where $P \approx 0.9899$, which makes intuitive sense as the *false* events have low confidence and are discarded as background noise.

https://doi.org/10.1017/S1471068422000382 Published online by Cambridge University Press

Scenario (b). The classifier C produces *true* and *false* with a high degree of uncertainty. However, it assigns *true* a slightly higher degree of confidence, for example $P_{true} = 0.501$ and $P_{false} = 0.499$. This domain description entails "$[F]$@20 **holds-with-prob** P" where $P \approx 0.653246$, which reflects that such a sequence of events does not carry as much information as in Scenario (a), due to the the small difference between P_{true} and P_{false}.

Scenario (c). The classifier C produces both *true* and *false* with low confidence, with $P_{true} \gg P_{false}$. For example, let $P_{true} = 0.49$ and $P_{false} = 0.01$. This domain description entails "$[F]$@20 **holds-with-prob** P" where $P \approx 0.979286$, which again makes intuitive sense as *false* readings are again discarded as background noise, with the repeated *true* readings making the probability of F steadily grow throughout the timeline.

These scenarios show that EPEC is sensibly more accurate than any logical framework that is not capable of handling probabilities in the presence of noisy sensors. In fact, these frameworks would have to set a threshold, and only accept events with probabilities higher than that threshold. In our example, setting a threshold for example of 0.5 would lead to significant errors especially in Scenarios (b) and (c). In Scenario (b), all *Cfalse* events would be discarded, implying that F is detected to hold true at instant 20 with certainty (compare this to the probability 0.653246 assigned to the same query by EPEC). In Scenario (c), all events would be discarded and it would not be possible neither that F holds at 20 nor that it does not (compare again to the probability 0.979286 assigned by EPEC).

5.3 An example from AVATEA

As discussed in Section 4, a layer of classifiers directly injects events into PEC-RUNTIME, which merges them together and decides what action to take according to a predefined conditional plan. We show how PEC-RUNTIME accounts for these classifiers in the following example.

We use the Head Pose Recognition module to decide whether the child is looking at the screen or not, and therefore this module can trigger an *EyesNotFollowingTarget* event (and possibly an associated probability). Similarly, the Shoulders Alignment Module triggers a *BadPosture* event if it detects that the child's shoulders are not correctly aligned. Finally, the Emotion Recognition Module triggers a *LowValence* event when it thinks the child is experiencing negative valence. On the basis of these events, we aim to evaluate the probabilities of *TaskCorrect*, that is, that the child is performing the therapeutic task correctly, and *Engagement*, that is, that the child and engaged to the task. Consider the following narrative of events:

EyesNotFollowingTarget **occurs-at** 0 **with-prob** 1

BadPosture **occurs-at** 0 **with-prob** 1

LowValence **occurs-at** 0 **with-prob** 1

EyesNotFollowingTarget **occurs-at** 1 **with-prob** 1

BadPosture **occurs-at** 1 **with-prob** 1

LowValence **occurs-at** 1 **with-prob** 1

EyesNotFollowingTarget **occurs-at** 2 **with-prob** 76/100

LowValence **occurs-at** 2 **with-prob** 87/100

https://doi.org/10.1017/S1471068422000382 Published online by Cambridge University Press

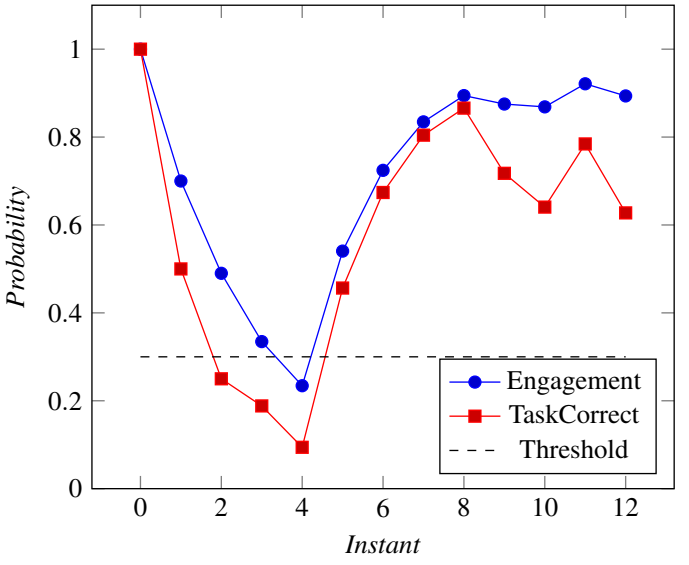


Fig. 8. *Engagement* and *TaskCorrect* as a function of time in the example from Section 5.3.

