Recognition of Concurrent and Interleaved Activities in a Smart Environment using Finite Automata

H. Karamath Ali
Research Scholar in Computer Science
Jamal Mohamed College (Autonomous)
Tiruchirappalli, India

D. I. George Amalarethinam, PhD
Associate Professor & Director-MCA
Jamal Mohamed College (Autonomous)
Tiruchirappalli, India

ABSTRACT
People have to do a number of activities in their day-to-day life. Though there are some activities that can be done only sequentially, most of the day-to-day activities do not impose such restriction. People very often tend to multitask with such activities. They interleave activities or go about them sequentially or concurrently. So, to be useful in real life situations, an activity recognition system must be able to recognize activities irrespective of how the user performs the activities. This paper proposes a novel simple approach that can be used to recognize sequential, interleaved and concurrent activities efficiently. The proposed method is tested with a publicly available dataset and is producing very promising results.

General Terms
Ubiquitous computing, mutiple goals, smart environments, proactivity.

Keywords
Pervasive computing, context awareness, activity recognition, concurrency, interleaving, finite automata.

1. INTRODUCTION
Proactivity is vital to offering context aware services in pervasive computing environments. To be context aware, among other things, a system must be able to recognize the user activities without any hindrance to the user[1]. The system should in no way impose any restrictions on the user. Users should be able to go about their normal routine of activities without having to make any adjustments for the sake of staying in a smart environment. Being allowed to freely interleave, combine or switch over activities will be very important for the users. So an activity recognition system must be able to recognize activities even if they are jumbled by the user. Users normally tend to do multiple activities either to get their jobs or goals quickly done or to avoid having to wait between activities. In this paper the terms 'activity' and 'goal' are used interchangeably. The constituent steps of an activity or goal are referred to as 'actions' or 'events'. Two ways of pursuing multiple goals are concurrency and interleaving[2][3]. Concurrent goals have some common actions. By doing the actions, the user accomplishes a certain portion of each goal. Then the remaining actions of the goals may be carried out in a sequential manner. In the case of interleaving, the user performs some actions of one goal, pauses it, performs some actions of another goal, pauses that goal and resumes the previous goal and so on. The non-common actions of concurrent goals also may be finished in interleaved manner. The various combinations of concurrency and interleaving of actions are illustrated in fig. 1, that has been redrawn from[3]. In this paper a simple and novel method that uses an automatically constructed finite automaton and a stack for recognizing concurrent and interleaved goals is presented. When tested with a publicly available data set, the method produces highly promising recognition rate for all types of goal compositions shown in fig. 1.

The remainder of this paper is organized as follows: Section 2 presents an overview of the related work in multiple activity recognition; section 3 defines the problem statement; section 4 explains the proposed method; section 5 discusses the dataset used and the experiment conducted; and section 6 presents conclusion.

2. RELATED WORK
Xiaoyong Chai and Qiang Yang[2], proposed a two level approach for recognizing multiple goals. The lower level determined the constituent actions of goals by measuring signal strength through a hand held device and using dynamic Bayesian network. The higher level used a model set in which models were instantiated and terminated dynamically. Each model was a finite state machine and functioned as a goal recognizer. Multiple-goal behavior was modeled as transitions among some pre-defined states of these models. By distinguishing the state of a model, it was inferred whether one of a user's goals was present or not.

A two-level probabilistic framework – CIGAR (Concurrent and Interleaving Goal and Activity Recognition) - to recognize both concurrent and interleaving goals was proposed by Derek Hao Hu and Qiang Yang[3]. Skip-chain conditional random fields (SCCRF) were used for modeling interleaving goals and concurrent goals were modeled by adjusting inferred probabilities through a correlation graph.

A Genetic Algorithm based method for Interleaved Sequential pattern detection(GAIS) from event sequences was suggested by Marja Ruotsalainen, et al.[4]. GAIS assumed the existence of models to detect the required kind of patterns from the event sequences. After generating an initial population of randomly created individuals GAIS calculated fitness value for each individual using models in the model set. Based on the fitness value, individuals were selected, crossed and mutated.

Using Factorial Conditional Random Fields (FCRFs) Tsu-yu Wu, et al.[5] designed experiments for recognition of multiple concurrent activities in the MIT House_n data set, which contains annotated data collected from multiple sensors in a real living environment.

