A Semantics-based Approach to Sensor Data Segmentation in Real-time Activity Recognition

Darpan Triboan[a,*], Liming Chen[a], Feng Chen[a], Zumin Wang[b]

[a] *Context, Intelligence and Interaction Research Group, De Montfort University, UK*
[b] *Department of Information Engineering, Dalian University, China*

Abstract

Activity Recognition (AR) is key in context-aware assistive living systems. One challenge in AR is the segmentation of observed sensor events when interleaved or concurrent activities of daily living (ADLs) are performed. In the past, several studies have proposed methods of separating and organising sensor observations and recognise generic ADLs performed in a simple or composite manner. However, little has been explored in semantically distinguishing individual sensor events directly and passing it to the relevant ongoing/new atomic activities. This paper proposes Semiotic theory inspired ontological model, capturing generic knowledge and inhabitant-specific preferences for conducting ADLs to support the segmentation process. A multithreaded decision algorithm and system prototype were developed and evaluated against 30 use case scenarios where each event was simulated at 10sec interval on a machine with i7 2.60GHz CPU, 2 cores and 8GB RAM. The result suggests that all sensor events were adequately segmented with 100% accuracy for single ADL scenarios and minor improvement of 97.8% accuracy for composite ADL scenario. However, the performance has suffered to segment each event with the average classification time of 3971ms and 62183ms for single and composite ADL scenarios, respectively.

Keywords: Sensor Segmentation, User Preferences, Activities of Daily Living

*Corresponding author
Email addresses: darpan.triboan@my365.dmu.ac.uk (Darpan Triboan), liming.chen@dmu.ac.uk (Liming Chen), fengchen@dmu.ac.uk (Feng Chen), wangzumin@dlu.edu.cn (Zumin Wang)

(ADL), Composite Activities, Ontology Modelling, and Activity Recognition.

1. Introduction

Ambient Assistive Living (AAL) systems [1, 2, 3] are being developed as a tool to support increasing ageing population [4] to carry out their Activities of Daily Living (ADL) and enable health care services to achieve higher quality-of-care. Human Activity Recognition (HAR) is a key part of AAL systems to allow accurate and timely assistance to the inhabitant. The application of HAR approaches can also be applied in other domains such as security, surveillance, smart cities, and e-commerce. The process of activity recognition (AR) can be described in five phases; (i) data collection from ubiquitous smart environment, (ii) segmentation of diverse and vast amount of data; (iii) modelling ADLs and domain knowledge, (iv) classification of composite actions performed by a single/multiple inhabitants; and (v) activity learning for the changing habits.

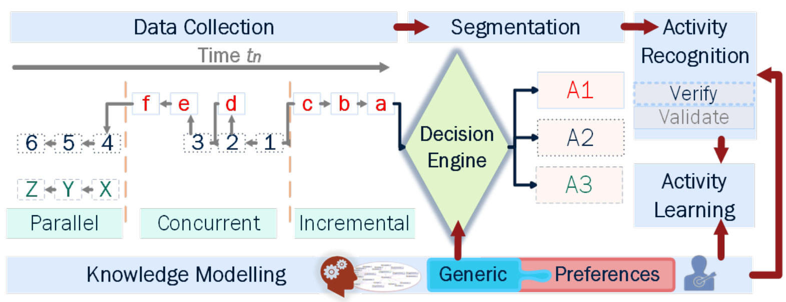

Figure 1: Five interdependent phases of AR: i) data collection ii) segmentation of sensor observations, iii) knowledge modelling, iv) AR and v) activity learning.

As the sensor data are collected from the smart environment in the initial phase of AR, the segmentation phase of organising the observed sensor based on the ongoing activities or detecting new activities performed by a single inhabitant in composite scenarios is a major challenge being investigated in this paper. Fig. 1 illustrates the five phases and role of segmentation to distinguish sensor observations actions relative to the ongoing activity to support AR and activity learning. In order to make segmentation decisions, prior knowledge model is required to verify association links such as what everyday object is the sensor attached to, contextual information (i.e., location, time and ambient attributes) of the object and what ADL(s) is this object is used for. The process of defining these complex sets of relationships between sensors, everyday objects/environments and ADLs is a challenge that has been investigated in the past studies and they can be categorised as syntactical, semantical and pragmatic in information theory[5]. In syntactical approach, a concept can be represented as statements in a given syntax and assumes any two non-syntactically equivalent statements are independent. In contrary, the semantical approach is concerned about representing the meaning of a concept using relationships[5, 6], hence, the same concept can be syntactically represented in more than one statements and mean the same thing. The pragmatics studies the relations between a concept and inhabitant. The benefit of adapting syntactical approach is that knowledge can be structured using defined syntax, queried and interpreted by the machine, however, suffering from the flexibility of expressing intricacy of relationships and meaning between two concepts. The semantic theory has its roots from semiotics in philosophy which in general is a study of signs and its significations (meaning)[7]. These signs can be words, images, sounds, gestures and objects. Hence, the semantical theory is studied heavily in cognitive philosophy, natural language and machine learning [8]. This paper further explores this notion to encapsulate generic knowledge using semantic theory, and inhabitant specific preferences with a pragmatic approach which has been paid little attention in previous studies of segmenting sensor events.

The common approaches to obtaining domain-specific knowledge and modelling are known as data-driven, knowledge-driven and hybrid approach. In the data-driven (DD) [9, 10] approach, activity models are generated after processing pre-recorded datasets using generative or discriminative classification techniques. In contrast, the KD approach is where domain experts in the field of interest conceptualise and intricately describe factual elements of the being into a model that is interlinked, known as the ontological model. The 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. The KD approach overcomes the “cold start” issue by not processing pre-recorded dataset, however, falls short in handling unseen or uncertain data [1]. The shared 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 [11, 12, 13] 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 approaches [12] are influenced by the selection of modelling approach and the quality of the segmented sensor data for reasoning. Activity classification is a two-fold process: verification of the relationships between ADLs and a set of sensor observations; and validation of the activity occurring with a degree of confidence. Whereas, the activity learning approaches evolve initial knowledge model by analysing the AR results and un-/related sensor observations during the period to discover new activities, patterns, and inhabitants preferences in real-time or offline. The data-driven approaches are commonly adopted for this purpose. The activity classification and activity learning topics are beyond the scope of this paper, nevertheless, for more details see [14, 15]. The segmentation approaches, however, mainly rely on verification results of the activity classification process to reduce the computational complexity and time delay to incrementally grow the set of segmented data for a given activity.

The data collection and monitoring of the environmental changes and nearly every inhabitants actions can be now sensed with the advancement of ubiquitous sensing technology. There are a wide variety of sensing technologies available to collect meaningful data and 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 (or embedded) and wearable sensing[16]. The ambient sensing is performed to collect environmental data such as temperature, luminosity, motion, sound, and door/window opening. The dense sensing is used to monitor inhabitants 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[17]. 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 neural activities in 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 to later perform AR.

There are a number of human factors that further increase the complexity when designing the semantical knowledge model, developing segmentation and AR algorithms. One of which is the nature in which one can perform single or composite (multiple) ADLs at a given time as illustrated in Fig 1. Individual ADLs (A1, A2 and A3), can have a set of atomic actions ({abcdef}, {123456} and {XYZ}) which can be performed in any order. A single ADL (A1) can also be performed along with multiple other ADLs; either incrementally (i.e. A1 then A2), concurrently (i.e. A1 with A2), and in parallel (A2 and A3 running simultaneously). Furthermore, an individual is subjected to follow a 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.

