Modeling and Reasoning with ProbLog: An Application in Recognizing Complex Activities

Timo Sztyler
University of Mannheim
Mannheim, Germany
timo@informatik.uni-mannheim.de

Gabriele Civitarese
University of Milano
Milano, Italy
gabriele.civitarese@unimi.it

Heiner Stuckenschmidt
University of Mannheim
Mannheim, Germany
heiner@informatik.uni-mannheim.de

***Abstract*—Smart-home activity recognition is an enabling tool for a wide range of ambient assisted living applications. The recognition of ADLs usually relies on supervised learning or knowledge-based reasoning techniques. In order to overcome the well-known limitations of those two approaches and, at the same time, to combine their strengths to improve the recognition rate, many researchers investigated Markov Logic Networks (MLNs). However, MLNs require a non-trivial effort by experts to properly model probabilities in terms of weights. In this paper, we propose a novel method based on ProbLog. ProbLog is a probabilistic extension of Prolog, which allows to explicitly define probabilistic facts and rules. With respect to MLN, the inference mode of ProbLog is based on the closed-world assumption and it has faster response times. We propose a simple and flexible ProbLog model, which we exploit to recognize complex ADLs in an online fashion. Considering a dataset with 21 subjects, our results show that our method reaches high F-measure (83%). Moreover, we also show that the response time of ProbLog is satisfying for real-time applications.**

I. Introduction

In the last years, the rapidly aging society required a rapid surge in assisted living technologies to reduce health care costs [1]. The miniaturization and the availability of cheap sensing infrastructures allow to build the so called smart-homes: homes instrumented with sensors and actuators to help people in aging at home in safety and independence [2]. One of the most important application of smart-homes is the recognition of the Activities of Daily Living (ADLs) performed by the inhabitants. Indeed, continuously and unobtrusively monitoring the execution of ADLs is crucial for several applications, like the early diagnosis of cognitive diseases [3] or the detection of emergency situations [4].

Activity recognition techniques are divided into two categories: *data-driven* and *knowledge-based*. The former is based on supervised learning while the latter exploits logic formalisms (e.g., ontologies) to formally represent sensor events and activities. In order to combine the strength points of both approaches, several studies investigated Markov Logic Networks (MLN) [5]–[8]. This is a probabilistic logic that enables uncertain inference by generalizing first-order logic. However, even if MLN is a powerful tool that has a strong expressiveness in respect of logic formalisms, there are drawbacks considering the model's representation, interpretation, and intuition [9]. In this work, we focus on *ProbLog* [10], a probabilistic version of Prolog. *ProbLog* and MLN are closely related and there are several works that investigate the connection between them [9]. However, especially for activity recognition, there are considerable differences. First, *ProbLog* considers probabilities while MLN relies on weights which make it less suitable as a knowledge representation tool [9]. Indeed, even if the translation of probabilities into weights is feasible, it requires a non-trivial effort by an expert. In contrast, in *ProbLog* the probability values (e.g., manually designed or mined from a dataset) can be directly used without any transformation. Further, *ProbLog* is faster concerning answering queries as it only instantiates required groundings which make it more suitable in the light of larger knowledge bases. More precisely, the inference mode in *ProbLog* usually computes an interval on the probability of queries, while MLN is typically based on the MPE principle approximating a most likely state given some evidence [9]. In this context, *ProbLog* is defined by terms of groundings of probabilistic facts while MLN is defined through sets of weighted ground instances of the formulas. Finally, *ProbLog* relies on a closed-world assumption while MLN uses an open-world assumption. To the best of our knowledge there exists almost no work in respect of using *ProbLog* for activity recognition. We identified only a single work [11] that focused on rewriting event calculus with *ProbLog*.

In this paper, we study how to use *ProbLog* to efficiently recognize ADLs in almost real-time. For the sake of simplicity, we focus on a supervised approach which consists in mining a dataset to derive prior probabilities between sensor events and ADLs. Indeed, those probabilities could be also computed considering unsupervised approaches, e.g., semantic reasoning [8] or web-based mining [12], [13]. However, we focus on a supervised approach to evaluate the impact of *ProbLog* in recognizing activities. For that purpose, we performed experiments with a well-known real-world dataset, named CASAS [14], covering eight ADLs which were performed by 21 individuals.