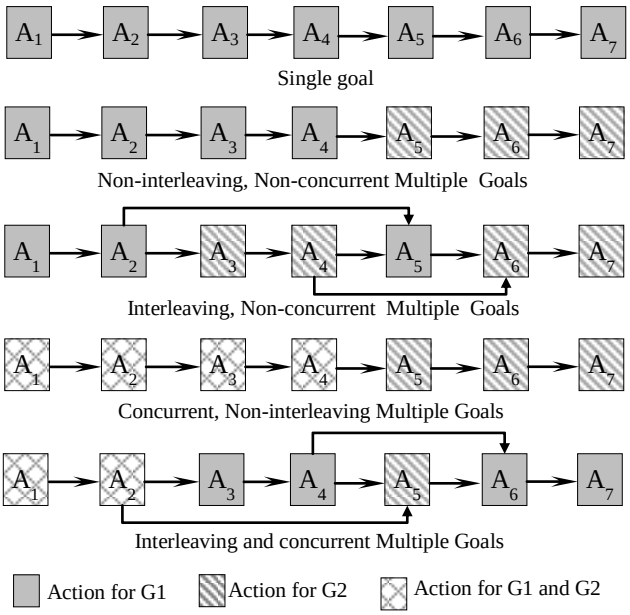

Figure 1. Concurrent and Interleaved Goals

Chia-chun Lian, et al.[6] also used FCRFs to model the conversational dynamics of concurrent chatting behaviors to accommodate co-temporal relationships among multiple activity states. They observed that the Loopy Belief Propagation (LBP) algorithm is inefficient, and proposed Iterative Classification Algorithm (ICA) as the inference method for FCRFs.

Interleaved Hidden Markov Models were used by Joseph Modayil, et al.[7] for recognizing multitasked activities. The model captured the dynamics of both inter and intra activities. Geetika Singla and Diane J. Cook[8] demonstrated that interleaved activities can be recognized by sensors in physical environments. According to them, HMM performed better than Naïve Bayes model.

Niels Landwehr[9] demonstrated that an inference algorithm obtained by extending structured approximate inference methods used with factorial hidden Markov models performs better than a standard hidden Markov model in recognizing multiple interleaved activities observed by a stream of sensor outputs.

Derek Hao Hu, et al.[10] defined a goal taxonomy that contained several classes of complexity levels and different granularities of activities, and related the recognition accuracy with different complexity levels or granularities. They used skip chain CRF for recognizing multiple concurrent and interleaving activities.

Eunju Kim, et al.[11] proposed a method named activity pattern discovery for activity recognition by building a hierarchical activity model. The lower-level activities, such as sitting, standing, eating, and driving, were recognized using a supervised learning algorithm. The higher level of the model discovered combinations of the lower-level activities that represent more complex activity patterns.

Rim Helaoui, et al.[12] showed how Markov logic can be combined with common-sense background knowledge to develop a framework for recognizing interleaved and concurrent activities.

Jianxia Chen, et al.[13] presented a logic-based approach using a heuristic search planner to solve the multigoal recognition problem efficiently, without the need of plan libraries. They first proposed the formulation of a multigoal recognition problem based on automated planning. Then a two level probabilistic plan recognition approach was used to recognize concurrent and interleaving goals from observed activity sequences.

The models used in the above mentioned works are either computationally very expensive[11], or require manual construction of the domain model initially. The need to devise methods for automatic construction of hierarchy of activities is observed by Derek Hao Hu, et al.[10]. In view of these observations, in this work a simple and novel method for recognizing concurrent and interleaved activities is proposed. In the proposed method a finite automaton is constructed automatically, as illustrated in [14], to hierarchically represent the different action sequences possible for performing each goal. Then interleaved and concurrent pursuance of goals are identified by simply traversing the paths in this automaton and retracing or backtracking whenever it is necessary to reassign the actions to different goals. The proposed method does not require any calculation in the recognition phase. The traversing and retracing are accomplished by simple push and pop operations in a stack. The method is tested using a publicly available dataset and is found to give highly promising recognition rate.

