Real-Time Sensor Observation Segmentation For Complex Activity Recognition Within Smart Environments

Darpan Triboan[1], Liming Chen[1], Feng Chen[1], Sarah Fallmann[1], Ismini Psychoula[1]
Context, Intelligence and Interaction Research Group, De Montfort University, UK [1]
Email: {darpan.triboan@my365., liming.chen@, fengchen@, sarah.fallmann@, ismini,psychoula@}dmu.ac.uk

***Abstract*—**Activity Recognition (AR) is at the heart of any types of assistive living systems. One of the key challenges faced in AR is segmentation of the sensor events when inhabitant performs simple or composite activities of daily living (ADLs). In addition, each inhabitant may follow a particular ritual or a tradition in performing different ADLs and their patterns may change overtime. Many recent studies apply methods to segment and recognise generic ADLs performed in a composite manner. However, little has been explored in semantically distinguishing individual sensor events and directly passing it to the relevant ongoing/new atomic activities. This paper proposes to use the ontological model to capture generic knowledge of ADLs and methods which also takes inhabitant-specific preferences into considerations when segmenting sensor events. The system implementation was developed, deployed and evaluated against 84 use case scenarios. The result suggests that all sensor events were adequately segmented with 98% accuracy and the average classification time of 3971ms and 62183ms for single and composite ADL scenarios were recorded, respectively.

***Keywords*—** *Sensor Segmentation, Activities of Daily Living (ADL), Composite Activities, Ontology Modelling, Web Ontology Language (OWL), and Activity Recognition (AR).*

I. INTRODUCTION

The global aging population has already outnumbered people under 18 in many countries. For instance, the United Kingdom (UK) now has more people aged over 60 than under 18 and the number is estimated to be double by 2030 according to a recent report from the non-profit organisation Age UK[1]. Some of the main concerns with the rising aging population are that the degradation of quality of life and the care receiving/providing by health care services. The goal of Ambient Assistive Living (AAL) systems[2]–[4] is to provide assistive tools to target those concerns.

Human Activity Recognition (HAR) plays a critical role in AAL systems in order to provide timely assistance. The applications of efficient HAR is not only restricted for AAL systems but can also be applied to a raft of other domains such as security surveillance, smart cities, smart grids, and Ecommerce, just to name a few. Extensive efforts are being made to accurately perform HAR of an individual living in ubiquitous smart environments. With recent advancement in sensing technology, it is now possible to monitor nearly every part of our daily activities. However, performing activity recognition (AR) with large amounts of contextual data being generated from a smart environment in a short burst and the complex nature of human performing activities of daily living (ADL) that is always changing, still remains as an open challenge. The key contributions of this paper include: (i) proposing a segmentation approach by utilising generic and personalised ADL knowledge and recognise simple and composite ADLs in real-time; (ii) proposing a light-weight mechanism to allow inhabitants to specify personal preferences for conducting a given ADL; (iii) evaluating the system developed and present the findings.

The nature of how humans perform ADLs must be understood before diving deeper into the processes of performing AR. A person can perform single or multiple (composite) ADLs at a given time – see Fig. 1. ADLs (A_1 and A_2) can have a set of atomic actions performed in unordered manner ({abcdef} and {123456}). For instance, single `MakeTea` (A_1) activity can be performed with actions such as adding: `TeaBag`(a), `HotWater`(b) and then `Milk`(c) in any order. This single ADL can also be performed along with multiple other ADLs; either incrementally (i.e. A_1 then `MakeToast` (A_2)), concurrently (i.e. A_1 with A_2), and in parallel (A_1 and A_2 running simultaneously. Furthermore, each individual may follow specific tradition, ritual or culture to perform a given activity which cannot be generalised when describing ADL. In addition, even when two individuals share the same values, they may still have their unique preferences to perform the same activity which can also change over time. Therefore, when designing the ADL model, developing segmentation and AR algorithms, one must take aforementioned parameters into consideration.

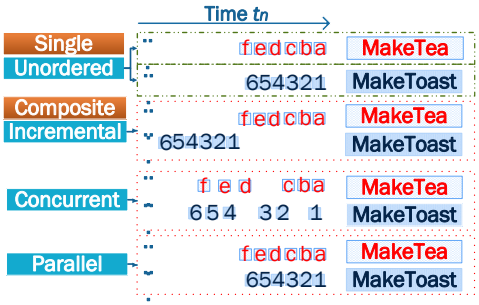

Fig. 1 - Nature of human to perform ADL

The process of AR can be described as having four main phases; activity modelling, classification, learning and sensing. Many studies have proposed a number of approaches and they can be classified as data-driven, knowledge-driven and hybrid approaches to model the ADL and to perform AR. In the data-

driven (DD) [5], [6] modelling approach, activities models are generated after processing pre-recorded datasets using generative or discriminative classification techniques. One of the key advantages of using the data-driven modelling approach is that it can discover unknown or unseen activities and handle uncertainty in events being performed by the inhabitants. However, one of the major shortfalls with the data-driven approach is the pre-process time required to compute the large dataset historically recorded, commonly known as the "cold start" problem. In contrast, the knowledge driven approach, is where domain experts in the field of interest conceptualise and describe factual elements of the being into a model that is interlinked, known as the ontological model. The knowledge-driven (KD) approach uses formal and logical theories to create a well-defined knowledge that is based on the ontological model that is human and machine friendly to interpret. Despite overcoming the "cold start" issue of the data-driven approach still falls short in handling unseen or uncertain data [2]. Although, the fuzzy logic and probabilistic based ontologies were proposed in the past, problems such as adequately expressing rules using fuzzy DL persist [2]. The common problem for both of these approaches is that it assumes complete description of all the entities and concepts within the activity model. Therefore, the hybrid approach [7], [8] is used to combine the expressivity power from KD and the ability to handle unseen or uncertainty in events from the DD approach to incrementally grow the initial model.