In the remainder of the paper, the existing studies related to segmentation, semantical knowledge modelling and AR process are reviewed in Section 2. A novel segmentation method and algorithm is then proposed in Section 3 with system implementation details and evaluation results in Section 4 and 5. The conclusion and future research direction is discussed in Section 6.

2. Related Work

Recent studies have applied time series (fixed/dynamic time window[18, 19, 20]), statistical and probabilistic [21] based approaches which have failed to separate sensor observations based on the relation to ongoing activity in real time. Therefore, KD approach has received an increasing amount of interest to express complex relationships between sensors and domain-specific knowledge. For instance, studies in [22, 23, 24, 25, 20, 26, 27, 28] 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), the temporal relationship of activities, static/dynamic timing window protocol, Semantic Web Rule Language (SWRL) based rules, and SPARQL Protocol and RDF Query Language (SPARQL) queries.

These studies, [22, 23, 24, 25, 20, 26, 27, 28], however, do not directly inspect each sensor event as they arrive and then segment to the appropriate queue related to ongoing activities. Instead, the continuous queries or rules are executed on events stored in the database and knowledge model without using any automatic reasoners to determine the relationship between events and ADLs. For example, work in [28] proposed extension of C-SPARQL, an extension to SPARQL querying language where individual sensor events in a stream are annotated with a timestamp and continuously queried using a specific window size. The key limitations of the approach are the classical multi-query optimization problem where the challenge is to identify the common parts, adapting/reformulating the order in which queries are executed and the ability to dynamically change the window size. Another work in [22, 23] 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 an attempt to use generic ontology reasoner is made, however, it is unclear if reclassification of the whole ontology is done incrementally or not. In the 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 [29]. However, each time the new rules are added or updated to enrich the knowledge base (KB), the whole ontological model is reclassified. In addition, managing models generated using generic and inhabitant specific rules exclusively adds to the complexity further.

Similarly, work in [30] presents a layered ontology and complex event processing (CEP) engine based framework, namely, AALISABETH, to segment the sensor observations. 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 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 an 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 affected 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 inhabitants preferences when performing a generic ADL, however, it is extendable.

Work in [27] 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 amongst more than one activities and it can either be part of a main activity or precondition actions for another 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 the scope for improvement in terms of recognising other scenarios. Another work in [31] leveraged evidential theory and proposed three segmentation algorithms based on location, activity model and dominant-centred actions for non-interleaved and interleaved activities. The location and activity model-based segmentation algorithm fall short in distinguishing activities when performed in the same location and with similar everyday objects for activities compared to the dominant algorithm. There is a little implementation detail provided by the authors, however, one of the key limitations of all the three algorithms is the lack of support for user preferences and a reasoner to automatically detect and recognise the activity.

This paper made five contributions by proposing: (i) a semantic-enabled segmentation approach which combines generic and personalised ADL knowledge that enables simple and composite ADLs to be recognised in real-time; (ii) a KB model capturing the relationships between entities in the house and ADLs; (iii) a light-weight mechanism to manage inhabitants specify preferences for conducting a given ADL; (iv) a semantical decision engine algorithm; (v) system implementation details and a prototype to evaluate the approach and present the findings.

3. The Semantical Segmentation Approach

The semantic theory based segmentation approach is proposed which analysis the relationship of the sensor event with an everyday object and its significance as an action to a set of known ADLs. This will enable disentangling composite activities with actions performed in no particular order and organising them separately to allow further activity classification and learning tasks. A knowledge modelling building block is developed in section 3.1 which conceptualises and captures the environmental context (i.e., ambient attributes, everyday objects, location, sensors), generic and inhabitant specific preferences to perform ADLs and their semantic relationships into an ontological model. A semantical decision engine is developed in 3.2 to make segmentation decisions based on three inputs: the new observed sensor event, the ontological model and a set of previously segmented sensors for a given activity. A notion of multithreading is adapted to separate tasks of buffering sensor data stream, event recycling, decision engine, managing ADL threads and manipulating data from the triplestore database (TDB). This multithreading mechanism to semantically segment sensor event is described with a pseudo algorithm in section 3.3.

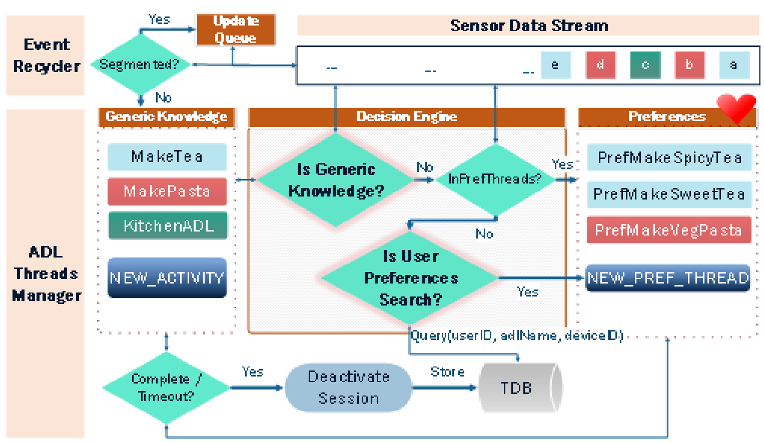

Figure 2: Overview of the semantically enabled segmentation approach with generic (T-box) and preferences (A-box) KB for reasoning.

Fig. 2 depicts the overall segmentation approach. As the sensor events are initially added to the data stream, multiple ADL threads, generic and preference, analyses the sensor events using decision engine and store the relevant events independently. Therefore, one sensor event can be shared between two different activity threads with different ADL goals. For instance, opening

Fridge action detected by sensor e at T_n can be shared with $MakeTea$ ADL and *MakePasta* ADL thread. The ADL threads manager creates new ADL thread (*NEW_ACTIVITY*) only when the sensor event is not part of any ongoing ADL threads otherwise the event recycler thread updates the sensor data stream. There are two types of ADL threads being created to capture generic actions (sensor b attached to *PastaBag*), for a given activity (*MakePasta*), and if the observed event (sensor d attached to $HotSauce$ at T_n) is part of the personalized actions for that activity (i.e., *PrefMakeVegPasta*). The decision engine determines if the new sensor event, along with the previous set of sensors for a given activity is part of the pre-defined generic set of actions by performing semantic reasoning and invoking queries the TDB for personalised actions. The new preference thread (*NEW_PREF_THREAD*) is only created when the new sensor event is part of a personalised action for a given ongoing activity and there is no active preference thread. Moreover, each ongoing activity thread with the segmented set of sensors data will enable further validation of AR accuracy, timeout and completion procedures, i.e. storing relevant information and prompting the inhabitant when appropriate in future work.