$$\textit{EyesNotFollowingTarget} \textbf{ occurs-at } 3 \textbf{ with-prob } 1$$
$$\textit{BadPosture} \textbf{ occurs-at } 3 \textbf{ with-prob } 1$$
$$\textit{LowValence} \textbf{ occurs-at } 3 \textbf{ with-prob } 1$$
$$\textit{EyesNotFollowingTarget} \textbf{ occurs-at } 7 \textbf{ with-prob } 7/100$$
$$\textit{EyesNotFollowingTarget} \textbf{ occurs-at } 8 \textbf{ with-prob } 89/100$$
$$\textit{EyesNotFollowingTarget} \textbf{ occurs-at } 9 \textbf{ with-prob } 74/100$$
$$\textit{EyesNotFollowingTarget} \textbf{ occurs-at } 11 \textbf{ with-prob } 1$$

These events are given a meaning by means of a domain description (see Appendix A for the full domain) that specify what bearing each event has on fluents *TaskCorrect* and *Engagement*.

Furthermore, consider a conditional plan of the following form:

$$\forall I,\ \textit{PlaySound} \textbf{ performed-at } I \textbf{ if-believes } (\textit{Attention}, [0, 0.3])$$
$$\forall I,\ \textit{LowerDifficultyLevel} \textbf{ performed-at } I \textbf{ if-believes } (\textit{TaskCorrect}, [0, 0.3])$$

which states that a sound must be played by the system whenever the *Attention* of the child is low, and that the difficulty level must be lowered if the child does not manage to perform it correctly.

PEC-RUNTIME works out the probabilities in Figure 8, and triggers decisions when they fall below the given threshold. In particular, *PlaySound* is triggered at instant 4, and *LowerDifficultyLevel* is triggered at instants 2, 3 and 4. It is worth noting here that thresholds may induce the so-called *rubberband effect*, where abrupt probability changes (due e.g., to noise) may lead to decisions being taken too frequently. Unfortunately, EPEC (and therefore also PEC-RUNTIME) does not allow to control this effect, therefore we

https://doi.org/10.1017/S1471068422000382 Published online by Cambridge University Press

implemented a bespoke semaphore in the underlying Python script that only allows decision to be taken every 20 s.

These decisions are forwarded to the game. At the end of each therapeutic session, this narrative is saved to a file so that it can be used to provide feedback to the therapists and re-consulted if needed. In this scenario, PEC-RUNTIME provides feedback such as "The child performed the task correctly for a total of 10 instants out of 13 (76.9%). The game difficulty was lowered at instants 2, 3 and 4 as s/he was unable to perform the exercise. His attention was lowest at instant 4.". A graph similar to that in Figure 8 is also shown to the therapists, so that they can have a quick overview of the session.

6 Related work

To our knowledge, our architecture is one of the first that merges *gamification* strategies for rehabilitation and *probabilistic logic programming* techniques. In the following, we briefly survey these two fields, and motivate our design choices.

6.1 Gamification

Gamification strategies consist in using game-like elements (e.g., points, rewards, performance graphs, etc.) in serious contexts such as ours. These have proven to be extremely successful to engage young children in diagnostic and therapeutic exercises, even before the advent of digital gaming. While games to test cognitive capabilities (Belpaeme *et al.* 2012) do not form a sharply defined class, games designed to test and improve motor skills are usually referred to as *exergames*. The effects of exergames have been found to be generally positive (Vernadakis *et al.* 2015). Gamification systems are usually effective in engaging young users in playful activities, while adapting the current challenge according to level of user competence. Sessions are typically logged in order to provide detailed feedback to therapists. On the cognitive side, these adaptive systems have been designed to evaluate subjective well-being (Wu *et al.* 2019) and phonological acquisition (Origlia *et al.* 2017) among others. Adaptive exergames have been used for example in the context of children with spinal impairments (Mulcahey *et al.* 2008), and to test gross motor skills (Huang *et al.* 2018).