Activity classification and activity learning methods [8] are influenced by the modelling approach selected and the quality of the segmented data relevant to a given activity is also critical in order to achieve higher efficiency and accuracy when performing AR. Recent studies applied a number of segmentation and activity classification approaches, i.e., correlation between dynamic/fixed timing window[9], [10], statistical and probabilistic [11] approaches. More relevant segmentation and AR studies will be further analysed in section II. Activity learning is a process whereby discovery of new activities, patterns, and user preferences are dynamically learned, mainly the data-driven approach to evolve initial knowledge model.

In order to collect and monitor an inhabitant's actions and the changes to their environment, a wide variety of sensing technologies are available. The sensing methods can be categorised as vision and sensor based approaches. Whilst the vision based sensing approach has been successfully applied in areas such as security surveillance, the sensor-based approach has become more appealing in smart home (SH) environments due to lower ethical and privacy concerns. The sensor based sensing approach can be classified into ambient, dense and wearable sensing[12]. The ambient sensing+-g is performed to collect environmental data such as temperature, luminosity, motion, sound, and door/window opening. The dense sensing is used to monitor inhabitant's interactions with everyday objects, i.e. by embedding sensors into kettle, knife, television and fridges to retrieve information such as touch, and object movement/position and location. The wearable sensing can be further classified into outerwear and implantable[13]. The wearable sensors are generally used to monitor human body movement and physiological parameters such as heart rates, electrocardiogram (ECG), body postures/movements and the mind. Due to such a diversity in sensors and the type of contextual data being generated at different frequencies simultaneously, one inherent challenge is to separate the sensor events in relation to the ongoing activity queue in order to perform AR.

In the remainder of the paper, the existing studies related to segmentation and AR process that takes generic and/or personalised knowledge into consideration is reviewed in Section II and based on findings a new segmentation method is proposed in Section III. An overview of the system implementation is provided in Section IV and an evaluation and results will be in section V. The conclusion and future research direction is discussed in Section 0.

II. Related Work

In recent studies, the KD approach is commonly used but ranges in the methods that are employed. For instance, studies in [14]–[20] adopt ontological models to describe ADLs, environmental entities and their relations along with other methods to classify and infer unfolding activities. These methods include: description logics (DLs), temporal relationship of activities, static/dynamic timing window protocol, Semantic Web Rule Language (SWRL) based rules, SPARQL Protocol and RDF Query Language (SPARQL) queries and reasoning tools. Work in [20] presents an event filtering approach by adding preconditions with probabilities on the phases when carrying out each ADL in order to segment the incoming events. It is unclear how the algorithm can detect new activity when an action is shared or part of a precondition for one activity but is not for the new activity. For instance, `MozzarellaCheese` can be part of the precondition of `MakePizza` ADL and post condition for `MakeCheesyToast` ADL. This approach has achieved good accuracy in segmenting and recognising composite activities but there is scope of improvement in terms of recognising other scenarios.

Meanwhile, other studies in [14]–[20], do not directly inspect each sensor event as they arrive and then segment to the appropriate queue related to ongoing activities. Instead, the events are queried from the database and then rules/continuous queries are executed along with the ontology by the reasoner before inferring the activity and its execution complexity. In particular, work in [14], [15] used SWRL based inferencing rules to define the nature of activities with a temporal representation technique. These SWRL rules and Java Expert System Shell (JESS) rule engines were used to segment the sensor events using their timestamp information and perform entailments for the complexity of the ongoing activities. One of the major limitations of this approach is that the segmentation of sensor events is performed using a generic ontology reasoner and it is unclear if reclassification of the whole ontology is done incrementally or not. In case of the non-incremental reclassification approach, the performance and scalability can degrade exponentially as the size of an ontological model and data grow. Furthermore, rules can be generated for general purpose and also for inhabitant specific preferences as provided in the study in [21]. Therefore, one would have to reclassify the generic ontology with generic rules and inhabitant specific rules exclusively.

Similarly, work in [22] presents a layered ontology, namely OntoAALISABETH and complex event processing (CEP) engine based framework, namely, AALISABETH. The framework integrates temporal based reasoning with a dynamic time window sizing mechanism to segment the incoming data and perform AR in real-time. The approach leverages between the Esper solution for CEP and D2RQ engine to map data into RDF graphs. Although, the framework utilises highly optimised, scalable Esper CEP engine solution and is open source, the system falls short in directly segmenting the incoming sensor data semantically in real-time as it arrives from the sensor network. This limits the client applications to receive event-based notification which is critical in an emergency situation such as fall detection. Another key limitation of the framework is that the event data from the sensor network is stored directly into a traditional relational database management system (RDBMS) without inspecting individual events and segmenting them appropriately or appending to an ongoing activity queue. Instead, to filter or segment sensor events for a given ADL, continuous queries are required to be executed in order to be returned to as a set of sensor events and then perform Web Ontology Language (OWL) reasoning capabilities. Alternatively, the Pellet reasoner which has incremental reasoning support (i.e., only the effected changes in the ontology are classified) could be further utilised instead of creating an overhead to query and map each of the events from the RDBMS database using the D2RQ tool. Furthermore, the framework is not intended to cater for inhabitant's preferences when performing a generic ADL but it can be potentially extended.

III. The Proposed Approach

Generic knowledge to conduct a given ADL activity is represented as an ontological model (schema) and inhabitant's personal preferences as individuals (data). Both of this knowledge is used to segment each sensor event to relevant ongoing activity threads as they arrive. The overview of the approach is depicted in Fig. 2. The sensor events are initially added to the queue and multiple activity threads analyse the sensor events in the queue and store the relevant events independently. Therefore, one sensor event can be shared independently by two different activity threads with different ADL motives. For instance, opening `Fridge` action can be shared with thread T_1 which was inferred to be performing `MakeDrink` ADL and thread T_2 with `MakeMeal` ADL. Each activity thread initially performs generic knowledge (T-box) reasoning and then the inhabitant's preferences (A-box) are inspected to check for association of the sensor event and current ADL class motive of the thread. The twofold segmentation steps are depicted in Fig. 3 and further elaborated in part B. Moreover, each ongoing activity thread not only performs further AR with the current sensor information in the queue but also has timeout and completion procedures, i.e. storing relevant information and prompting the inhabitant when relevant. However, in the case where a sensor event or action is not associated with any ongoing activity threads or none created, a new activity thread is created by a session event manager running on a different thread. The key elements of the semantical approach is elaborated in next sections, however, details can be found in [23].