3.1. ADL Relationships Modelling

The key building block of ADL modelling consists of three phases; (1) environmental context (EC) modelling, (2) semantical relationships (SR_i) modelling and (3) personalised ($Pref_j$) object interactions. In the first phase, the object-oriented notion (classes and instances) is adapted to conceptually describe the physical or metaphysical entities (ET_k) and their attributes in the environmental context (EC) as classes (C). The key entities considered are a person (X_n), rooms (Location, L_m) and ambient characteristic (AC_p), sensor characteristics (S_o) and everyday fixed/portable objects (Obj_x); see eq. 1.

$$EC = \{X_n, L_m, AC_p, S_o, Obj_x\} \tag{1}$$

The second phase records semantic relationship (SR) properties between EC classes and ADLs. The instances of EC classes (i.e., everyday objects) are then created for sensor environment (SE) to create a relationship (R_e) between sensor event, object it is attached to and this objects use in ADLs; see eq. 2. This abstraction in ADL actions description encourage decoupling, reuse and adding the further meaning of the actions to the activity using R_e. For example, *MakeTeaADL* (subset of *MakeHotDrinkADL*) class described the actions using *hasHotDrinkType* (R) relationship property with Tea (C) and the characteristics of the property are described to be only used for *MakeHotDrinkADL* (*domain*) and everyday objects that are used for *HotDrinkType (range)*. This means if no other ADL that is a subset of *MakeHotDrinkADL* that has a *hasHotDrinkType* property with *Tea*, it can be deduced that this action is potentially a part of *MakeTeaADL*. Similarly, other actions for *MakeTeaADL* can be described using *hasUtensil*, *hasContainer* and *hasAddings* properties for using the kettle and adding sugar and milk to the teacup. Fig. 3 show the relationships between a set of EC classes and *MakeTea* ADL to show the meaning of inhabitants action.

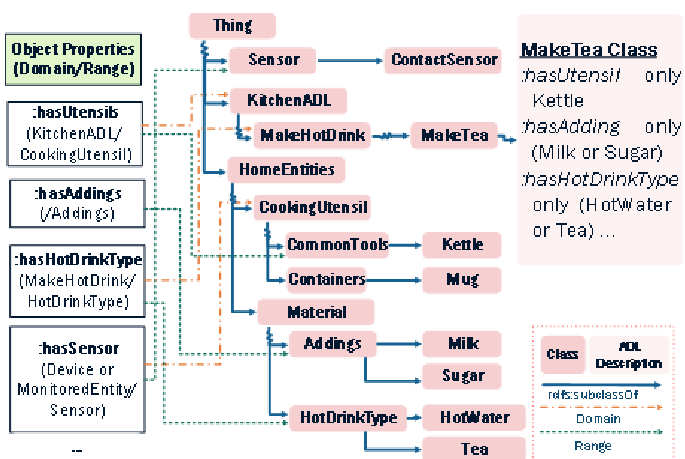

Figure 3: Semantical relationship properties between everyday objects, set of actions for MakeTea ADL and sensor characteristics.

Moreover, the sensor environment (SE) information is then encoded to describe existing set of EC items available in the given residential environment and the sensor attached to it as instances (I_w). Therefore, instances of $EC(iEC_w)$ such as environmental objects ($iObj_w$) and sensor (iS_w) with their relevant classes (C_n) are explicitly described with the relationship (R_e) between them initially. For example, to_1 is an instance of *ContactSensor* (S) that *isAttachedTo* (R) a *RedKettleObj*$_1$ $(iObj_w)$ which is a class type of *Kettle* (Obj_x). The observed values/states of an iS_w are stored as primitive data types (pt_u) for a single observation or creating another instance of an observation class containing the primitive data for multiple observations; see eq. 3.

$$SR = ADL_n(R_e, EC_n) \rightarrow R_e \rightarrow SE; \tag{2}$$

$$SE = I_w(R_e, S_o) \rightarrow R_e \rightarrow I_w(R_e, ET_k) \| I_w(R_e, \{pt_u\}) \tag{3}$$

The final phase is to capture inhabitant specific preferences $(Pref_j)$ and extend the generic SR description of ADLs. It is important to keep the generic and personalised sets of ADL description disjointed to avoid generalising or assuming both must be actioned to complete the activity. Therefore, instances that are members (R_e) of *Preference* and ADL_n classes are created to capture actions or ambient attributes using iEC_w that are specific to a person (X_s); see eq. 4 and 5. For example, an individual *Bob* (I) who is a type of *Male* (C) has set of instances of *Preferences* that are linked with *hasPreference* relationship (R). An example of a preference instance is *BobMakeSpicyTeaPref* $(Pref)$ which is a type of *Preference* (C) and *MakeTeaADL* (ADL) with a set of iEC instances, i.e., *GingerObj*(I) and *CinnamonObj*(I). This statement means that *Bob* has a preference to make tea and he may/like to put a ginger and cinnamon in his tea.

$$X_n = I_w(R_e, Human \subseteq Male) \rightarrow R_e \rightarrow Pref_1, ...Pref_j \tag{4}$$

$$Pref_j = I_w(R_e, ADL_n \sqcap Preference) \rightarrow R_e \rightarrow I_w(R_e, iEC_w) \tag{5}$$

3.2. Semantic Decision Engine

The decision engine takes three inputs, processes them into two stages and outputs the updated results. The three inputs are (1) semantic-based KB model created in section 3.1, (2) activity thread (AT_n) attempting to find relations with the (3) new sensor event (e_m). Each AT_n contains structured information about generic and preferred actions observed as sensor events, ADL class and list of preferences matched that are associated the inhabitant. The two-stage decision-making process updates the activity thread accordingly as the new sensor events are inspected incrementally for any association.

$$\begin{aligned} AT_n = \{&tbox[class : someADL, s\{..., e_m\}], \\ &abox[Pref_j[name : somePref, s\{..., e_m\}]\} \end{aligned} \tag{6}$$

In the first stage of decision-making process, generic semantical relationships are traced from EC to SR and SR to SE compared to inverse when developing the KB model [32]. Therefore, the metadata of a sensor observation em is analysed to find the ET the sensor is attached to and deduce the potential Rn with a set of ADL_n description. This metadata within KB consists relationship properties such as domain and range for a given ET. Therefore, the association between ET, (i.e., everyday objects) and ADLs can be automatically inferred using semantic reasoners or simply querying the KB model. This process is known as terminology box (T-box) reasoning [33].

The second stage is only executed when the result returned from T-box reasoning identifies any conflicts with the ADL class description. The conflicts can be raised when a given sensor attached to an ET is forced to be part of a given ADL which is outside the restricted set of ET_k. In this case, it is assumed that ET is part of inhabitants preferences or part of a new set of actions for $ADLn$. The preferences are currently pre-defined and stored as individuals containing a list of iEC_s that an inhabitant prefers to use to perform a given ADL. Therefore, semantic queries are made to extract all preferences of the inhabitant ($userID$) for a given ADL ($adlName$) that as sensor observation ($deviceID$) as an action. This process is known as assertion box (A-box) reasoning.

The semantic reasoner carries out several tasks using T-box and A-box knowledge which includes but not limited to: satisfiability, subsumption, consistency checking equivalence, disjointness, and instance checking [32, 34]. The satisfiability task is to ensure the class description (axioms) is not contradictory. The subsumption task ensures class B satisfies all the inheriting properties (R) of parent class A. The consistency checking ensures classes and their instances do not violate the axioms descriptions. The instance checking ensures the relationships with other instances are within the boundary of a set of classes it can subsume. The equivalence task is to match the two concepts with respect to its properties in contrary to disjointness tasks. The conjunctive querying answering is performed at the second phase of decision engine to identify inhabitants preferences with a given ET using relationships between instances of EC and ADLs.