The main contributions of our work are the following:

- We present a *ProbLog*-based method to efficiently recognize ADLs in an online fashion with an F-measure of 83%.
- We show the potential advantages of *ProbLog*'s marginal inference by achieving an F-measure of 93% considering the top-2 ranked activities.

- We clarify the benefits of *ProbLog* and introduce a guidance on how to practically write *ProbLog* programs for activity recognition.

The paper is structured as follows: In Section II, we introduce the theoretical part concerning *ProbLog*. Then, in Section III, we present our method in terms of how to create a *ProbLog* program and subsequently the implementation of our online activity recognition approach. Section IV shows our experimental results. Finally, Section V covers the conclusion and future work of this paper.

II. ProbLog

As the name already suggests, *ProbLog* [10] is an extension of Prolog [15]. More precisely, Prolog is a declarative logical programming language that expresses concepts in terms of facts and rules while *ProbLog* enables to associate these facts and rules with mutually independent probabilities, i.e., $P(A \cap B) = P(A)P(B)$. Thus, certain dependencies or correlations between triggered sensor events or ADLs have to be explicitly modeled.

In general, a *ProbLog* program (a.k.a. model) specifies a probability distribution over all possible worlds according to the distribution semantic [16]. In this context, a world ω can be considered as a possible and valid instance of our program. As a program consists of a set of defined clauses c_i that are labeled with probabilities p_i, we define a program by $T = \{p_1 : c_1, \ldots, p_n : c_n\}$ while the probability of a world ω, i.e., a certain instance of the program T, is defined as follows:

$$P(\omega|T) = \prod_{c_i \in \omega} p_i \prod_{c_i \in \omega_T \setminus \omega} (1 - p_i) \tag{1}$$

where $\omega_T \setminus \omega$ describes the set of clauses that were not instanced in w but are part of T, i.e., the set of false ground probabilistic atoms.

In *ProbLog*, we have to execute a query to answer a certain question, e.g., how likely a certain instance of a clause is. Unlike in Prolog, where a query can only succeed or fail, in *ProbLog* we have to compute the probability that it succeeds. In this context, the success probability $P(q|T)$ of a query q in a program T is defined as follows:

$$P(q|\omega) = \begin{cases} 1, & \exists \theta : \omega \models q\theta \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

$$P(q, \omega|T) = P(q|\omega) \cdot P(\omega|T) \tag{3}$$

$$P(q|T) = \sum_{\omega \subseteq W} P(q, \omega|T) \tag{4}$$

where W is the set of all possible worlds in respect of T. Hence, the probability that q succeeds is the sum of the probabilities of those worlds where q can succeed. In other words, the success probability of query q corresponds to the probability that the query q has a proof, given the distribution over logic programs.

III. Method

In the following, we introduce a comprehensive example how we use *ProbLog* for recognizing ADLs. Then, we present our approach, i.e., how we perform online activity recognition.

A. ProbLog for Activity Recognition

As our smart-home is equipped with sensors that detect interactions with items and furniture but also presence in certain locations, our sensor network produces a continuous stream of sensor events. In this context, a sensor event is characterized by a unique identifier, a sensor type, and a timestamp. As a *ProbLog* program consists - as Prolog - of a set of definite clauses, every clause c_i is labeled with a probability p_i. Thus, we have to define our sensor events as part of our program:

*Program Snippet 1 (**Event Clauses**):* Suppose that our sensor network produced two sensor events e_1 and e_2, then we have to define the following clauses with probability 1.0, i.e., as ground truth in our program.

$1.0{::}event(e_1, water, 5).$
$1.0{::}event(e_2, absent, 6).$

This implies that these two clauses have to be part of each possible world. Please note that strings that start with a lowercase character (e.g. water) are not considered as variables but values. The operator "::" enables to define probabilistic predicates.