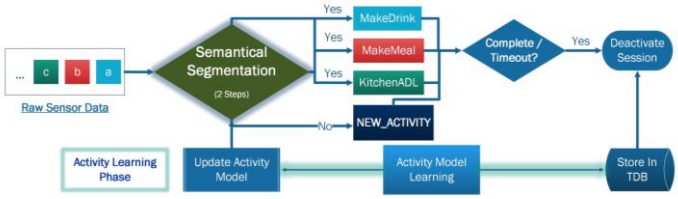

Fig. 2 - Overview of Semantical Segmentation Approach

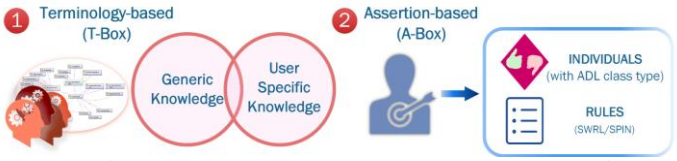

Fig. 3 - Twofold segmentation process: (1) generic (T-box) and (2) preference (A-box) reasoning

A. ADL Modelling

The ontological modelling technique is employed to capture (1) generic knowledge of being such as ADLs, environmental objects, sensing environment and inhabitant's profile; (2) inhabitant specific preferences to carry out a given ADL. Fig. 3 presents a snapshot of how generic `MakeTea` ADL knowledge to be linked to Inhabitant1's preference as an individual using `rdf:type` property.

1) Generic Knowledge For Segmentation (T-box)

The semantic web framework provides web ontology language OWL to express the complex knowledge into classes, relationships (object & data properties) and data (individuals) [23]. The common vocabularies can be shared across applications to create an ever growing web of knowledge. Some of the vocabularies are reused to create this ontological model for automatic reasoning to identify inexplicit knowledge. The main goal of the ontological model is to express what, where and how the actions are required in order to satisfy a given ADL. For instance, what kind of everyday objects inhabitants need to interact with for performing the `MakeTea` ADL, where should the activity be performed in the house and how should they execute the task. Fig. 4 partially describes the `MakeTea` ADL in the ontology editor tool named Protégé. The `MakeTea` ADL class inherit the properties described from super classes and uses `rdfs:subclassOf` object property to define actions or the context to carry out the activity. The actions for the `MakeTea` ADL are described using object properties and the classes of everyday objects; `hasAdding`, `hasContainer`, `hasHeatingAppliances`, `hasHotMealMaterial` and so on. These object properties can have characteristics and relationships between everyday objects classes and the ADLs. For instance, `hasHotDrinkType` object property has domain of `MakeHotDrink` ADL class and `HotDrinkType` material as range property. This means that any everyday object that is a subclass of `HotDrinkType` is part of the actions defined for `MakeHotDrink` ADL class or it's subclasses. These object properties are used to add further restrictions such as universal and existential quantification ($\forall$, $\exists$) using some and only, logical operations such as not, and, or ($\neg$, $\wedge$, $\vee$), and cardinality restrictions ($\leq$, $\geq$, $=$). Other common operators are also available and can be used to increase the expressivity of the ADL model in terms of class, relationships

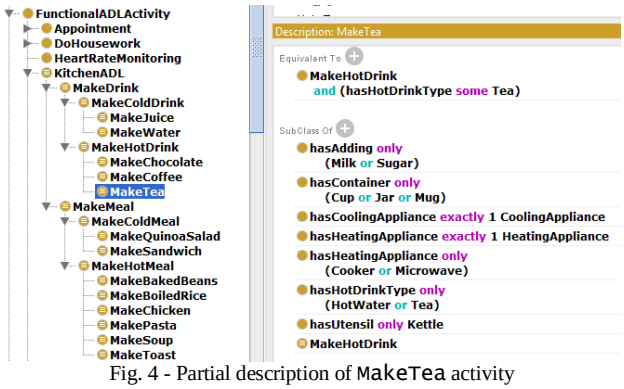

Fig. 4 - Partial description of `MakeTea` activity

and data. Similarly, the other 12 subclasses of `MakeDrink` and `MakeMeal` ADL classes are also described. With multiple relationships created as a data (individual), a reasoning engine can perform automatic inferencing to determine the type of the ADL class the actions in the individual belongs to.

One of the issues is to describe one or more supporting activities required to perform one activity. For instance, using a tea recipe book (`Reading`) to make a healthy tea (`MakeTea`). One method available is to use disjoint restriction. When applied to `MakeTea` and `Reading` ADL classes (both with different parent classes), the conflicts in the model will be raised by the reasoner to say that two ADL classes are independent. Alternatively, removing disjoint restrictions to recognise both ADLs in one set of actions specified in the individual as depicted in Fig. 5. Both of these methods have been used for the ADLs at different levels of the hierarchy. For instance, `MakeTea`, `MakeCoffee` and `MakeChocolate` activities are disjointed but `FunctionalADLActivity` and the sibling `RecreationalADLActivity` classes are not. Another issue faced when modelling the ADL is with premature assumptions of the completeness and accuracy ADL knowledge described

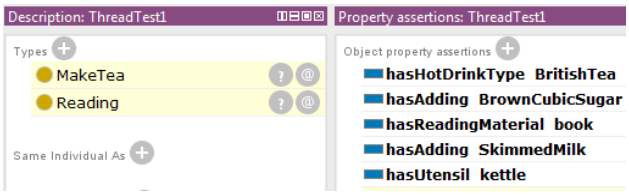

Fig. 5 - Linking main `MakeTea` activity with Reading a recipe book ADL.