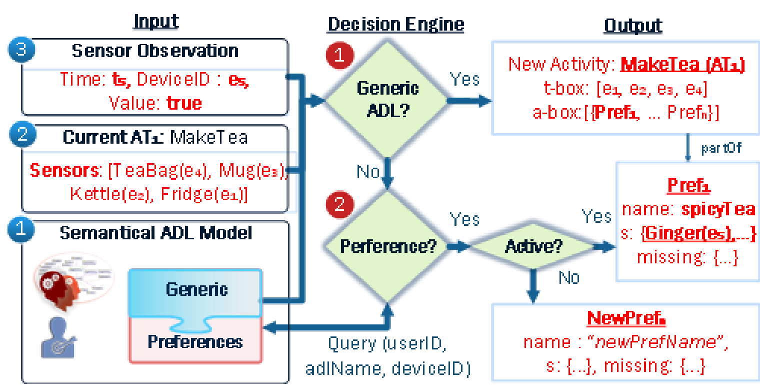

Figure 4: Semantic-based Decision Engine; Input: new sensor observation (e_5), current activity with set of sensors and semantical ADL model, Output: new activity result

Fig. 4 illustrates the three inputs taken by the decision engine to verify if the new sensor observation Ginger(e_5) is part of the generic/personalised action of the ongoing *MakeTea* activity (AT_1). Initially, a new activity thread, AT_1, is created to add the first sensor observation, *Fridge* (e_1), into the empty set of sensors and the results returned from two-stage reasoning process. In this case, e_1 is inferred by the generic T-box reasoner to be part of *KitchenADL* in the first stage of decision engine. As the new sensor event, e_2 occurs, the current AT_1, temporarily add it to the list $\{e_1, e_2\}$ and perform the generic reasoning again with the same activity result. This means that the action is part of A_1, however, more than one sub-activities share the same actions. Similarly, other events were added to $AT_1 = \{e_1, e_2, e_3, e_4\}$ as they occurred with new *MakeTea* activity name which is a descendant class of *MakeDrink* and *KitchenADL*. Until now, only first stage decision due to generic nature of the ADL actions. The next sensor observation, e_5, is attached to *Ginger* running any personalised actions. The activity name, *MakeTea* of A_1 and the new sensor observation $Ginger(e_5)$ is used to perform subsumption reasoning in the first stage of decision engine and returned inconsistency in ADL description error. In the second phase, the decision engine checks if the $Ginger(e_5)$ sensor is part of an inhabitants preference(s) stored in the triplestore and add it to A_1. In this case, *spicyTea* preference was identified and as there were no sub-activity preference threads already active for A_1, new thread $Pref_1$ was created along with other missing *spicyTea* actions.

$$
\begin{aligned}
AT_1 = \{&tbox\{name : makeTea, s : \{e1, e2, e3, e4\},\\
&abox[Pref_1[name : spicyTea, s : \{e5\}, missing : \{...\}]]\}.
\end{aligned}
\tag{7}
$$

3.3. Segmentation Algorithm

The pseudo algorithm defined in TABLE 1 illustrates the segmentation process, use of decision engine and multithreading mechanism discussed in section 3 to separate sensor observations. The algorithm is performed by the ADL threads manager and it is broken down into four stages. The first stage is to iterate over all the active T-Box threads and use the current list of sensors observations in each thread along with the observed sensor event (en) being investigated to execute a new T-Box inferencing result. This new result will return a representative OWL class of an ADL and it is then compared to the current activity class to decide if the sensor event is part of the ongoing activity. The comparison is made with the result class and current class if they are equal or if the new result is within the sub-classes/hierarchy of the current ADL class. If the result is true, the sensor observation is stored as part of the activity, no other thread is created and waits for the next sensor event. In the second stage, where the result is false or if there is a conflict in ADL action description, all the active A-Box threads are check if the observation is part of it. A binary flag (found) is used to indicate if A-Box thread has already processed the sensor or not. The third stage is where the decision is made whether to create a new A-Box thread or T-Box if the found flag is still false. The A-Box thread is only created if the new sensor event is a part of an ongoing activity and has some user personal preference(s) stored in the triplestore; for which, multiple A-Box threads are executed. Otherwise, it is assumed that the new sensor observation is a start of a new activity, hence, starting a new T-Box thread. The final stage is where all the housekeeping for the sub-threads and the process of re-evaluating the session timeout window, timeout and further tasks such as activity recognition, learning takes place based on the data of the segmented set of observations. Details of the semantical segmentation mechanism can be found in our previous work [35, 36].

```
Input:  events[e1, ei ..., en], userID
Output: void

//1) e is part of ongoing ADLthread or start of a new ADL
boolean found = false; List<Sensors> tempList; List<Preference> prefs;
for each TBoxThread t: tboxThreads
   tempList = t.getCurrentSensorsList();   tempList.add(e);
   // returns representative class for a given list of sensors
   OWLClass c = DecisionEngine.runTBoxInferencing(tempList);
   String adlName = c.getIRI().getFragment();
   if !t.isSubClassOREqualToExisitingTBoxThreads(c)
      //2) search active preferences threads
      for each ABoxThread at: aboxThreads
         at.checkLinks(tempList)? found = true; end for each
  //3) query database to matching inhabitant preferences
  prefs = DecisionEngine.isAboxPreferences(en, adlName, userID);
end for each
//3) create new A-box or T-box if not part of ongoing active thread
if !found && prefs.size() > 0
   aboxThreads.add(newABoxThreads(prefs));
   // otherwise new create new ADL thread (T-Box)
else if !found && !prefs.size()>0
     tboxThreads.add(newTBoxThreads(e)); end else if
end if
//4) maintain completed and timeout scenarios
manageCompeletedAndTimeoutCases(tboxThreads, aboxThreads);
```

Table 1: Pseudocode for Semantical Segmentation Algorithm

4. System Implementation

An android mobile application and RESTful web service have been used to create a service-oriented architecture (SOA) system. An 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 4.2. 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 presents the information to the inhabitant. Fig. 5 shows a snapshot of how concurrent actions of three activities are separated into different threads and presented on the Android application. Details of the SOA implementation and multithreading concept can be found in previous studies [37, 38].

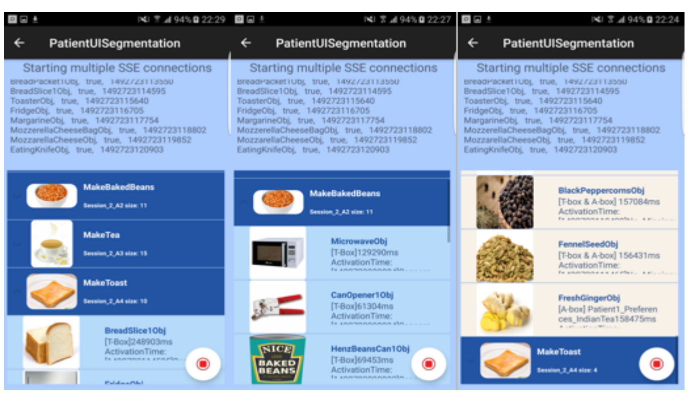

Figure 5: Segmentation results for three concurrent ADLs

4.1. Ontological Modelling

The generic knowledge for segmentation is represented using semantic web framework. This framework provides web ontology language OWL to formally express the complex knowledge into classes, relationships (object & data properties) and data (individuals) [39]. In addition, common vocabularies are used to represent the KB and encourage sharing across applications to create an ever-growing, human and machine-readable web of knowledge. There are a number of automatic reasoning tools available to read this KB to identify inexplicit facts based on relationship definition and the selection of a reasoner is elaborated in section 4.3. 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 this, EC, SR, and $Pref$ are modelled in three phases using ontology editor tool named Protg. Initially, EC concepts such as everyday objects, person, sensor characteristics and location were modelled as classes. Fig. 6 illustrates the fragments of EC classes and their subclasses.