(Deep) Machine Learning techniques are not advantageous in the case of exergames if they are used alone, as they must be trained on big amounts of data even when expert knowledge can be (relatively) easily extracted from experts and encoded in symbolic form (e.g., using logic programming), with the further advantage that symbolic knowledge can be used to automatically produce explanations and reports. As a centralized reasoning system, we use probabilistic logic programming techniques instead. These systems are a convenient option as they allow both for the representation of formal constraints needed to implement a clinically effective exercise, and for the statistical modeling of intrinsically noisy data sources.

6.2 Probabilistic reasoning about actions

Recently, logic-based techniques have been successfully applied to several fields of Artificial Intelligence, including among others event recognition from security cameras

https://doi.org/10.1017/S1471068422000382 Published online by Cambridge University Press

(Skarlatidis *et al.* 2015a,b), robot location estimation (Belle and Levesque 2018), understanding of tenses (Van Lambalgen and Hamm 2008), natural language processing (Nadkarni *et al.* 2011), probabilistic diagnosis (Lee and Wang 2018) and intention recogntion (Acciaro *et al.* 2021). Given the importance of Machine Learning and Probability Theory in AI, these frameworks and languages have gradually started employing probabilistic semantics (Sato 1995) to incorporate and deal with uncertainty. This has given birth to the field of *Probabilistic Logic Programming* (Riguzzi 2018), and we are particularly concerned with Probabilistic Reasoning about Action frameworks as they can deal with agents interacting with some (partially known) environment. For example Bacchus *et al.* (1999), Belle and Levesque (2018), that are based on the *Situation Calculus* ontology (Reiter 2001), can model imperfect sensors and effectors. The Situation Calculus' branching structure makes these frameworks mostly suitable for *planning* under partial states of information. A recent extension of language $\mathscr{C}+$ (Giunchiglia *et al.* 2004), dubbed $p\mathscr{BC}+$, is presented in Lee and Wang (2018). The authors show how $p\mathscr{BC}+$ can be implemented in LP^{MLN}, a probabilistic extension of ASP that can be readily implemented using tools such as *LPMLN2ASP* (Lee *et al.* 2017). Unlike our work, which focuses on runtime reasoning, $p\mathscr{BC}+$ is mostly suited for probabilistic diagnosis and parameter learning. The two languages *MLN-EC* (Skarlatidis *et al.* 2015b) and *ProbEC* (Skarlatidis *et al.* 2015a) extend the semantics of the *Event Calculus* (Kowalski and Sergot 1986; Miller and Shanahan 2002) ontology, using Markov Logic Networks (Richardson and Domingos 2006) and ProbLog (De Raedt *et al.* 2007) to perform event recognition from security cameras. In their proposed case study, the logical part of the architecture receives time-stamped events as inputs and processes them in order to detect complex long-term activities (e.g., infer that two people are fighting from the fact that they have been close to each other and moving abruptly during the last few seconds). Given their semi-probabilistic nature, these frameworks are able to handle uncertainty in the input events (ProbEC) or in the causal rules linking events and fluents (MLN-EC) but they do not deal with any form of epistemic knowledge. On the other hand, EPEC (D'Asaro *et al.* 2020) has the advantage of being based on the Event Calculus (making it particularly well suited for Event Recognition tasks) and being able to deal with epistemic aspects. Its semantics abstracts from any specific programming language and therefore it lends itself to task-specific optimizations, such as the runtime adaptation presented in this work.

7 Conclusion and future work

To summarize, the contribution of this paper is two-fold:

- It presents an architecture, currently being actively developed and tested, for the rehabilitation of children with DCDs. The suggested approach mixes machine learning, logic-based and gamification techniques.
- It introduces an novel implementation of a state-of-the-art probabilistic logic programming framework (EPEC) that can work at runtime.

The logical framework plays the role of an interface between the exergame and the machine-learning techniques, and is mainly used for two reasons: (i) guide the exergame according to a predefined strategy that is hard-encoded in the domain description, and

https://doi.org/10.1017/S1471068422000382 Published online by Cambridge University Press

(ii) provide human-understandable feedback after each therapeutic session, that can be used to log user performance over time.