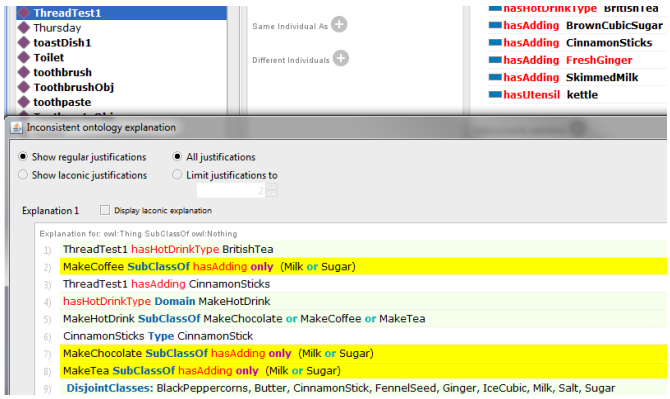

Fig. 6 - Inconsistency on `hasAdding` object property due the restriction applied to `MakeTea` ADL class.

by the domain experts, which ultimately leads to conflicts and an inconsistent model. The assumptions can occur due to two cases; human error or lack of domain knowledge and some actions that are subjective to individual inhabitant's preferences. In the first case perhaps, not all adding(ingredient) or type of containers for `MakeTea` ADL are defined and in second case `FreshGinger` and `CinnamonSticks` adding can be subjective to individual as highlighted by inconsistent ontology explanation window in Fig. 6.

2) User-specific Knowledge For Segmentation (A-Box)

To support incomplete and inhabitant specific preferences, individuals are created. These individuals are associated to the inhabitant and to a given ADL class which have a list of sensors that are attached to the objects and other attributes as shown in Fig. 7 and Fig. 8. Fig. 7 provides an example of how the generic and inhabitant specific knowledge can be created and linked using classes, individuals and rules. SWRL rules can also be added as a string in the individual, however this may require an independent rule engine to be executed. Alternatively, SPARQL Inferencing Notation (SPIN) rules can be used to perform queries on the data stored in the triplestore containing the inhabitant's preferences[24]. Fig. 8 presents multiple inhabitant preferences that are related to specific ADLs. The top section presents individually named, `Patient1_Preferences_IndianTea`, that has a type of preference for `MakeTea` ADL class along with a list of sensors using `hasSensor` object properties and data properties to describe other attributes such as preference name and creation timestamp. Similarly, other preferences can be created as shown in the middle and bottom of the figure to describe `MakeToast` and `MakeBakedBeans` preference. This method is lightweight and no inhabitant specific reasoner is required to be running. The SPIN rules or just a SPAQRL query language can be executed on the triplestore to retrieve multiple inhabitant's preferences for a given ADL class simultaneously. Therefore, this method is considered appropriate during the segmentation phase as the inhabitant's preferences can be scalable and has lower latency in terms of query time and there are no additional overheads for running multiple reasoners per inhabitant.

Another method is to layer the inhabitant specific and generic ADL ontology descriptions along with SWRL rules. This can be achieved by using the OWL API and Jena API to create and manipulate the model once generic and inhabitant specific models are combined and rules are loaded into the virtual memory. The reasoning can be performed using the Pellet reasoner and JESS rule engine after combining the generic and inhabitant specific ontology that is managed dynamically. However, the main limitation of this method is that the changes made to the inhabitant specific ontologies will need to be tracked along with the mechanism to resolve any

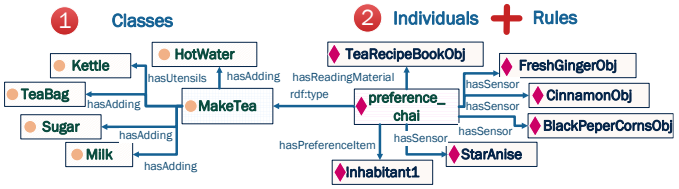

Fig. 7 - Example of generic (T-box) and user preference (A-box) specification

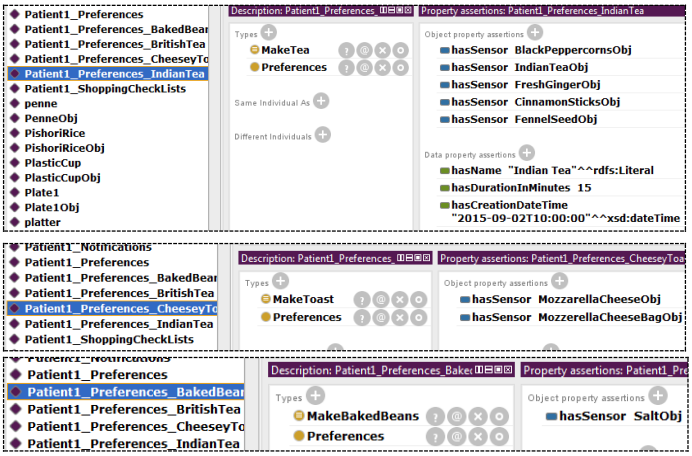

Fig. 8 - Inhabitant preferences as individuals with a list of sensors.

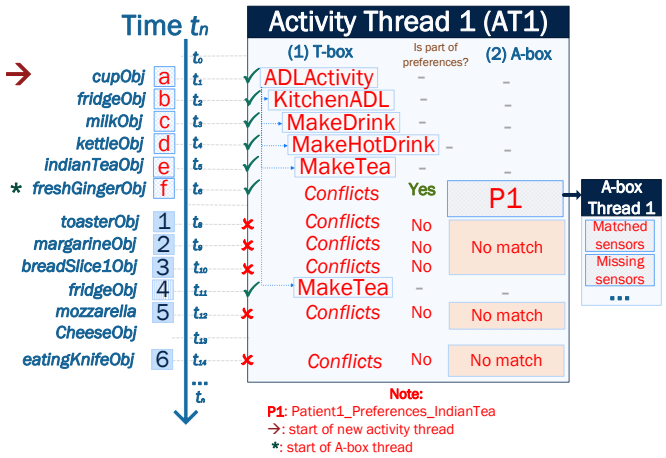

Fig. 9 - Scenario 1: concurrent `MakeTea` and `MakeToast` activity – initial thread created containing actions for `MakeTea`.

conflicts in the knowledge that may arise. Another key limiting factor is that inhabitant a specific reasoner will need to be created and maintained[25] at run-time. Hence, the amount of in-memory space and computation power required can grow exponentially. This can potentially create high latency in segmenting individual sensor events and undermine the scalability of the approach. Therefore, this method may not be suitable for real-time segmentation purposes however it can provide a higher accuracy when performing personalised AR.