3. THE PROBLEM

Given are a set of goals $\mathbb{G} = \{G_1, G_2, G_3, \dots, G_n\}$ and a set of actions $\mathbb{A} = \{A_1, A_2, A_3, \dots, A_m\}$. Each goal is associated with a fixed number of sequences of actions in $\mathbb{A}$. An action in $\mathbb{A}$ may appear in more than one action sequence in any of the goals. A sequence of actions associated with a goal represents the actions that need to be taken to achieve the goal. Hence each goal may be achieved by one or more different sequences of actions. The exact number of action sequences possible for a goal depends upon the application and environment. As has been said in the introduction, a user may pursue multiple goals either in a concurrent or interleaved manner. As the user performs the constituent actions of the currently pursued goals, the system has to identify the goals and the corresponding sequence of actions.

4. THE METHOD

As mentioned in the problem, each goal may be achieved by one or more sequences of actions in $\mathbb{A}$. A set $\mathbb{S}$ is formed by collecting each possible action sequence of every goal in $\mathbb{G}$. That is

$$\mathbb{S} = \{ A^i_{j1} A^i_{j2} A^i_{j3} \dots A^i_{jk_j}, 1 \le i \le n, \\ 1 \le j \le p_i \; and \; each \; A^i_{jk} \in \mathbb{A}\}$$

where n is the number of possible goals, p_i is the number of possible action sequences of the i^{th} goal and k_j is the number of actions in the j^{th} sequence.

A DFA equivalent to $\mathbb{S}$ is constructed using the SL-infer algorithm [15] as explained in [14]. The SL-infer algorithm is extended to store in each state of the DFA, a list of labels that represent the goals corresponding to the action sequences that will lead to the state in a path from the starting state to any one of the final states. So given the current state, it can be easily decided which of the goals share the action sequence that led to the state. Obviously, in each of the final states only

one label will get stored. This will indicate which goal is accomplished when the final state is reached.

To scan the action sequence generated by the user, a stack and an input buffer are used. The stack will hold, at any point of time, the states traversed so far in the order of traversal, and the action inputs scanned. Each action input in the stack will be preceded and followed by the previous and current states. The input buffer will consist of the action inputs that are yet to be scanned. Initially, the stack will consist only of the starting state of the automaton. The top most element in the stack will always be the current state of the DFA. Depending upon the current state γ, and the action input a, which is the leftmost element of the input buffer, the automaton performs any one of the three actions Push, Pop and Print. 'δ' denotes the transition function of the DFA. The three actions are explained below:

i) Push : if $\delta(\gamma, a) \neq \emptyset$, then a is pushed into the stack followed by $\delta(\gamma, a)$.

ii) Pop : if $\delta(\gamma, a) = \emptyset$, then the topmost two elements in the stack (the current state and the input that led to the state) are removed from the stack.

iii) Print : if γ is a final state, then the action inputs in the stack, in the order from bottom of the stack to its top, are printed out along with the goal label available in γ. This means that the user has accomplished the goal by completing its corresponding sequence of actions.

5. DATA AND EXPERIMENT

To test the method explained above, the data set generated by Xiaoyong Chai and Qiang Yang[2] is used. The data set was generated in an office area by observing eight special goals of a Professor's activity. The office building has three entrances and 7 hallways. The hallways are named HW1 to HW7. When the user is in HW1 action a1 is generated, when in HW2 action a2 is generated and so on. There is an office room and two other rooms numbered 1 and 2. In each of the rooms either of the two actions 'printing' and 'attending seminar' can take place. Printing in rooms 1 and 2 generate actions a8 and a10 respectively. Attending seminars in rooms 1 and 2 generate actions a9 and a11 respectively.

The eight goals that are observed are: seminar_print(1), seminar_seminar(2), restarea_print(3), restarea_seminar(4), office(5), upper-exit(6), left-exit(7) and lower-exit(8). Given within parentheses are the labels used to represent the goals in the dataset. There are eleven actions numbered from 1 to 11 as mentioned above. The actions are determined by measuring the strength of signals broadcast by three access points. Each action gives information about the location of the user. The action sequences for each of the goals are given. Each goal is achievable using more than one action sequence. For example, the goal 'lower-exit' is achievable by three action sequences : (a1→a3→a6→a7), (a2→a3→a6→a7) and (a5→a4→ a6→a7). In this work it is assumed that the actions performed by the user are determined from signal strength measurements by some recognizer and are readily available for the recognition of interleaved and concurrent multiple goals.