As these sensor events were initially triggered by a resident, we aim to determine the activities that were performed at that time. In this context, we distinguish between an activity class and an activity instance where the former is considered as an abstract concept of an activity (e.g., *clean*) while the latter describes the actual occurrence of such an activity class during a certain time period. Consequently, we also have to define clauses that represent these facts. The following statement is called an annotated disjunction. It expresses that at most one of these choices is true. There is always an implicit null choice which states that none of the options is taken. In this example, however, that extra state has zero probability because the probabilities of the other states sum to one:

*Program Snippet 2 (**Instance Clauses**):* We assume that two activity instances ai_1 and ai_2 took place in our smart-home. Further, we consider two activity classes: ac_1 (*clean*) and ac_2 (*water plants)*. Initially, we define that the probability is equally distributed concerning that a certain activity class is assigned to an activity instance.

$0.5{::}inst(ai_1, ac_1, 0, 7); 0.5{::}inst(ai_1, ac_2, 0, 7).$
$0.5{::}inst(ai_2, ac_1, 4, 10); 0.5{::}inst(ai_2, ac_2, 4, 10).$

Note that these clauses do not imply that these two instances have to reflect different activities. The operator ";" represents a logical *XOR*.

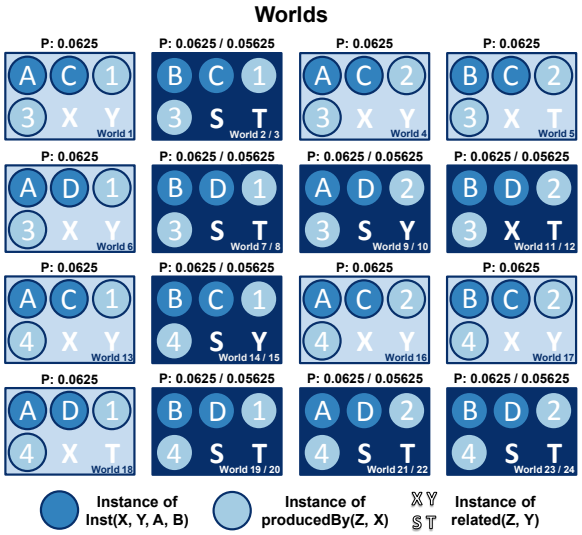

Fig. 1. It depicts all possible worlds that will be constructed considering Program Snippets 1-5. For reasons of clarity, we excluded the random variable *bond(X, Y)*. The boxes which represent two worlds (darker background) depict if the rule of Snippet 5 is true or false. For the remaining worlds this rule is always false. The probability of a world is computed by multiplying the probability of the individual instances of the random variables. If someone wants to know how likely it is that sensor event e_1 (Z) is related to *waterplants* (Y), we have to sum up the probabilities of each world which covers *related*(e_1,*waterplants)* and subsequently normalize this value. In this figure, this instance is derived from the combinations *1B* and *2D* and by implication has a probability of 0.909. The related *ProbLog* program can be found here[1].

Subsequently, we also have to associate the recorded sensor events and the defined activity instances. Analogous to the preceding program snippet, we also define annotated disjunction clauses to ensure that each sensor event is assigned to exactly one activity instance. In this context, to handle possible sensor network errors, i.e., recorded sensor events which actually did not occur and which should not be assigned to any activity instance, can be handled by assigning probability values which do not sum up to one.

*Program Snippet 3 (**Domain Constraints**):* Suppose that two activities were executed: ai_1 and ai_2. Then the following clauses ensure that each event is assigned to exactly one activity instance.

0.5::$producedBy(e_1, ai_1)$; 0.5::$producedBy(e_1, ai_2)$.
0.5::$producedBy(e_2, ai_1)$; 0.5::$producedBy(e_2, ai_2)$.

This implies that each event was produced either by *clean* (ai_1) or *water plants* (ai_2) with equal probability.

For evaluation, we express dependencies of random variables on the preceding introduced random variables. This includes the relation between a sensor event and an activity class but also between an activity instance and an activity class. For that purpose, the following rules consider the concept of *parentness* where the right part of the clause are parents and enable to derive the child. Thus, if the parents exist also the child has to exist but the child can exist without the parents.