B. Semantical Segmentation Process

The semantical segmentation process involves inspecting individual sensor events in a twofold process depicted in Fig. 3. In the first step, an ontological model is used that describes generic ADLs and the inhabitant's environment. Each sensor is linked to an everyday object in the household and the type of that object is assigned as part actions required for ADLs. Therefore, the association between everyday objects and ADLs can be automatically inferred using reasoners such as Pellet and HermiT or even by running SPARQL queries on the ADL model. This process is known as terminology box (T-box) reasoning. The second step is only executed when the result returned from T-box reasoning identifies any conflicts with the ADL class description in the model. The inhabitant's preferences are currently manually defined in advance and stored as individuals containing a list of objects that an inhabitant prefers to use to perform a given ADL. Therefore, SPARQL queries are used in extract preferences of the inhabitant for a given ADL in the second step. The process in the second step is known as assertion box (A-box) reasoning. A-box reasoning is performed on individuals which includes: instance checking, conjunctive query answering and consistency checking[26].

Fig. 9 and Fig. 10 illustrate how actions for `MakeTea` and `MakeToast` ADL performed concurrently are segmented by two activity threads. Both generic and preferred actions are triggered at a given time (t_n). Fig. 9 shows how the initial activity thread (AT1) is created when `cupObj` sensor is activated at t_1 and continuously stores events relevant to the ADL class in the thread. The object attached to `cupObj` sensor is queried from the triplestore, added to new individual and incremental T-box reasoning is executed. The T-box reasoning result indicated that the object is related to `ADLActivity` class with no conflicts with the model, hence the A-box reasoning is not required to be executed. Next, the sensor event at t_2 is received and AT1 `fridgeObj` performs T-box reasoning along with previous sensor(s). The new result, `KitchenADL` class, is checked against the current `ADLActivity` class if it is equivalent or subsuming class. In this case, `KitchenADL` class is a subclass of `ADLActivity`, therefore satisfying the subsuming conditions and storing `cupObj` and `fridgeObj` sensor events in the AT1. Similarly, `milkObj`, `kettleObj` and `indianTeaObj` sensor events are processed by AT1 where the ADL classes are incrementally classified and the sensor events are stored into the thread. The `freshGingerObj` sensor event however, is not described as adding in the generic `MakeTea` ADL description, therefore the reasoner returns with conflicts. The A-box reasoning is then performed to find any inhabitant's preferences related to `MakeTea` ADL containing items from the current list of sensors in thread with `freshGingerObj`. Multiple preferences could be returned however, in this case only one preference named, `Patient1_Preferences_IndianTea` is retuned as a result from SPARQL query. A single sub-thread (A-box Thread 1) is created with other missing sensors and other relevant information from the preference into the thread. This sub-thread then inspects the incoming sensor events and updates the missing and matched sensors list independently. AT1 thread and the sub-thread(s) for A-box reasoning can continue inspecting unfolding events until the completion criteria is satisfied i.e. having no child ADL class and missing sensors in A-box threads or a dynamic timeout mechanism for the ADL. The completion criteria for the ADL will be inspected in future work.

The next set of actions for `MakeToast` ADL received between t_8-t_{14} and inspected by AT1 but only one shared `fridgeObj` event is stored. The session event manager running in parallel inspects the sensor events in the queue and detects `toastObj` is not part of the `MakeTea` ADL class in AT1 and A-box thread 1. Therefore, another activity thread 2 (AT2) is created as depicted in Fig. 10. The same process described for AT1 is executed for AT2 to perform incremental T-box reasoning on a generic model and if the inhabitant's preferences are found A-box threads are created. As the other sensor events are received, the incremental T-box reasoner was able to capture events relevant to `MakeToast` ADL with one

conflicting `mozzarellaCheeseObj`. However, with A-box reasoning, it was identified that the `mozzarellaCheeseObj` is part of the inhabitant's preference named `Patient1_Preferences_CheeseyToast` to perform the `MakeToast` activity.

This approach not only takes inhabitant specified preferences and rules into consideration but also supports overcoming incompleteness of ADL, conflicting ADL descriptions in the ontology models and ability to dynamically evolve the generic knowledge and inhabitant preferences at segmentation level. Although, explicit ADL preference specified by the inhabitant is used to segment sensor events at run-time, the activity learning mechanism will be investigated in the future but is out of the scope of this paper. The approach leverages between the pellet reasoner to perform incremental reasoning on generic knowledge encoded into an ontological model (T-Box reasoning) and inhabitant specific preferences knowledge using SPARQL-based queries (A-Box reasoning).