Nonetheless, there is much room for extending the scope and functionalities of our application. As we collect data about the users of our system, we can compare the decisions taken by the system to those that therapists and psychologists would take, and adapt our domain descriptions accordingly. In the future, we would like to do this semi-automatically by *learning* the probabilities in our domain descriptions from data-streams annotated by experts.

It is also possible to expand the feedback functionalities of our system. At the moment, feedback includes a series of graphs about the level of engagement of the patient and limited textual reports about his/her performance level (see Section 5.3). However, a full history of what happened throughout each therapeutic session is recorded in an EPEC-readable narrative. Implementing a device to automatically translate these narratives into natural language would greatly enhance the transparency of our system, and constitutes a future step of this work. In fact, "explanations" coming from our system are limited to plots such as the one depicted in Figure 8, that shows what the engagement level of the child is throughout the therapeutic session, and reasons behind decisions are explained in terms of trespassed thresholds. Transforming them into natural language would greatly enhance intelligibility of our system.

Conflicts of interest

The authors declare they were partially supported by MIUR within the POR Campania FESR 2014-2020 AVATEA "Advanced Virtual Adaptive Technologies e-hEAlth" research project.

Supplementary material

To view supplementary material for this article, please visit http://doi.org/10.1017/S1471068422000382.

References

Acciaro, G. D., D'Asaro, F. A. and Rossi, S. 2021. Predicting humans: A sensor-based architecture for real time intent recognition using problog. In *Proceedings of the 22nd Workshop "From Objects to Agents", Bologna, Italy, September 1–3, 2021*, R. Calegari, G. Ciatto, E. Denti, A. Omicini and G. Sartor, Eds. CEUR Workshop Proceedings, vol. 2963. CEUR-WS.org, 72–82.

Bacchus, F., Halpern, J. Y. and Levesque, H. J. 1999. Reasoning about noisy sensors and effectors in the situation calculus. *Artificial Intelligence 111,* 1–2, 171–208.

Belle, V. and Levesque, H. J. 2018. Reasoning about discrete and continuous noisy sensors and effectors in dynamical systems. *Artificial Intelligence 262*, 189–221.

Belpaeme, T., Baxter, P. E., Read, R., Wood, R., Cuayáhuitl, H., Kiefer, B., Racioppa, S., Kruijff-Korbayová, I., Athanasopoulos, G., Enescu, V., Looije, R., Neerincx, M., Demiris, Y., Ros-Espinoza, R., Beck, A., Cañamero, L., Hiolle, A., Lewis, M., Baroni, I., Nalin, M., Cosi, P., Paci, G., Tesser, F., Sommavilla, G. and Humbert, R. 2012. Multimodal child-robot interaction: Building social bonds. *Journal of Human-Robot Interaction 1,* 2, 33–53.

D'Asaro, F. A., Bikakis, A., Dickens, L. and Miller, R. 2017. Foundations for a probabilistic event calculus. In *Proceedings of the 14th International Conference Logic Programming and Nonmonotonic Reasoning, LPNMR 2017*, M. Balduccini and T. Janhunen, Eds. Springer International Publishing, Cham, 57–63.

D'Asaro, F. A., Bikakis, A., Dickens, L. and Miller, R. 2020. Probabilistic reasoning about epistemic action narratives. *Artificial Intelligence 287*, 103352.

D'Asaro, F. A., Origlia, A. and Rossi, S. 2019. Towards a logic-based approach for multimodal fusion and decision making during motor rehabilitation sessions. In *Proceedings of the 20th Workshop "From Objects to Agents" (WOA)*.

De Raedt, L., Kimmig, A. and Toivonen, H. 2007. Problog: A probabilistic prolog and its application in link discovery. In *IJCAI*, vol. 7, 2462–2467.

Gebser, M., Kaminski, R., Kaufmann, B. and Schaub, T. 2014. Clingo = ASP + control: Preliminary report. In *Proceedings of Technical Communications of the 30th International Conference on Logic Programming (ICLP'14)*, M. Leuschel and T. Schrijvers, Eds. Theory and Practice of Logic Programming, Online Supplement, vol. 14(4–5). URL: http://arxiv.org/abs/1405.3694v1.