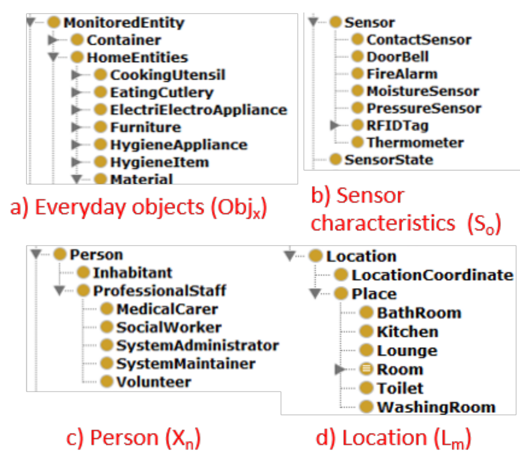

Figure 6: Conceptualising environmental context (EC) into Classes

In the second phase, the EC classes are used to define SR between ADL classes and describe their actions iteratively using object properties. Fig. 7 partially describes the $MakeTea$ ADL in Protg. 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 properties and the classes of everyday objects; for the *MakeTea* ADL are described using object *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 a *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 its 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 ($\leq, \geq, \equiv$). Other common operators are also available and can be used to increase the expressivity of the ADL model in terms of class, relationships and data. Similarly, the other 12 subclasses of *MakeDrink* and *MakeMeal* ADL classes are also described. As multiple relationships are 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.

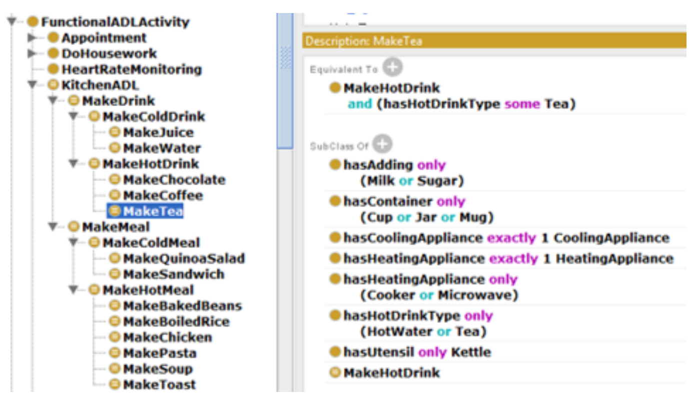

Figure 7: Partial description of MakeTea ADL with Semantic Relationship (SR) with environmental context (EC) in Protg.

Finally, the inhabitant specific preferences (A-Box) are captured by creating individuals with a direct relationship with instances of sensors in order to avoid the inconsistency in ontology description for generic knowledge. In the generic knowledge, not all adding (ingredient) for *MakeTea* ADL are defined and ingredients such as *FreshGinger* and *CinnamonSticks* are subjective to the individual. Hence, forcefully adding ingredients in an instance that is the type of *MakeTea* ADL will result in the inconsistent ontology as highlighted by the explanation window in Fig. 8. Therefore, instances of preferences are associated to the inhabitant and to a given ADL class which have a list of sensors that are attached to the everyday objects and other attributes. Fig. 9 presents an example of three inhabitant preferences. The top section presents individual named, *Patient1_Preferences_IndianTea*, which has a type of *Preference* class for *MakeTea* ADL class along with a list of sensors using *hasSensor* object properties and data properties to describe other attributes such as preference

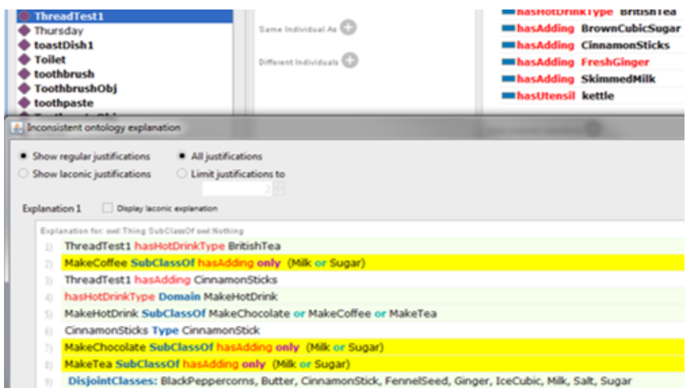

Figure 8: Inconsistency on hasAdding object property due the restriction applied to MakeTea ADL class.

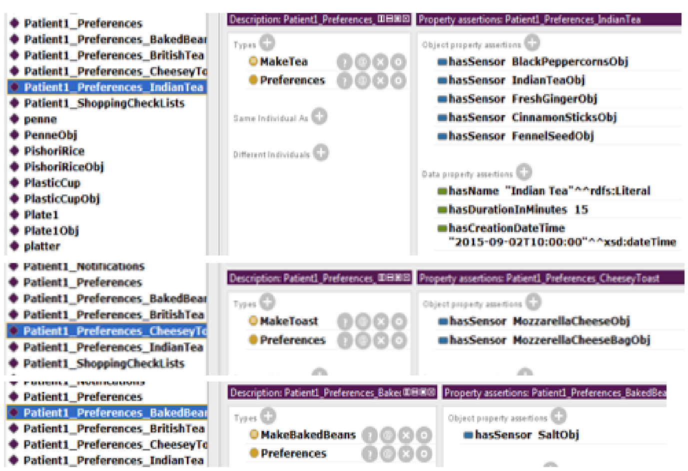

Figure 9: Inhabitant preferences as individuals with a list of sensors

name and creation timestamp. Similarly, other preferences are shown in the middle and bottom of the figure to describe *MakeToast* and *MakeBakedBeans* preference.

Another method is available 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 conflicts in the knowledge that may arise. In addition, inhabitant specific reasoner will need to be created and maintained [40] 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, the first method is selected as it is lightweight, and no inhabitant specific reasoner is required to be running. The SPARQL Inferencing Notation (SPIN) [41] rules or just a SPARQL query language can be executed on the triplestore to retrieve multiple inhabitants preferences for a given ADL class simultaneously. Therefore, this method is considered appropriate during the segmentation phase as the inhabitants 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.

4.2. Multithread Segmentation Process

The multithreaded segmentation processes are depicted in Fig. 10 where actions for *MakeTea* and *MakeToast* ADLs are performed concurrently. The generic and preferred actions are observed at a given time (t_n). The T-box activity thread (AT_1) is initially created when the *cupObj* sensor is activated at t_1. The AT_1 continuously stores the events into the thread if the decision engine infers an association with generic ADL class in the ontological model or personalised preference(s). The object attached to the *cupObj* sensor is queried from the triplestore, added to new individual and incremental T-box reasoning is conducted. 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 AT_1 performs T-box reasoning with observed sensor *fridgeObj* along with previous sensor(s), in this case, *cupObj*. The decision engine returned a new result, *KitchenADL* class and it was compared against the current *ADLActivity* class for equivalent or subsuming class. In this case, the subsuming condition is satisfied and stores the *cupObj* and *fridgeObj* sensor events in the AT_1.