[1]https://sensors.informatik.uni-mannheim.de/#results2018modelling

*Program Snippet 4 (**Hidden Predicates**):* Assume that "*inst*(ai_1,ac_1,0,7)" and "*producedBy*(e_1,ai_1)" are part of our world. Then the following rules derive that "*related*(e_1,ac_1)" and subsequently also "*bond*(ai_1, ac_1)" are part of that world.

$related(Z,Y)$:- $inst(X,Y,A,B)$, $producedBy(Z,X)$.
$bond(X,Y)$:- $related(Z,Y)$, $inst(X,Y,A,B)$.

The uppercase letters are variable arguments. Further, ":-" can be considered as implication where the left part is the head.

Indeed, considering all introduced program snippets as a united *ProbLog* program enables to query the random variables *related(Z, Y)* and *bond(X, Y)*, i.e., to compute the probabilities of all possible instantiations to determine the most probable assignment. Obviously, executing this program would not be helpful as everything has the same probability. For that reason, we enhance our *ProbLog* program with knowledge-based and temporal constraints.

*Program Snippet 5 (**Probabilistic Facts**):* The following rule states that it is most likely that the water sensor was used in context of *water plants*. This kind of rule allows to incorporate, for instance, previously computed prior probabilities.

0.9::$related(X,waterplants)$:-$event(X,water,T)$.

Note that assigning a probability of 1.0 to this rule implies that the corresponding activity instance has to be assigned to the class *water plants*.

*Program Snippet 6 (**Temporal Constraints**):* Intuitively, temporally close sensor events more likely belong to the same activity instance instead of distant ones. For that purpose, we define *closeAfter*(T_1,T_2) as a numerical constraint that allows to compute the distance in time of two events and compares it to a threshold.

$closeAfter(T_1,T_2)$:-$T_1 > T_2, T_3$ is T_1 - $T_2, T_3 < 2$.
0.6::$producedBy(X_2,I)$:-$event(X_1,Y_1,T_1)$, $event(X_2, Y_2, T_2)$,$closeAfter(T_2,T_1)$,$producedBy(X_1,I)$.

The "," operator represents a logical *AND* where "is" signifies an assignment.

*Program Snippet 7 (**Knowledge-based Facts**):* Assume that we know that the *can* is only used for *waterplants*. When *water* is taken shortly afterwards, then it is very likely that this event also belongs to *waterplants*.

0.9::$bond(Y,waterplants)$:-$event(X_1,water,T_1)$,$event(X_2,can,T_2)$,$closeAfter(T_1,T_2)$,$producedBy(X_1,Y)$.

In general, it should be avoided to assign a probability of 1.0 to prevent contradictions.

Please note that Program Snippets 5-7 are limited in respect of the number of rules and that our final program which we consider for experiments is more comprehensive[1]. Besides, including such rules has to be done with caution as they can also introduce side-effects. For instance, if we consider the Program Snippets 4 and 5, it points out that there is no restriction that avoids to instantiate the predicates $related(e_1, clean)$ and $related(e_1, waterplants)$ as the first one was derived by Program Snippet 4 and the second by Program Snippet 5. This violates our assumption that a sensor event always has a one-to-one relationship to an activity class. Hence, in this case, we have to explicitly restrict the assignment of an activity class to a sensor event.

*Program Snippet 8 (**Domain Constraints**):* Presuming that we only have two activity classes, the following rule ensures that within a world either *clean* or *waterplants* is associated with sensor event e_1:

$r1$:- $related(e_1, ac_1)$, \+$related(e_1, ac_2)$;
$related(e_1, ac_2)$, \+$related(e_1, ac_1)$.

$evidence(r1, true)$.

In this context, "\+" reflects a *NOT* operator. The evidence function is necessary to avoid the construction of worlds where this rule is violated, i.e., false.

Finally, to compute the probabilities of all possible instances of our random variables, we have to query those. For that purpose, *ProbLog* allows to execute *marginal inference* by, e.g., just adding $query(related(\_,\_))$. to the program. In this context, *ProbLog* constructs all necessary worlds to compute the probability of each instance of *related(Z, Y)*. Based on our introduced program snippets, Figure 1 illustrates these worlds and also explains the composition and probabilities.