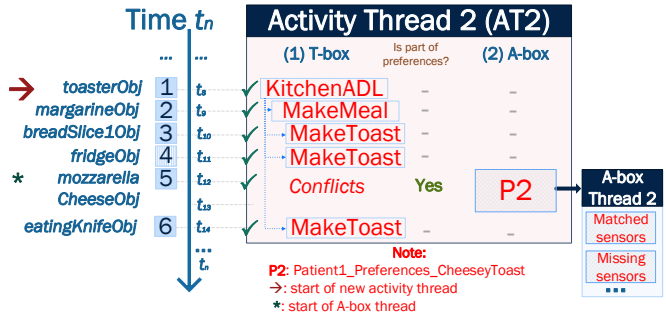

Fig. 10 – Scenario 1: identifying new `MakeToast` activity (AT2 & A-box 2 thread)

IV. IMPLEMENTATION

An android mobile application and RESTful web service has been used to create a service-oriented architecture (SOA) system. A SOA enables the web service to execute computation tasks such as segmentation and AR on the sensor events stream and storing the results into the Jena Fuseki triplestore using Jena API. The web service exposes these resources to multiple client devices running on independent operating systems using hypertext transfer protocol (HTTP) asynchronously. The web service receives all the sensor events from the sensing environment using wired/wireless connections methods and performs four main tasks; broadcast, store, segment sensor events and performs AR. The sensor events and the results from segmentation and AR are broadcasted independently using server-sent (SSE) protocol and stored in the triplestore. Multithreading concepts have been employed to segment each ADL into a thread described in Section III. A single ADL thread runs the T-Box reasoning and one more A-Box thread. The reasoning result and sensor events are broadcasted to the clients and the android application continuously capture and present information to the inhabitant. Details can be found in previous studies [27], [28]. Fig. 11 shows a snapshot of how concurrent activities are separated into separate threads and presented on the mobile.

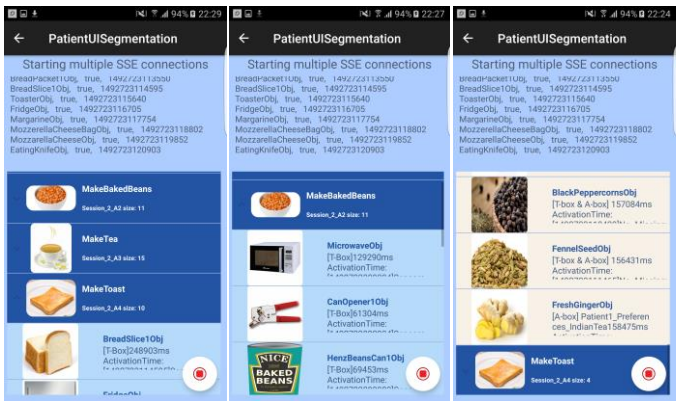

Fig. 11 – Segmentation results for three concurrent ADLs

A. *Reasoner and Supporting Tools Selection*

To perform T-box reasoning, an automatic reasoner is used. Reasoner is a software tool developed to derive new facts from the existing ontological model and a dataset. There are a number of reasoners developed over the years and most of them use first-order predicate logic [26] and perform forward and backward chaining. Some of the key requirements for selecting a reasoner are that it supports the incremental classification for only the part of ontology that was effected by the changes [29], full DL family support for higher expressivity, rules support, justification of conflicts, low latency in classification and support both T-Box and A-Box reasoning. Studies in [26], [30] describe a number of popular reasoners using large ontologies, compare against their key features and categorise according to their characteristics. The Pellet reasoner has been selected as it supports most requirements stated above along with being open source and supported by a number of application programming interfaces (APIs) and ontology editors such as Protégé and NeOn toolkit. OWL API are Jena API both support the Pellet reasoner to perform incremental reasoning and manipulate the ontology. Jena API further supports other reasoners to be implemented easily. Although, the pellet reasoner takes up higher heap space and has higher delay time than FaCT++ when performing concept satisfiability checking after classification but out performs in subsumption query[30].

V. EVALUATION

A. *Experiment design*

The actions for three ADLs were scripted in no particular order to perform with only generic actions and another with the inhabitant's preferences; namely, `MakeTea`, `MakeToast` and `MakeBakedBeans`. The relevant actions for the generic ADL and some inhabitant's preferences are described in TABLE I. These three ADLs are first tested individually and then combined to create composite activity scenario; incremental, concurrent and parallel; see TABLE II. A total of 30 activity scenarios were created for the experiment with 10s interval between sensor events. The accuracy and average classification time is then calculated for the segmentation algorithm to allocate each sensor event into a corresponding ADL thread and for the overall activity scenario. The Samsung S6 edge smartphone running 6.0.1 Android OS was used and the web service was deployed on the HP EliteBook Folio 1040 G2 with i7 2.60GHz processor and 8GB RAM. The sensor events are

TABLE I. SINGLE ACTIVITY SEQUENCES EXAMPLE

Activities	Related actions/ sensors attached to objects	#
MakeTea		
Generic actions (G)	KettleObj, Cup1Obj, TeaJarObj, IndianTeaObj, KitchenSinkTap1Obj, SugarJarObj, FridgeObj, Milk1Obj, Spoon2Obj	9
Personalised actions (P)	[FreshGingerObj], [CinnamonSticksObj], [BlackPeppercornsObj], [FennelSeedObj]	4
MakeBakedBeans		
Generic actions (G)	Spoon1Obj, HenzBeansCan1Obj, HenzBeansObj, CanOpener1Obj, MicrowaveBowl1Obj, MicrowaveObj, Plate1Obj, EatingKnifeObj	8
Personalised actions (P)	[SaltObj]	1
MakeToast		
Generic actions (G)	Plate1Obj, BreadPacket1Obj, BreadSlice1Obj, ToasterObj, FridgeObj, MargarineObj, EatingKnifeObj	7
Personalised actions (P)	[MozzerellaCheeseBagObj], [MozzarellaCheeseObj]	2
Note: [*SensorName*] - User preference item, # - number of sensors		

TABLE II. COMBINATIONS OF SIMPLE ACTIVITIES

Activity Comb.	ADL Sequences	Expected no. threads	Actions	
			Gen. (G)	Gen. & pref. (G+P)
AC1	MakeTea, MakeToast	2	16	22
AC2	MakeTea, MakeBakedBeans	2	17	22
AC3	MakeToast, MakeBakedBeans	2	15	18
AC4	MakeToast, MakeBakedBeans, MakeTea	3	24	31
AC5	MakeBakedBeans, MakeTea, MakeToast	3	24	31
AC6	MakeTea, MakeToast, MakeBakedBeans	3	24	31
Total		15	120	155

TABLE III. SINGLE ACTIVITY PERFORMED IN NO PARTICULAR ORDER WITH GENERIC AND PERSONAL PREFERENCES

Activity Type	Test cases	In relevant thread	Unexp. actions in threads *	Excess threads created		Avg. time (ms) +
MakeTea	G	9	0	0	0	2394.67
MakeToast	G	7	0	0	0	2468.57
MakeBaked Beans	G	8	0	0	0	2372.25
MakeTea	G+P	13	0	1	1	10828.85
MakeToast	G+P	9	0	0	0	3786.87
MakeBaked Beans	G+P	9	0	0	0	1972.44
Total	6	55	0	1	1	3970.61 (avg)
* excludes additional thread(s) actions, + including excess threads						

currently simulated due to limited number of sensors and time.