Giunchiglia, E., Lee, J., Lifschitz, V., McCain, N. and Turner, H. 2004. Nonmonotonic causal theories. *Artificial Intelligence 153,* 1-2, 49–104.

Huang, C.-Y., Tung, L.-C., Chou, Y.-T., Wu, H.-M., Chen, K.-L. and Hsieh, C.-L. 2018. Development of a computerized adaptive test of children's gross motor skills. *Archives of Physical Medicine and Rehabilitation 99,* 3, 512–520.

Kowalski, R. and Sergot, M. 1986. A logic-based calculus of events. *New Generation Computing 4,* 1, 67–95.

Krizhevsky, A., Sutskever, I. and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*. 1097–1105.

Lee, J., Talsania, S. and Wang, Y. 2017. Computing lpmln using asp and mln solvers. *Theory and Practice of Logic Programming 17,* 5–6, 942–960.

Lee, J. and Wang, Y. 2018. A probabilistic extension of action language bc+. *Theory and Practice of Logic Programming 18,* 3–4, 607–622.

Lin, F. and Reiter, R. 1997. How to progress a database. *Artificial Intelligence 92,* 1, 131–167.

Malek, S. and Rossi, S. 2021. Head pose estimation using facial-landmarks classification for children rehabilitation games. *Pattern Recognition Letters 152*, 406–412.

Miller, R. and Shanahan, M. 2002. Some alternative formulations of the event calculus. In *Computational Logic: Logic Programming And Beyond*. Springer, 452–490.

Mollahosseini, A., Hasani, B. and Mahoor, M. H. 2017. Affectnet: A database for facial expression, valence, and arousal computing in the wild. *IEEE Transactions on Affective Computing 10,* 1, 18–31.

Mulcahey, M., Haley, S. M., Duffy, T., Ni, P. and Betz, R. R. 2008. Measuring physical functioning in children with spinal impairments with computerized adaptive testing. *Journal of Pediatric Orthopedics 28,* 3, 330.

Nadkarni, P. M., Ohno-Machado, L. and Chapman, W. W. 2011. Natural language processing: an introduction. *Journal of the American Medical Informatics Association 18,* 5, 544–551.

Origlia, A., Cosi, P., Rodà, A. and Zmarich, C. 2017. A dialogue-based software architecture for gamified discrimination tests. In *GHITALY@ CHItaly*.

Papandreou, G., Zhu, T., Chen, L.-C., Gidaris, S., Tompson, J. and Murphy, K. 2018. Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 269–286.

https://doi.org/10.1017/S1471068422000382 Published online by Cambridge University Press

Reiter, R. 2001. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press.

Rescigno, M., Spezialetti, M. and Rossi, S. 2020. Personalized models for facial emotion recognition through transfer learning. *Multimedia Tools and Applications 79,* 47, 35811–35828.

Richardson, M. and Domingos, P. 2006. Markov logic networks. *Machine Learning 62,* 1, 107–136.

Riguzzi, F. 2018. *Foundations of Probabilistic Logic Programming*. River Publishers Series in Software Engineering. River Publishers.

Sato, T. 1995. A statistical learning method for logic programs with distribution semantics. In *Proceedings of the 12th International Conference on Logic Programming (ICLP95)*. Citeseer.

Skarlatidis, A., Artikis, A., Filippou, J. and Paliourasz G. 2015a. A probabilistic logic programming event calculus. *Theory and Practice of Logic Programming 15*, 213–245.

Skarlatidis, A., Paliouras, G., Artikis, A. and Vouros, G. A. 2015b. Probabilistic event calculus for event recognition. *ACM Transactions on Computational Logic (TOCL) 16,* 2, 11.

Van Lambalgen, M. and Hamm, F. 2008. *The Proper Treatment of Events*, vol. 6. John Wiley & Sons.

Vernadakis, N., Papastergiou, M., Zetou, E. and Antoniou, P. 2015. The impact of an exergame-based intervention on children's fundamental motor skills. *Computers & Education 83*, 90–102.

Wu, Y., Cai, Y. and Tu, D. 2019. A computerized adaptive testing advancing the measurement of subjective well-being. *Journal of Pacific Rim Psychology 13*, 1–10.