B. Implementation

The essential idea of our approach is to segment in almost real-time the continuous stream of sensor events into windows. For each window, we build a *ProbLog* program to compute the probability distribution over the considered activities. In the following, we go into detail and explain this process.

B.I. Extracting prior probabilities

The training phase of our recognition method consists in deriving for each event type et_i (e.g., *"water"* meaning that the water faucet was touched) the prior probability distribution over the set of considered activities $\mathbf{A}$:

$$P(A=ac_k|E=et_i)=\frac{P(A=ac_k,E=et_i)}{P(E=et_i)}=\frac{freq(ac_k,et_i)}{\sum_{ac_j\in\mathbf{A}} freq(ac_j,et_i)}$$

where $P(A=ac_k|E=et_i)$ represents the likelihood that et_i is related to the activity class ac_k, while $freq(ac_k, et_i)$ is the number of times that ac_k triggers et_i in the dataset.

The resulting probabilities are incorporated as probabilistic facts in our *ProbLog* program.

B.II. Segmentation

We consider a segmentation strategy based on sliding windowing. In particular, we generate overlapping windows where each window comprises n and overlaps by o consecutive sensor events. The parameters n and o need to be chosen empirically. Even if more sophisticated segmentation techniques have been proposed in the literature [17]–[19], we choose a simple approach to better evaluate the impact of *ProbLog* in recognizing activities. Each window is independently analyzed and classified as soon as it is finalized.

B.III. ProbLog Program Generation

For each window w, we generate a *ProbLog* program (see Section III-A). In this context, the program covers for each event $e_i \in w$ a corresponding clause (see Snippet 1). Further, for simplification, we make the assumption that each window represents or is part of a single activity instance ai_i. Thus, the corresponding program of a window covers always only one instance clause in respect of all considered activity classes (see Snippet 2). Then, we add the required domain constraints (see Snippets 3 and 8). Finally, considering the computed probabilities $P(A=ac_k|E=et_i)$, we add for each event $e_i \in w$ and each considered activity class $ac_k \in \mathbf{A}$ rules which reflect these probabilities, i.e., that a certain event type occurs in respect of a certain activity (see Snippets 5).

B.IV. Online Activity Recognition

Once a window w was finalized and the corresponding *ProbLog* program generated, we immediately execute this program. More precisely, we execute the query "$query(bond(ai_1,\_))$.", which allows us to compute the probability distribution over the set of activities $\mathbf{A}$. We consider the most likely activity as the predicted activity for w. In a post-processing step, consecutive windows that were labeled equally can be concatenated. Indeed, a possible extension could be to combine the ranked activities with the classification results on the previous windows to refine and smooth the prediction (e.g., considering common-sense reasoning on the temporal sequence of activities). We consider that as future work.

IV. Results

In the following, we present our experimental setup and results. For that purpose, we use the well-known dataset of Cook et al. [14], [20], named *CASAS*. This dataset covers interleaved ADLs of twenty-one subjects acquired in a smart-home environment. A sensor network collected data about movement, temperature, use of water, interaction with objects, doors, and a phone. In this context, eight different activities were observed, i.e., *answer phone* (ac_1), *choose outfit* (ac_2), *clean* (ac_3), *fill medication dispenser* (ac_4), *prepare birthday card* (ac_5), *prepare soup* (ac_6), *watch DVD* (ac_7), and *water plants* (ac_8). The order and expenditure of time were up to the subject and it was allowed to perform the activities in parallel. During the data collection only one single person was present in the smart-home. To provide the possibility to reconstruct

TABLE I
RESULTS SHOW THE RECOGNITION RATE FOR THE INDIVIDUAL ACTIVITIES FOR THE PROPOSED APPROACH.