B. Results

The average segmentation time taken per sensor event for single activity is 3971ms in contrast to 62183ms milliseconds for composite ADL scenarios as shown in TABLE III. and TABLE IV. The result in TABLE III. is where all the sensor

TABLE IV. MULTIPLE ACTIVITIES PERFORMED IN COMPOSITE MANNER

Activity Comb.	Test cases	All actions in thread(s)?		Excess threads	Unexp. actions in threads *	Total Avg. time + (ms)
Inc.						
AC1	G	✓	16	1	1	36,330.64
AC2	G	✓	17	1	4	41543.17
AC3	G	✓	15	1	1	30354.98
AC4	G	×	15/24	3	3	95819.25
AC5	G	✓	24	1	5	60742.14
AC6	G	✓	24	1	6	72690.97
AC1	G+P	✓	22	1	1	54949.21
AC2	G+P	✓	22	0	5	21905.05
AC3	G+P	✓	18	0	1	12561.28
AC4	G+P	×	31	3	3	99807.19
AC5	G+P	×	30/ 31	1	4	62016.20
AC6	G+P	✓	31	1	3	87298.32
Con.						
AC1	G+P	✓	22	1	0	56752.83
AC2	G+P	✓	22	1	5	23993.51
AC3	G+P	✓	18	2	1	64074.61
AC4	G+P	✓	31	1	1	70289.79
AC5	G+P	✓	31	2	6	131784.92
AC6	G+P	✓	31	2	5	181894.97
Par.						
AC1	G+P	×	21/ 22	2	0	43055.55
AC2	G+P	✓	22	0	3	8309.10
AC3	G+P	×	16/ 18	1	0	35944.94
AC4	G+P	✓	31	1	4	63737.04
AC5	G+P	✓	31	1	5	77355.87
AC6	G+P	✓	31	1	4	59173.90
Total	24	572/585		29	71	62182.73 (avg.)
* excludes additional thread(s) actions, + including excess threads						

events for activity case scenario were adequately placed in the correct thread. Only the MakeTea activity case scenario created additional threads with more than double the average time when processing 9 generic and 4 preference items. On the other hand, TABLE IV. shows 20 out of 24 activities performed in a composite manner or 572 out of 585 sensor events were added to the relevant thread. The segmented activity thread captured a total of 71 additional unexpected sensor events in the segmented thread, i.e., multiple spoon objects or heating/cooling appliances. However, 29 additional threads were created and failed to classify any ongoing activity.

VI. CONCLUSION AND FUTURE WORK

The semantical segmentation approach is proposed which combines generic knowledge conceptualised as an ontological model and inhabitant specific preferences to conduct a specific ADL as asserted individuals. Upon sensor activation, the event is inspected by one or more active ADL threads. Each ADL thread first runs an automatic T-box reasoning to find relevant ADLs and if no result is returned, inhabitant specific preferences are queried from the triplestore relevant to the current ADL thread. When all active ADL threads fail to find any relevance for a given sensor event, a new ADL thread is

created. The approach leverages between the incremental Pellet reasoner, OWL & Jena API, and the notion of multithreading. The proposed method was implemented and tested against 84 test scenarios. The results indicate that the proposed segmentation approach takes on average of 3971ms and 62183ms for single and composite ADL scenarios, respectively. It can be seen that 98% of the sensor events were accurately segmented into relevant threads. A future study is proposed to optimise the thread management for ADLs to perform incremental reasoning asynchronised [31], add support for personalised specific rules[32], measure/reduce excess heap space and add time series analysis to detect start and completion of the activity. The study would then focus on AR and learning algorithms.

REFERENCES

[1] B. J. Banks, "The 'Age' of Opportunity," IEEE Pulse, vol. 8, no. 2, pp. 12–15, 2017.

[2] S. Zolfaghari, R. Zall, and M. R. Keyvanpour, "SOnAr: Smart Ontology Activity recognition framework to fulfill Semantic Web in smart homes Samaneh," in Proceedings of the Annual Hawaii International Conference on System Sciences, 2016, vol. 2016–March, pp. 3339–3348.

[3] C. Debes, A. Merentitis, S. Sukhanov, M. Niessen, N. Frangiadakis, and A. Bauer, "Monitoring Activities of Daily Living in Smart Homes," IEEE Signal Process. Mag., no. March, pp. 81–94, 2016.

[4] Z. Tang, J. Guo, S. Miao, S. Acharya, and J. H. Feng, "Ambient intelligence based context-aware assistive system to improve independence for people with autism spectrum disorder," Proc. Annu. Hawaii Int. Conf. Syst. Sci., vol. 2016–March, pp. 3339–3348, 2016.

[5] L. Chen, C. D. Nugent, and H. Wang, "A knowledge-driven approach to activity recognition in smart homes," IEEE Trans. Knowl. Data Eng., vol. 24, no. 6, pp. 961–974, 2012.

[6] L. Chen, C. Nugent, and G. Okeyo, "An ontology-based hybrid approach to activity modeling for smart homes," IEEE Trans. Human-Machine Syst., vol. 44, no. 1, pp. 92–105, 2014.