Similarly, *milkObj, kettleObj* and *indianTeaObj* sensor events are processed by AT_1 where the ADL classes are incrementally classified, and the sensor events are stored in the thread. However, the *freshGingerObj* sensor event is not described as part of a set of adding in the generic *MakeTea* ADL description, therefore the decision engine returns with traceable conflicts. The decision engine then performs A-box reasoning to find any inhabitants preferences related to *MakeTea* ADL containing *freshGingerObj*. Multiple preferences could be returned, however, in this case, only one preference named,

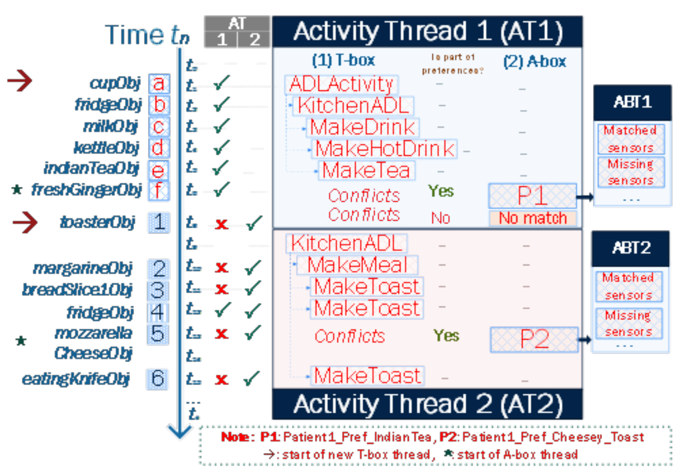

Figure 10: Concurrent actions for *MakeTea* and *MakeToast* ADL and segmentation process to create generic (AT_1 and AT_2) and preference (APT_1 & APT_2) threads when required.

Patient1_Pref_IndianTea (P_1) is returned as a result of SPARQL query. A single A-box sub-thread (APT_1) is created with other missing sensors and other relevant information from the preference into the thread. The APT1 thread then inspects the incoming sensor events and updates the missing and matched sensors list independently. AT_1 thread and the sub-thread(s) for A-box reasoning can continue inspecting unfolding events in the data stream until the completion criteria are 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/timeout criteria for the ADL will be inspected in future work.

The next set of actions for *MakeToast* ADL are observed between $t_8 - t_{14}$ and inspected by AT_1 but only one shared *fridgeObj* event is stored. The ADL manager running in parallel inspects the sensor events in the queue and detects *toastObj* is not part of the *MakeTea* ADL class in AT_1 and APT_1 threads. Therefore, another T-box activity thread (AT_2) is created *MakeToast* ADL as depicted at the bottom-right of Fig. 10. The same process is described for AT1 is executed for the AT_2 thread to capture events from $t_{10}-t_{15}$ to AT_2 thread with one conflicting *mozzarellaCheeseObj* observation. Therefore, the APT_2 thread was created when identified by decision engine that *mozzarellaCheeseObj* is part *Patient1_Pref_CheeseyToast* (P_2) to perform the *MakeToast* activity.

4.3. Reasoner and Supporting Tools

A reasoner is a software tool developed to perform A-box and T-box reasoning by the decision engine to perform tasks such as consistency check of the ontological model and derive new facts from the KB dataset. There are a number of reasoners developed over the years and most of them support first-order predicate logic [32] reasoning or procedural reasoning (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 affected by the changes [42], 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 [32, 33] describe a number of popular reasoners using large ontologies, compare against their key features and categorise according to their characteristics. The incremental 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 Protg and NeOn toolkit. OWL API and Jena API both support the Pellet reasoner to programmatically perform reasoning and KB 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 outperforms in subsumption query [32].

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 2: Single Activity Sequences Example

5. Evaluation

5.1. Experiment Design

The actions for three ADLs were scripted in no particular order to perform with only generic actions and another with the inhabitants preferences; namely, *MakeTea, MakeToast* and *MakeBakedBeans*. The relevant actions for the generic ADL and some inhabitants preferences are described in TABLE 2. These three ADLs are first tested individually and then combined to create composite activity scenario; incremental, concurrent and parallel; see TABLE 3. A total of 30 activity scenarios were created for the experiment with the 10ms interval between sensor events. The degree of accuracy to recognise an activity scenario is calculated in percentage by matching and tallying actual sensors events segmented divided by a total number of sensors events activated for each ADL. The average classification time is calculated by taking sensor observation segmented time by the reasoner minus the sensor observation time recorded for each activity scenario. The unexpected sensor observations within the activity scenario are omitted and recorded separately when calculating the accuracy and average classification time for the activity. In addition, a number of duplicate activity threads created in the activity scenario are also recorded to see the effect on the overall classification times. 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 the i7 2.60GHz processor, 2 cores, 4 logical processors and 8GB RAM. The sensor events are currently simulated

Activity Comb.	ADL Sequences	Expected no. threads	Actions	
			Gen. (G)	+ 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 3: Combinations of Simple activities

due to a limited number of sensors and time.

5.2. Results

The average segmentation time taken per sensor event for single activity is 3971ms in contrast to 62183ms for composite ADL scenarios as shown in TABLE 4 and TABLE 5. The result in TABLE 4 shows that all the sensor events for a single activity case scenario were adequately placed in the correct thread with 100% accuracy. Only the MakeTea activity case scenario created additional threads with more than double the average time when processing 9 generic actions and 4 preferred actions. On the other hand, TABLE 5 shows 20 out of 24 activities performed in a composite manner or 572 out of 585 sensor events were added to the relevant thread, giving 97.8% accuracy. However, the segmented activity threads captured a total of 71 additional unexpected sensor events in the segmented threads which are not necessarily incorrect, i.e., multiple spoon objects or heating/cooling appliances when performing multiple activities interweavingly. Furthermore, 29 additional threads were created and failed to classify any ongoing activity.

5.3. Discussion

To compare against recent KD studies presented in section 2, the accuracy of single and composite activity segmentation for evidential theory-based approach [31] is 81.8% and 76.2% on average and ontology and temporal [23] achieved 100% and 88.3%, respectively. Therefore, there is a significant evidence that

Activity Type	Case Type	In relevant thread	Unexp. actions in thread(s)*	Excess thread (s)	Avg. time (ms) +
MakeTea	G	9	0	0	2394.67
MakeToast	G	7	0	0	2468.57
MakeBaked Beans	G	8	0	0	2372.25
MakeTea	G+P	13	0	1	10828.85
MakeToast	G+P	9	0	0	3786.87
MakeBaked Beans	G+P	9	0	0	1972.44
Total	6	55/55	0	1	3970.61 (avg)

Note: * excludes additional thread(s) actions, + including excess threads

Table 4: Single Activity performed in no Specific order with Generic and Personal Preferences

Activity Comb.	Test cases	All actions in the thread(s)?	Excess thread (s)	Unexp. actions in the thread (s) *	Total Avg. time + (ms)
Inc.					
AC1	G	✓ 16	1	1	36330.64
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.)

Note: * excludes additional thread(s) actions, + including excess threads

Table 5: Multiple activities performed in a composite manner

the proposed approach improves the accuracy of sensor segmentation with 100% and 97.8%, respectively. In addition, user-preferences are taken into consideration by adopting basic query based approach and automatic Pellet reasoner for generic KB reasoning compared to their counterparts which adapt solely query-based approach inheriting classical multi-query optimization problem in [28] and [30]. Nevertheless, one of the benefits for adapting multi-query approach is that higher performance and scalability can be achieved, however, suffer from the expressivity capabilities of KB due to explicit query development/maintenance efforts and the ability to use automatic reasoners.