As explained in section 4, a set $\mathbb{S}$ is formed by collecting all action sequences for all the goals, and a DFA is constructed. The transition table of the constructed DFA is given in table 1. There are 19 states in the DFA numbered s0 to s18. s0 is the initial state. Each column corresponds to an action. The first column corresponds to action1, the second to action2 and so on. $\delta(x, y) =$ - means there is no transition from state x on action y. Final states are shown by underlining.

Table 1. Transition Table for the DFA

	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11
s0	s1	s12	-	-	s14	-	s17	-	-	-	-
s1	s1	s2	s3	-	-	-	-	-	-	-	-
s2	-	-	-	-	-	-	-	-	-	-	-
s3	-	-	s3	s4	-	s8	-	-	-	-	-
s4	-	-	-	s4	s5	-	-	s6	s7	-	-
s5	-	-	-	-	-	-	-	-	-	-	-
s6	-	-	-	-	-	-	-	-	-	-	-
s7	-	-	-	-	-	-	-	-	-	-	-
s8	-	-	-	-	-	s8	s9	-	-	s10	s11
s9	-	-	-	-	-	-	-	-	-	-	-
s10	-	-	-	-	-	-	-	-	-	-	-
s11	-	-	-	-	-	-	-	-	-	-	-
s12	s13	s12	s3	-	-	-	-	-	-	-	-
s13	-	-	-	-	-	-	-	-	-	-	-
s14	-	-	-	s15	s14	-	-	s6	s7	-	-
s15	-	-	s16	s15	-	s8	-	-	-	-	-
s16	s13	s2	s16	-	-	-	-	-	-	-	-
s17	-	-	-	-	-	s18	s17	-	-	s10	s11
s18	-	-	s16	s4	-	s18	-	-	-	-	-

The goal labels stored in each state by the algorithm are given below.

s0 : 1 2 3 4 5 6 7 8
s1 : 1 2 3 4 6 7 8
s2 : 6
s3 : 1 2 3 4 7 8
s4 : 1 2 7
s5 : 7
s6 : 1
s7 : 2
s8 : 3 4 8
s9 : 8
s10 : 3
s11 : 4

s12 : 1 2 3 4 5 7 8
s13 : 5
s14 : 1 2 3 4 5 6 8
s15 : 3 4 5 6 8
s16 : 5 6
s17 : 1 2 3 4 5 6 7
s18 : 1 2 5 6 7

How multiple goals are identified using the DFA is illustrated below with a sample multiple goal action sequence : (a1→a3→a4→a8→a5)

Stack		Input
s0	Push	a1 a3 a4 a8 a5
s0 a1 s1	Push	a3 a4 a8 a5
s0 a1 s1 a3 s3	Push	a4 a8 a5
s0 a1 s1 a3 s3 a4 s4	Push	a8 a5
s0 a1 s1 a3 s3 a4 s4 a8 s6	Print	a5
s0 a1 s1 a3 s3 a4 s4 a8 s6	Pop	a5
s0 a1 s1 a3 s3 a4 s4	Push	a5
s0 a1 s1 a3 s3 a4 s4 a5 s5	Print	- -

As explained in section 4, Print is invoked when the DFA reaches a final state, and the action inputs in the stack are printed out. So, when Print is invoked for the first time, action sequence (a1 a3 a4 a8) is printed out. This is a sequence to achieve goal 1. At the second invocation of Print the sequence (a1 a3 a4 a5) is printed out. This is a sequence for goal 7. Thus by a simple retracing using the stack, the two goals in the concurrent multiple goal sequence are identified.

When trying to identify goals in real life situations, two problems that normally arise are : repetition of an action input and irrelevant action input. Repetition of an input may be handled by allowing the DFA to remain in the current state until some other valid input occurs or till the lapse of a predefined duration of time after which appropriate remedial action may be initiated. The occurrence of irrelevant inputs can be very easily identified by simply checking the entries in the row of the current state in the transition table. Inputs corresponding to the empty entries in the row are irrelevant and can be simply ignored.