Activity	Precision	Recall	FP Rate	F-measure
ac_1	0.61	0.61	0.01	0.61
ac_2	0.81	0.85	0.01	0.83
ac_3	0.65	0.70	0.05	0.67
ac_4	0.90	0.66	0.01	0.76
ac_5	0.90	0.96	0.04	0.92
ac_6	0.79	0.93	0.05	0.85
ac_7	0.89	0.88	0.03	0.89
ac_8	0.90	0.57	0.01	0.69
$avg.$	0.83	0.83	0.03	0.83

TABLE II
PERFORMANCE ($\mathbf{F_1}$) OF OUR APPROACH IN RESPECT OF WHETHER THE CORRECT CLASS IS AMONG THE TOP-k RANKED RECOGNIZED CLASSES.

Activity	Top-1	Top-2	Top-3
ac_1	0.61	0.75	0.77
ac_2	0.83	0.93	0.96
ac_3	0.67	0.87	0.93
ac_4	0.76	0.90	0.91
ac_5	0.92	0.98	0.98
ac_6	0.85	0.94	0.96
ac_7	0.89	0.95	0.96
ac_8	0.69	0.90	0.98
$avg.$	0.83	0.93	0.95

our approach and experiments, we want to point to a web interface which is publicly available and allows to process *ProbLog* programs[1] as well as to our final *ProbLog* program[2].

For the experiments, we choose empirically $n = 6$ and $o = 50\%$ as our sliding window parameters. We compute the prior probabilities considering all available data, i.e., subject-independent. Table I shows the result and clarifies that all activities are well recognized. The activity *answer phone* (ac_1) has the lowest (61%) while *prepare birthday card* (ac_5) has the highest (92%) recognition rate. The reason for the worse performance of recognizing *answer phone* can be attributed to our simple segmentation technique. Indeed, we only have sensor events that indicate the start and stop of that activity but usually several windows between these two events are generated. Consequently, these interweaving windows are not classified as *answer phone* since we analyze each window separately. However, these interweaving windows cannot be simply classified as *answer phone* as we also have to consider situations where the phone is just touched or interleaved activities were performed.

Considering precision and recall metrics, it points out that especially *fill medication dispenser* (ac_4) and *water plants* (ac_8) have very unbalanced values (i.e., the precision is significantly higher than recall). These activities are not characterized by a single sensor type, but they consist in several variations of sensor event patterns. While most of these patterns are clearly recognizable, there are also those which are quite similar to the ones of other activities. In this context, Figure 2 clarifies that these two activities are mainly confused with *clean* (ac_3) and *prepare soup* (ac_6), i.e., activities which take place in the same location or require partly the same items. The confusion matrix also depicts that there are no conflicting classes that stand out and that the classification errors are minor and well distributed. Overall, the results show that *ProbLog* is quite suitable for the recognition of complex activities.

As we apply *marginal inference*, we have for each classified window the probabilities of all considered activity classes, i.e., it is possible to rank the individual activity classes from the most probable to the most unlikely. This enables to apply, e.g., rules in a post-processing step in respect of the preceding classified windows. In this context, Table II shows the performance of our proposed approach if we consider whether the correct activity class (ground truth) is among the top-k ranked classes. It strikes that in 93% of all cases the correct class in among the top-2 ranked classes (out of 8). This militates for the quality of the ranking and also for the reliability of our approach.

Finally, Figure 3 shows the empirical cumulative distribution function that describes the computational time needed by our method on each window. In 52% of the cases the computation time requires at most one second, while in 77% of the cases it requires at most 3 seconds. Unfortunately, there are some cases where the model requires more computational effort. Indeed, for more or less 6% of the windows our method takes more than 5 seconds (and up to 8) in order to derive the most likely activity. However, we believe that these cases are a minority and that in general our method achieves an acceptable response time considering the target applications, since our segmentation technique generates segments which median length is 25 seconds. All experiments were performed using only a single CPU core.