The proposed method in this paper seeks to strike a balance by taking advantage of incremental Pellet reasoning feature introduced by Pellet which was developed in above challenges in consideration and query-based approach capabilities to manage the changing user-preferences. The average segmentation

Studies (by year) / Features	C-SPARQL [28], 2010	Evidential theory [31], 2013	Onto. and temporal [23], 2014	AALIS ABETH [30], 2015	Proposed
Knowledge expressivity	High	High	High	High	High
SPARQL query support	Yes	Yes	Yes	Yes	Yes
Automatic reasoner support	No	No	Yes	No	Yes – incr. Pellet
Direct stream inspection	No	Yes	Yes	No	Yes
RDF stored	Yes	NA	Yes	No	Yes
User prefs. support	No	No	No	No	Yes
Sliding window support	Yes - Fixed size	No	Yes	Yes	No – future work
Potential scalability issue	Low	Med. – High	Med.	Low	Med. – High
Accuracy: S; C (%)	-	81.8; 76.2	100; 88.3	-	100; 97.8
Average time: S; C (ms)	-	-	-	-	3971; 62183
Note: S: simple activity, C: composite activity					

Table 6: Summary of recent KB approaches

time information is not available in the presented KB studies; however, the proposed approaches observes 3971ms and 62183ms with sensors events activated at the 10s interval for simple and composite activities scenarios. These results are still not suitable for the real-time system at this stage. However, the optimisation opportunities such as multi-thread safe reasoning [43], ADL threads management, partitioning workload to graphics processing units (GPUs) [44], and using machine with higher cores (i.e., quad-core, octa-core CPU or higher) to support more two threads execution at same time remain an open challenge. TABLE 6 presents a summary of the key components of the recent KB studies presented in section 2 against the proposed semantical segmentation approach in this paper.

6. Conclusion and Future Work

A 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 individual. Upon sensor activation, the event is inspected by one or more active ADL threads running in parallel. Each ADL thread relies on a two-stage decision engine to find any association with observed sensor event. The decision engine conducts T-box reasoning with generic KB in the first stage and A-box reasoning with observed sensor event and inhabitant specific preferences by querying the triplestore in the second stage. The second stage of decision engine is only invoked when the use of entity on which observed sensor is attached to has a contradiction or not been explicitly specified in generic ADL description. The ADL thread discards the observed event when decision engine has failed to find any relationship. When the whole set of 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 30 test scenarios. The results indicate an improvement in segmentation accuracy compared to the counterpart studies with 100% and 88.3% for single and composite ADL scenarios with an average time of 3971ms and 62183ms. The main bottlenecks for high processing time are the synchronised incremental reasoning and duplicate ADL threads creation which ultimately created additional reasoning tasks and slowed down the overall process on the machine which was limited two cores. A future study is proposed to address above shortfalls, add support for rules based reasoning and integrate dynamic time series analysis to detect start and completion of the activity. The study would then focus on accurate fine-grained AR and learning algorithms.

References

[1] S. Zolfaghari, R. Zall, 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, Vol. 2016-March, 2016, pp. 3339–3348. doi:10.1109/HICSS.2016.417.

[2] C. Debes, A. Merentitis, S. Sukhanov, M. Niessen, N. Frangiadakis, A. Bauer, Monitoring Activities of Daily Living in Smart Homes, IEEE Signal Processing Magazine (March) (2016) 81–94. doi:10.1109/MSP.2015.2503881.

[3] Z. Tang, J. Guo, S. Miao, S. Acharya, J. H. Feng, Ambient intelligence based context-aware assistive system to improve independence for people with autism spectrum disorder, Proceedings of the Annual Hawaii International Conference on System Sciences 2016-March (2016) 3339–3348. doi:10.1109/HICSS.2016.417.

[4] B. J. Banks, The "Age" of Opportunity, IEEE Pulse 8 (2) (2017) 12–15.

[5] Y. Zhong, A theory of semantic information, China Communications 14 (1) (2017) 1–17. doi:10.1109/CC.2017.7839754.

[6] A. Tarski, The Semantic Conception of Truth: and the Foundations of Semantics (1944). doi:10.2307/2102968.
URL http://www.jstor.org/stable/2102968?origin=crossref

[7] P. Vickers, Understanding Visualisation: A Formal Foundation using Category Theory and Semiotics, IEEE Transactions on Visualization and Computer Graphics X (X) (2013) 1–14.

[8] Y. Wang, Formal rules for concept and semantics manipulations in cognitive linguistics and machine learning, in: 2017 IEEE 16th International Conference on Cognitive Informatics Cognitive Computing (ICCI*CC), 2017, pp. 43–50. doi:10.1109/ICCI-CC.2017.8109728.

[9] L. Chen, C. D. Nugent, H. Wang, A knowledge-driven approach to activity recognition in smart homes, IEEE Transactions on Knowledge and Data Engineering 24 (6) (2012) 961–974. doi:10.1109/TKDE.2011.51.

[10] L. Chen, C. Nugent, G. Okeyo, An ontology-based hybrid approach to activity modeling for smart homes, IEEE Transactions on Human-Machine Systems 44 (1) (2014) 92–105. doi:10.1109/THMS.2013.2293714.

[11] L. Chen, C. Nugent, J. Rafferty, Ontology-based Activity Recognition Framework and Services, Proceedings of International Conference on Information Integration and Web-based Applications & Services - IIWAS '13 (2013) 463–469doi:10.1145/2539150.2539187.
URL http://doi.acm.org/10.1145/2539150.2539187{%}5Cnhttp://dl.acm.org/citation.cfm?doid=2539150.2539187

[12] L. Chen, G. Okeyo, H. Wang, R. Sterritt, C. Nugent, A systematic approach to adaptive activity modeling and discovery in smart homes, Proceedings - 2011 4th International Conference on Biomedical Engineering and Informatics, BMEI 2011 4 (2011) 2192–2196. doi:10.1109/BMEI.2011.6098760.

[13] P. Barnaghi, W. Wang, L. Dong, C. Wang, A linked-data model for semantic sensor streams, Proceedings - 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, GreenCom-iThings-CPSCom 2013 (2013) 468–475doi:10.1109/GreenCom-iThings-CPSCom.2013.95.

[14] D. Cook, K. Feuz, N. Krishnan, Transfer Learning for Activity Recognition: A Survey, Knowledge and Information Systems 36 (3) (2013) 537—-556. doi:10.1007/s10115-013-0665-3.
URL http://eecs.wsu.edu/{~}cook/pubs/kais12.pdf

[15] O. Brdiczka, J. L. Crowley, P. Reignier, Learning situation models in a smart home, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 39 (1) (2009) 56–63. doi:10.1109/TSMCB.2008.923526.

[16] C. Ye, Y. Xia, Y. Sun, S. Wang, H. Yan, R. Mehmood, ERAR: An Event-Driven Approach for Real-Time Activity Recognition, 2015 International Conference on Identification, Information, and Knowledge in the Internet of Things (IIKI) (2015) 288–293doi:10.1109/IIKI.2015.69.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7428373

[17] M. Cornacchia, Y. Zheng, A Survey on Activity Detection and Classification Using Wearable Sensors, IEEE SENSORS JOURNAL 17 (2) (2017) 386–403.