For recognition of interleaved multiple goals, that is, goals that do not have common actions, a setup similar to the one shown above is used. But instead of a single stack, there will be one stack for each new goal identified. Whenever a new goal is likely to begin, a new stack is created and the likely inputs and the corresponding states are pushed into that stack. When it is not possible to proceed further on an input, backtracking is done to realign the inputs to the active goals. In addition to the stacks and an input buffer, an array, named 'Trace', is used to keep track of which input is assigned to which goal. This is illustrated in the following example. The underlined number is the next input to be considered.

Stack1		Input
s0	Push	a<u>1</u> a7 a3 a6 a4 a3 a2 a9
s0 a1 s1		a1 <u>a7</u> a3 a6 a4 a3 a2 a9

Trace								
	1							

The content of the array Trace indicates that the leftmost input is assigned to first goal. The label of the actual goal it belongs to, cannot be decided now. Since δ(s1, a7) is undefined, Stack1 becomes inactive; since δ(s0, a7) is defined a new stack is created and the configuration becomes as follows :

Stack1	Input
s0 a1 s1	a1 a7 <u>a3</u> a6 a4 a3 a2 a9

Stack2
s0 a7 s17

Trace								
	1	2						

Now, δ(s17, a3) is undefined. So, stack2 becomes inactive and control switches to the other stack. Proceeding in this way, the following configuration is reached, where it is not possible to proceed further.

Stack1	Input
s0 a1 s1 a3 s3 a6 s8	a1 a7 a3 a6 <u>a4</u> a3 a2 a9

Stack2
s0 a7 s17

Trace								
	1	2	1	1				

Since both δ(s8, a4) and δ(s17, a4) are undefined, it is not possible to proceed further. It does not mark the beginning of a new goal because δ(s0, a4) also is undefined. This situation warrants backtracking. So, the last assignment of an input to a goal is undone, and the configuration becomes as follows.

Stack1	Input
s0 a1 s1 a3 s3	a1 a7 a3 <u>a6</u> a4 a3 a2 a9

Stack2
s0 a7 s17

Trace								
	1	2	1					

After backtracking, the alternative assignment is tried as shown below.

Stack1	Input
s0 a1 s1 a3 s3	a1 a7 a3 a6 <u>a4</u> a3 a2 a9

Stack2
s0 a7 s17 a6 s18

Trace								
	1	2	1	2				

Proceeding in this way, the final configuration becomes as

Stack1	Input
s0 a1 s1 a3 s3 a4 s4 a9 s7	a1 a7 a3 a6 a4 a3 a2 a9 -

Stack2
s0 a7 s17 a6 s18 a3 s16 a2 s2

Trace								
	1	2	1	2	1	2	2	1

The states s7 and s2 are final states. Also, the goal labels in s7 and s2 are 2 and 6 respectively. So it is decided that the sequence (a1 a3 a4 a9) belongs to goal 2 and (a7 a6 a3 a2) belongs to goal 6. The contents of Trace represent which of the inputs belong to the first goal and which to the second one.

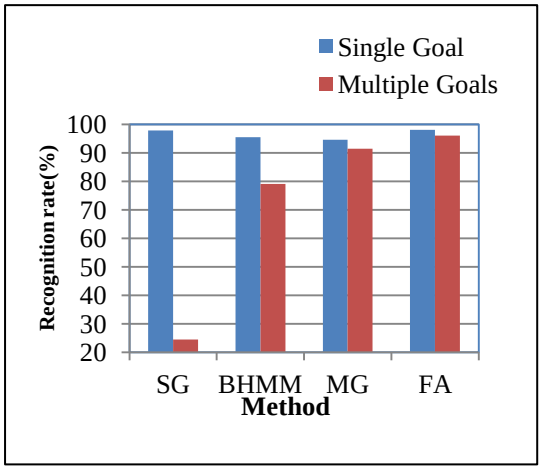

Figure 2. Performance Comparison

It is obvious that in this method, no calculations are necessary for recognizing the goals from action sequences. Simple push and pop operations are all that the method requires for goals recognition. The recognition rate of the presented method is 98% and 96% for single and multiple goals respectively. Figure 2 illustrates this with recognition rates achieved by other methods as presented in [2]. Performance of the method presented in this paper is denoted by FA in the fig. 2.