[7] L. Chen, C. Nugent, and J. Rafferty, "Ontology-based Activity Recognition Framework and Services," Proc. Int. Conf. Inf. Integr. Web-based Appl. Serv. - IIWAS '13, pp. 463–469, 2013.

[8] L. Chen, G. Okeyo, H. Wang, R. Sterritt, and C. Nugent, "A systematic approach to adaptive activity modeling and discovery in smart homes," Proc. - 2011 4th Int. Conf. Biomed. Eng. Informatics, BMEI 2011, vol. 4, pp. 2192–2196, 2011.

[9] G. Okeyo, L. Chen, H. Wang, and R. Sterritt, "Dynamic sensor data segmentation for real-time knowledge-driven activity recognition," Pervasive Mob. Comput., vol. 10, pp. 155–172, 2014.

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

[11] D. R. Faria, M. Vieira, C. Premebida, and U. Nunes, "Probabilistic Human Daily Activity Recognition towards Robot-assisted Living," pp. 582–587, 2015.

[12] C. Ye, Y. Xia, Y. Sun, S. Wang, H. Yan, and R. Mehmood, "ERAR: An Event-Driven Approach for Real-Time Activity Recognition," 2015 Int. Conf. Identification, Information, Knowl. Internet Things, pp. 288–293, 2015.

[13] M. Cornacchia and Y. Zheng, "A Survey on Activity Detection and Classification Using Wearable Sensors," IEEE Sens. J., vol. 17, no. 2, pp. 386–403, 2017.

[14] G. Okeyo, L. Chen, H. Wang, and R. Sterritt, "A hybrid ontological and temporal approach for composite activity modelling," Proc. 11th IEEE Int. Conf. Trust. Secur. Priv. Comput. Commun. Trust. - 11th IEEE Int. Conf. Ubiquitous Comput. Commun. IUCC-2012, pp. 1763–1770, 2012.

[15] G. Okeyo, L. Chen, and H. Wang, "Combining ontological and temporal formalisms for composite activity modelling and recognition in smart homes," Futur. Gener. Comput. Syst., vol. 39, pp. 29–43, 2014.

[16] J. Rafferty, C. D. Nugent, J. Liu, and L. Chen, "From Activity Recognition to Intention Recognition for Assisted Living Within Smart Homes," IEEE Trans. Human-Machine Syst., vol. PP, no. 99, pp. 1–12, 2016.

[17] G. Meditskos, S. Dasiopoulou, and I. Kompatsiaris, "MetaQ: A knowledge-driven framework for context-aware activity recognition combining SPARQL and OWL 2 activity patterns," Pervasive Mob. Comput., 2015.

[18] G. Okeyo, L. Chen, H. Wang, and R. Sterritt, "Dynamic sensor data segmentation for real time activity recognition," Pervasive Mob. Comput., vol. 10, Part B, pp. 155–172, 2014.

[19] R. Helaoui, D. Riboni, M. Niepert, C. Bettini, and H. Stuckenschmidt, "Towards activity recognition using probabilistic description logics," Act. Context Represent. Tech. Lang., vol. 12, p. 5, 2012.

[20] U. Naeem, "Activities of daily life recognition using process representation modelling to support intention analysis," Int. J. Pervasive Comput. Commun., vol. 11, no. 3, p. 347, 2015.

[21] K. L. Skillen, L. Chen, C. D. Nugent, M. P. Donnelly, W. Burns, and I. Solheim, "Ontological user modelling and semantic rule-based reasoning for personalisation of Help-On-Demand services in pervasive environments," Futur. Gener. Comput. Syst., vol. 34, pp. 97–109, 2014.

[22] R. Culmone, P. Giuliodori, and M. Quadrini, "Human Activity Recognition using a Semantic Ontology-Based Framework," Int. J. Adv. Intell. Syst., vol. 8, no. 2, pp. 159–168, 2015.

[23] D. Riboni and C. Bettini, "OWL 2 modeling and reasoning with complex human activities," Pervasive Mob. Comput., vol. 7, no. 3, pp. 379–395, 2011.

[24] [W3C, "SPIN - Overview and Motivation," 2011. [Online]. Available: https://www.w3.org/Submission/spin-overview/. [Accessed: 22-Aug-2016].

[25] R. Volz, S. Staab, and B. Motik, "Incremental Maintenance Of Materialized Ontologies," Lect. Notes Comput. Sci., vol. 2888/2003, no. 2888/2003, pp. 707–724, 2003.

[26] S. Abburu, "A Survey on Ontology Reasoners and Comparison," Int. J. Comput. Appl., vol. 57, no. 17, pp. 33–39, 2012.

[27] D. Triboan, L. Chen, F. Chen, and Z. Wang, "Towards a Service-Oriented Architecture for a Mobile Assistive System with Real-time Environmental Sensing," TSINGHUA Sci. Technol., vol. 21, no. 6, pp. 581–597, 2016.

[28] D. Triboan, L. Chen, and F. Chen, "Towards a Mobile Assistive System Using Service- oriented Architecture," in 2016 IEEE Symposium on Service-Oriented System Engineering Towards, 2016, pp. 187–196.

[29] B. Cuenca Grau, C. Halaschek-Wiener, and Y. Kazakov, "History matters: Incremental ontology reasoning using modules," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 4825 LNCS, pp. 183–196, 2007.

[30] K. Dentler, R. Cornet, A. Ten Teije, and N. De Keizer, "Comparison of reasoners for large ontologies in the OWL 2 EL profile," Semant. Web, vol. 2, no. 2, pp. 71–87, 2011.

[31] Y. Ren, J. Z. Pan, I. Guclu, and M. Kollingbaum, "A combined approach to incremental reasoning for EL ontologies," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 9898 LNCS, no. August, pp. 167–183, 2016.

[32] M. Peters, C. Brink, S. Sachweh, and A. Zündorf, "Scaling parallel rule-based reasoning," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 8465 LNCS, pp. 270–285, 2014.