[18] G. Okeyo, L. Chen, H. Wang, R. Sterritt, Dynamic sensor data segmentation for real time activity recognition, Pervasive and Mobile Computing 10, Part B (2014) 155–172.
URL http://www.tech.dmu.ac.uk/{~}limingchen/PMC-S-11-00304.pdfhttp://www.sciencedirect.com/science/article/pii/S1574119212001393

[19] G. Okeyo, L. Chen, H. Wang, R. Sterritt, Dynamic sensor data segmentation for real-time knowledge-driven activity recognition, Pervasive and Mobile Computing 10 (2014) 155–172. doi:10.1016/j.pmcj.2012.11.004.
URL http://linkinghub.elsevier.com/retrieve/pii/S1574119212001393

[20] J. Wan, M. J. O'Grady, G. M. P. O'Hare, Dynamic sensor event segmentation for real-time activity recognition in a smart home context, Personal and Ubiquitous Computing 19 (2) (2015) 287–301. doi:10.1007/s00779-014-0824-x.

[21] D. R. Faria, M. Vieira, C. Premebida, U. Nunes, Probabilistic Human Daily Activity Recognition towards Robot-assisted Living (2015) 582–587.

[22] G. Okeyo, L. Chen, H. Wang, R. Sterritt, A hybrid ontological and temporal approach for composite activity modelling, Proc. of the 11th IEEE

Int. Conference on Trust, Security and Privacy in Computing and Communications, TrustCom-2012 - 11th IEEE Int. Conference on Ubiquitous Computing and Communications, IUCC-2012 (2012) 1763–1770doi:10.1109/TrustCom.2012.34.

[23] G. Okeyo, L. Chen, H. Wang, Combining ontological and temporal formalisms for composite activity modelling and recognition in smart homes, Future Generation Computer Systems 39 (2014) 29–43. doi:10.1016/j.future.2014.02.014.
URL http://dx.doi.org/10.1016/j.future.2014.02.014

[24] J. Rafferty, C. D. Nugent, J. Liu, L. Chen, From Activity Recognition to Intention Recognition for Assisted Living Within Smart Homes, IEEE Transactions on Human-Machine Systems PP (99) (2016) 1–12. doi:10.1109/THMS.2016.2641388.
URL http://ieeexplore.ieee.org/document/7807210/

[25] G. Meditskos, S. Dasiopoulou, I. Kompatsiaris, MetaQ: A knowledge-driven framework for context-aware activity recognition combining SPARQL and OWL 2 activity patterns, Pervasive and Mobile Computingdoi:10.1016/j.pmcj.2015.01.007.
URL http://www.sciencedirect.com/science/article/pii/S1574119215000243{%}5Cnhttp://linkinghub.elsevier.com/retrieve/pii/S1574119215000243

[26] R. Helaoui, D. Riboni, M. Niepert, C. Bettini, H. Stuckenschmidt, Towards activity recognition using probabilistic description logics, Activity Context Representation: Techniques and Languages 12 (2012) 5. doi:10.1145/2493432.2493501.
URL http://www.aaai.org/ocs/index.php/WS/AAAIW12/paper/viewPDFInterstitial/5269/5552

[27] U. Naeem, Activities of daily life recognition using process representation modelling to support intention analysis, International Journal of Perva-

sive Computing and Communications 11 (3) (2015) 347. doi:10.1108/IJPCC-01-2015-0002.

[28] E. D. Valle, M. Grossniklaus, C-SPARQL: A CONTINUOUS QUERY LANGUAGE FOR RDF DATA STREAMS, International Journal of Semantic Computing 04 (01) (2010) 3–25.
URL http://dx.doi.org/10.1142/S1793351X10000936

[29] K. L. Skillen, L. Chen, C. D. Nugent, M. P. Donnelly, W. Burns, I. Solheim, Ontological user modelling and semantic rule-based reasoning for personalisation of Help-On-Demand services in pervasive environments, Future Generation Computer Systems 34 (2014) 97–109. doi:10.1016/j.future.2013.10.027.

[30] R. Culmone, P. Giuliodori, M. Quadrini, Human Activity Recognition using a Semantic Ontology-Based Framework, International Journal on Advances in Intelligent Systems 8 (2) (2015) 159–168.
URL http://www.iariajournals.org/intelligent{_}systems/www.iaria.org

[31] X. Hong, C. D. Nugent, Segmenting sensor data for activity monitoring in smart environments, Personal and Ubiquitous Computing 17 (3) (2013) 545–559. doi:10.1007/s00779-012-0507-4.

[32] S. Abburu, A Survey on Ontology Reasoners and Comparison, International Journal of Computer Applications 57 (17) (2012) 33–39. doi:10.5120/9208-3748.

[33] K. Dentler, R. Cornet, A. Ten Teije, N. De Keizer, Comparison of reasoners for large ontologies in the OWL 2 EL profile, Semantic Web 2 (2) (2011) 71–87. doi:10.3233/SW-2011-0034.

[34] G. D. Giacomo, M. Lenzerini, TBox and ABox Reasoning in Expressive Description Logics, Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR'96) (1996)

316–327doi:10.1.1.22.8293.
URL http://www.aaai.org/Papers/Workshops/1996/WS-96-05/WS96-05-004.pdf

[35] D. Triboan, L. Chen, F. Chen, Z. Wang, Semantic segmentation of real-time sensor data stream for complex activity recognition, Personal and Ubiquitous Computing (2017) 1–15doi:10.1007/s00779-017-1005-5.
URL http://link.springer.com/10.1007/s00779-017-1005-5

[36] D. Triboan, L. Chen, F. Chen, S. Fallmann, I. Psychoula, Real-Time Sensor Observation Segmentation For Complex Activity Recognition Within Smart Environments, in: 2017 IEEE 14th International Conference on Ubiquitous Intelligence and Computing (UIC 2017), San Francisco, 2017.

[37] D. Triboan, L. Chen, F. Chen, Z. Wang, Towards a Service-Oriented Architecture for a Mobile Assistive System with Real-time Environmental Sensing, TSINGHUA SCIENCE AND TECHNOLOGY 21 (6) (2016) 581–597.

[38] D. Triboan, L. Chen, F. Chen, Towards a Mobile Assistive System Using Service- oriented Architecture, in: 2016 IEEE Symposium on Service-Oriented System Engineering Towards, IEEE, Oxford, 2016, pp. 187–196. doi:10.1109/SOSE.2016.41.
URL http://doi.ieeecomputersociety.org/10.1109/SOSE.2016.41

[39] D. Riboni, C. Bettini, OWL 2 modeling and reasoning with complex human activities, Pervasive and Mobile Computing 7 (3) (2011) 379–395. doi:10.1016/j.pmcj.2011.02.001.
URL http://dx.doi.org/10.1016/j.pmcj.2011.02.001

[40] R. Volz, S. Staab, B. Motik, Incremental Maintenance Of Materialized Ontologies, Lecture Notes in Computer Science 2888/2003 (2888/2003) (2003) 707–724.

[41] W3C, SPIN - Overview and Motivation (2011).
URL https://www.w3.org/Submission/spin-overview/

[42] B. Cuenca Grau, C. Halaschek-Wiener, Y. Kazakov, History matters: Incremental ontology reasoning using modules, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 4825 LNCS (2007) 183–196. doi:10.1007/978-3-540-76298-0_14.

[43] Y. Ren, J. Z. Pan, I. Guclu, M. Kollingbaum, A combined approach to incremental reasoning for EL ontologies, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 9898 LNCS (August) (2016) 167–183. doi:10.1007/978-3-319-45276-0_13.

[44] M. Peters, C. Brink, S. Sachweh, A. Zündorf, Scaling parallel rule-based reasoning, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 8465 LNCS (2014) 270–285. doi:10.1007/978-3-319-07443-6_19.