V. CONCLUSION AND FUTURE WORK

In this paper, we proposed a novel technique for the online recognition of complex ADLs in a smart-home environment

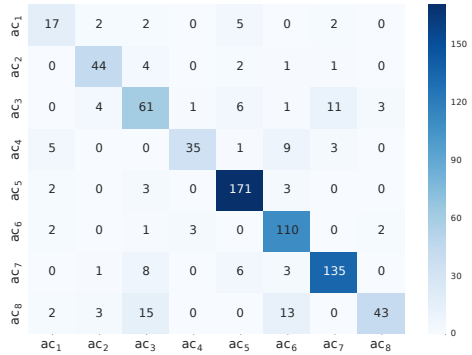

Fig. 2. The confusion matrix shows the classified windows. The x-axis represents the predicted activity class while the y-axis the actual one.

[1]https://dtai.cs.kuleuven.be/problog/editor.html
[2]https://sensors.informatik.uni-mannheim.de/#results2018modelling

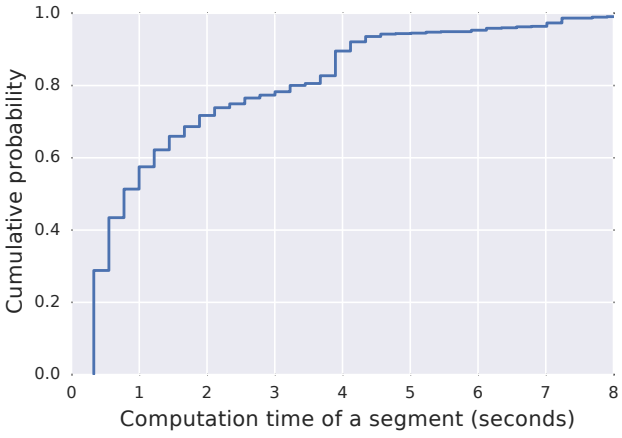

Fig. 3. Cumulative distribution function of the computation time for a single window.

based on *ProbLog*. Our method exploits simple hybrid probabilistic and knowledge-based rules to reconstruct, for a window of consecutive sensor events, the probability distribution over the considered activities. For that, we rely on prior probabilities between sensor events and activities, which are computed offline. The presented results show that our method already reaches satisfactory recognition results even with a simple segmentation strategy (83%). Moreover, considering the target applications, the response time is satisfying (median, one second per window) and we assume that using a smarter segmentation algorithm could lead to even better performance.

The benefits of using *ProbLog* for activity recognition are many. According to our previous experience with MLNs [8], the representation of knowledge and probabilities is easier and more straightforward, allowing to build programs more intuitively and to reduce the gap between the domain expert and the modeling framework. Moreover, since *ProbLog* considers the closed-world assumption, it is more scalable with respect to MLNs as less grounding operations are required. Even if scalability is not required for the approach considered in this paper, it is essential for other scenarios. For instance, different sensing infrastructures may require to include a higher number of sensor events for each window to accurately detect ADLs. Further, the number of constraints may grow with the time, since they can be learned from data or derived from common-sense knowledge. As future work, we want to fully exploit marginal inference in order to refine the classification of the windows. Further, we will also consider active learning techniques to personalize the recognition model on the monitored subject, by correcting prior probabilities and by learning *ProbLog* rules on-the-fly. Finally, we also aim to extend our ProbLog model in order to cope with parallel activities.

REFERENCES

[1] P. Rashidi and A. Mihailidis, "A survey on ambient-assisted living tools for older adults," *IEEE Journal of Biomedical and Health Informatics*, vol. 17, no. 3, pp. 579–590, 2013.

[2] M. Chan, D. Estève, C. Escriba, and E. Campo, "A review of smart homes - present state and future challenges," *Computer methods and programs in biomedicine*, vol. 91, no. 1, pp. 55–81, 2008.

[3] D. Riboni, C. Bettini, G. Civitarese, Z. H. Janjua, and R. Helaoui, "Smartfaber: Recognizing fine-grained abnormal behaviors for early detection of mild cognitive impairment," *Artificial Intelligence in Medicine*, vol. 67, no. Supplement C, pp. 57–74, 2016.

[4] T. Kleinberger, A. Jedlitschka, H. Storf, S. Steinbach-Nordmann, and S. Prueckner, "An approach to and evaluations of assisted living systems using ambient intelligence for emergency monitoring and prevention," in *Universal Access in Human-Computer Interaction. Intelligent and Ubiquitous Interaction Environments: 5th International Conference, UAHCI 2009, Held as Part of HCI International 2009, San Diego, CA, USA, July 19-24, 2009. Proceedings, Part II*. Springer Berlin Heidelberg, 2009, pp. 199–208.