The method has a few limitations. The data set is collected in an office area which has a well-defined structure. So it can be said that the method explained in this paper is suitable for such structured areas as big shopping malls, office complexes, hospitals, etc. Users in such areas may be provided with navigation guidelines and help using this method. For deciding suitability for other environments the method has to be tested with appropriate data sets. On some irrelevant action input, the system may have to backtrack, in the worst case, undoing all the assignments. This may result in an increase in the execution time.

6. CONCLUSION

In this paper a novel and simple method for finding concurrent and interleaved goals of a user in a smart environment is illustrated. The method does not involve any calculation during the recognition of goals. A sequence of simple push and pop operations is all that is required for single and multiple goal recognition. When tested with a dataset the proposed method gives better results than the previous similar works. To further test the suitability of the method for real-time environments other publicly available data sets need to be tried with the method.

7. REFERENCES

[1] M. Satyanarayanan, "Pervasive Computing: Vision and Challenges", IEEE Personal Communications, August 2001, pp. 10-17.

[2] Xiaoyong Chai and Qiang Yang, "Multiple-Goal Recognition from Low-Level Signals", American Association for Artificial Intelligence, 2005, pp. 3-8.

[3] Derek Hao Hu and Qiang Yang, "CIGAR: Concurrent and Interleaving Goal and Activity Recognition", Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, 2008, pp. 1363-1368.

[4] Marja Ruotsalainen, Timo Ala-Kleemola and Ari Visa, "GAIS: A Method for Detecting Interleaved Sequential Patterns from Imperfect Data", Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2007), 2007, pp. 530-535.

[5] Tsu-yu Wu and Chia-chun Lian and Jane Yung-jen Hsu, "Joint Recognition of Multiple Concurrent Activities using Factorial Conditional Random Fields", Proceedings of the 22nd Conference on Artificial Intelligence, (AAAI), 2007.

[6] Chia-chun Lian and Jane Yung-jen Hsu, "Probabilistic Models for Concurrent Chatting Activity Recognition", ACM Transactions on Intelligent Systems and Technology (TIST), Volume 2 Issue 1, January 2011,Article No. 4, pp. 1138-1143.

[7] Joseph Modayil, Tongxin Bai and Henry Kautz, "Improving the Recognition of Interleaved Activities", UbiComp'08, September 21-24, 2008, pp. 40-43.

[8] Geetika Singla and Diane J. Cook, "Interleaved Activity Recognition for Smart Home residents", Proceedings of the 5th International Conference on Intelligent Environments, 2009.

[9] Niels Landwehr, "Modeling Interleaved Hidden Processes", Proceedings of the 25th International Conference on Machine Learning, 2008.

[10] Derek Hao Hu, Sinno Jialin Pan, VincentWenchen Zheng, Nathan Nan Liu and Qiang Yang, "Real World Activity Recognition with Multiple Goals", UbiComp'08, September 21-24, 2008.

[11] Eunju Kim, Sumi Helal and Diane Cook, "Human Activity Recognition and Pattern Discovery", IEEE Pervasive Computing, Volume 9, issue 1, 2010, pp. 48-53.

[12] Rim Helaoui, Mathias Niepert and Heiner Stuckenschmidt, "Recognizing Interleaved and Concurrent Activities: A Statistical-Relational Approach", Proceedings of IEEE International Conference on Pervasive Computing and Communications, 2011, pp. 1-9.

[13] Jianxia Chen, Yixin Chen, You Xu, Ruoyun Huang, Zheng Chen, "A Planning Approach to the Recognition of Multiple Goals", International Journal Of Intelligent Systems, vol. 28, 2013, pp. 203–216.

[14] H. Karamath Ali and D. I. George Amalarethinam, "Automatic Construction of Finite Automata for Recognizing User Activities in Smart Environments", IOSR Journal of Engineering, Volume 3, Issue 12, December 2013, pp. 40-45.

[15] Henning Fernau, "Algorithms for learning regular expressions from positive data", Journal of Information and Computation, Volume 207, Issue 4, April 2009, pp. 521-541.