[5] R. Helaoui, M. Niepert, and H. Stuckenschmidt, "Recognizing interleaved and concurrent activities: A statistical-relational approach," in *2011 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE Computer Society, 2011, pp. 1–9.

[6] P. Chahuara, A. Fleury, F. Portet, and M. Vacher, "Using markov logic network for on-line activity recognition from non-visual home automation sensors," *Ambient Intelligence: Third International Joint Conference, AmI 2012, Pisa, Italy, November 13-15, 2012. Proceedings*, pp. 177–192, 2012.

[7] D. Riboni, C. Bettini, G. Civitarese, Z. H. Janjua, and R. Helaoui, "Fine-grained recognition of abnormal behaviors for early detection of mild cognitive impairment," in *2015 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE Computer Society, 2015, pp. 149–154.

[8] D. Riboni, T. Sztyler, G. Civitarese, and H. Stuckenschmidt, "Unsupervised recognition of interleaved activities of daily living through ontological and probabilistic reasoning," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2016, pp. 1–12.

[9] M. Bruynooghe, T. Mantadelis, A. Kimmig, B. Gutmann, J. Vennekens, G. Janssens, and L. De Raedt, "Problog technology for inference in a probabilistic first order logic," in *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2010, pp. 719–724.

[10] L. De Raedt, A. Kimmig, and H. Toivonen, "Problog: A probabilistic prolog and its application in link discovery," in *Proceedings of the 20th International Joint Conference on Artifical Intelligence*. Morgan Kaufmann Publishers Inc., 2007, pp. 2468–2473.

[11] A. Skarlatidis, A. Artikis, J. Filippou, and G. Paliouras, "A probabilistic logic programming event calculus," *Theory and Practice of Logic Programming*, vol. 15, no. 2, pp. 213–245, 2015.

[12] P. Palmes, H. K. Pung, T. Gu, W. Xue, and S. Chen, "Object relevance weight pattern mining for activity recognition and segmentation," *Pervasive and Mobile Computing*, vol. 6, no. 1, pp. 43–57, 2010.

[13] D. Riboni and M. Murtas, "Web mining & computer vision: New partners for object-based activity recognition," in *2017 IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*. IEEE Computer Society, 2017, pp. 158–163.

[14] D. J. Cook, A. S. Crandall, B. L. Thomas, and N. C. Krishnan, "CASAS: A smart home in a box," *Computer*, vol. 46, no. 7, pp. 62–69, 2013.

[15] W. F. Clocksin and C. S. Mellish, *Programming in Prolog*, 5th ed. Springer Science & Business Media, 2003.

[16] T. Sato, "A statistical learning method for logic programs with distribution semantics," in *In Proceedings of the 12th International Conference On Logic Programming (ICLP95)*. MIT Press, 1995, pp. 715–729.

[17] T. Gu, S. Chen, X. Tao, and J. Lu, "An unsupervised approach to activity recognition and segmentation based on object-use fingerprints," *Data & Knowledge Engineering*, vol. 69, no. 6, pp. 533–544, 2010.

[18] G. Okeyo, L. Chen, H. Wang, and R. Sterritt, "Dynamic sensor data segmentation for real-time knowledge-driven activity recognition," *Pervasive and Mobile Computing*, vol. 10, no. Part B, pp. 155–172, 2014.

[19] J. Wan, M. J. O'Grady, and G. M. O'Hare, "Dynamic sensor event segmentation for real-time activity recognition in a smart home context," *Personal and Ubiquitous Computing*, vol. 19, no. 2, pp. 287–301, 2015.

[20] G. Singla, D. J. Cook, and M. Schmitter-Edgecombe, "Tracking activities in complex settings using smart environment technologies," *International Journal of Biosciences, Psychiatry, and Technology*, vol. 1, no. 1, pp. 25–35, 2009.