Simulation Modelling Practice and Theory 19 (2011) 161–180

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

Linking temporal first-order logic with Bayesian networks for the simulation of pervasive computing systems

Eleftheria Katsiri [a,*], Alan Mycroft [b]

[a] Department of Computer Science and Information Systems, Birkbeck, University of London, Malet Street, London SW1E 7HX, UK
[b] Computer Laboratory, University of Cambridge, William Gates Building, 15 JJ Thompson Avenue, Cambridge CB3 0FD, UK

ARTICLE INFO

Article history:
Received 21 March 2010
Received in revised form 29 May 2010
Accepted 1 June 2010
Available online 8 June 2010

Keywords:
Ubiquitous computing
Pervasive computing
Mobile users
Location technology
Bayesian networks
Naïve Bayes classifier
Knowledge-base representation and reasoning
Temporal first-order logic

ABSTRACT

The authors' previous work discussed a scalable abstract knowledge representation and reasoning scheme for Pervasive Computing Systems, where both low-level and abstract knowledge is maintained in the form of temporal first-order logic (TFOL) predicates. Furthermore, we introduced a novel concept of a generalised event, an abstract event, which we define as a change in the truth value of an abstract TFOL predicate. Abstract events represent real-time knowledge about the system and they are defined with the help of well-formed TFOL expressions whose leaf nodes are concrete, low-level events using our AESL language.

In this paper, we propose to simulate pervasive systems by providing estimated knowledge about its entities and situations that involve them. To achieve this goal, we enhance AESL with higher-order function predicates that denote approximate knowledge about the likelihood of a predicate instance having the value True with respect to a time reference. We define a mapping function between a TFOL predicate and a Bayesian network that calculates likelihood estimates for that predicate as well as a confidence level, i.e., a metric of how reliable the likelihood estimation is for that predicate.

Higher-order likelihood predicates are implemented by a novel middleware component, the Likelihood Estimation Service (LES). LES implements the above mapping; first, for each abstract predicate, it *learns* a Bayesian network that corresponds to that predicate from the knowledge stored in the sensor-driven system. Once trained and validated, the Bayesian networks generate a *likelihood estimate* and a *confidence level*. This new knowledge is maintained in the middleware as *approximate knowledge* therefore providing a simulation of the pervasive system, in the absence of real-time data. Last but not least, we describe an experimental evaluation of our system using the Active BAT location system.

1. Introduction

Pervasive (a.k.a. Ubiquitous) computing is a post-desktop model of human–computer interaction in which information processing has been thoroughly integrated into everyday objects and activities. Marc Weiser that coined the term pervasive (ubiquitous) computing in 1993 states: we are trying to conceive a new way of thinking about computers in the world, one that takes into account the natural human environment and allows the computers themselves to vanish into the background. In the course of ordinary activities a natural interface means that someone using pervasive computing engages many computational devices and systems simultaneously, and may not necessarily even be aware that they are doing so.

Sentient computing [1] is a form of pervasive computing that advocates that applications may become more useful to the user if they are aware of their environment. Awareness comes through an infrastructure of sensors that are deployed on the

* Corresponding author. Tel.: +44 7771748048.
E-mail addresses: eli@dcs.bbk.ac.uk, ekatsiri@gmail.com (E. Katsiri), am21@cl.cam.ac.uk (A. Mycroft).

1569-190X/$ - see front matter
doi:10.1016/j.simpat.2010.06.002

physical space. Both mobile users and immobile objects are equipped with wearable transceivers. These are able to communicate with the sensor infrastructure so that the object's position is calculated continually. This information is used to construct a world model which allows location-aware or context-aware applications to be constructed. One famous research prototype of a sentient computing system was the work at AT&T Laboratories, Cambridge. It consisted of an ultrasonic indoor location system called the Active Bat [2] which provided a location accuracy of about 3 cm. The world model was managed via the SPIRIT database, using the CORBA middleware [3] to access information and spatial indexing to deliver high-level events such as *Alice has entered the kitchen* to listening context-aware applications. The research continues at the Digital Technology Group at the University of Cambridge.

Sentient computing presents challenges across computer science:

1. *Programmability and scalability*. First, as Weiser advocated, a user-centred approach is desirable; the computing infrastructure needs to serve the user not the other way around. This leads to the need of ease of programmability of pervasive systems; the targeted user is a layman not a skilled programmer and he/she should be able to program their own ubiquitous applications in a safe and efficient manner. The user should be able to express abstract, high-level concepts, close to human intuition. Knowledge on such concepts may not be available directly from the sensors and will need to be inferred by reliable mechanisms. Furthermore, the system needs to respond to the user request in near real time. As calculating a response involves correlating a potentially very large number of events, a scalable reasoning mechanism is necessary.
2. *Correctness and robustness*. Second, the system needs to be robust against failure of the sensor infrastructure that it relies on. What happens if the sensors fail? Can we still reason about contextual situations if no real-time data is available? How accurate is our reasoning then? Can we still provide enough feedback to the user so that they can make informed decisions? Can we store historical information about the environment and use it to detect trends in the data and predict user behaviour? Is this information useful for improving the Sentient system itself?

1.1. Programmable, scalable reasoning for pervasive computing applications

The development of even a conceptually simple sentient application is nontrivial because it involves the cooperation of several distributed elements, such as sensors, databases, or effectors. Imagine we want to automatically initiate the playback of a user's favourite music whenever he starts working. For that, the application would register with the world model (to receive that user's presence events) and with a keyboard activity monitor event source (to be notified when the keyboard activity of the user's computer changes significantly). When a user is both in his office and typing regularly, then the application would initiate the music playback.

On the other hand, significant work on middleware has been done in the area of distributed systems. There, the event-based paradigm is used to decouple communication between distributed objects, and enables the interchange of events between them in an asynchronous form through Notification Channels. Effort has also been directed towards defining more meaningful *composite events* [4], which are typically recognised by *finite-state machines*.

In previous work [5] the authors merged the two above paradigms in order to provide high-level tools for programming pervasive applications. More specifically, they defined a scalable knowledge-base system for pervasive computing. The authors' approach uses deductive systems in an unusual way, namely, in order to allow applications to register inference rules that generate abstract knowledge from low-level, sensor-derived knowledge. Scalability is achieved by maintaining a dual-layer knowledge representation mechanism for reasoning about the Sentient Environment that functions in a similar way to a two-level cache. Furthermore, we introduced a novel concept of a generalised event, an abstract event [6,7], which we define as a notification of transparent changes in distributed state. An extension to the publish/subscribe protocol is discussed, in which a higher-order service (*Abstract Event Detection Service*) publishes its interface; this service takes a TFOL abstract event definition in our *Abstract Event Specification Language (AESL)* as an argument and in return publishes an interface to a further service (an Abstract Event Detector) which notifies transitions between the values True and False of the formula, thus providing a more natural and efficient interface to applications. Abstract Event detectors are structured as Rete Networks [8] and form a deductive knowledge-base. In addition to Rete-based abstract event detectors, in previous work [9] we introduced the concept of detectors implemented as hidden Markov models (HMMs). We demonstrated that HMM-based detectors are appropriate for detecting invariant patterns such as user trajectories that correspond to sitting down, getting up, walking, being titted and being still.

AESL, enhanced as above, allows the programmer to design pervasive applications that have a number of desirable features. First, they are easy to program by linking an abstract event specification to a desirable action through a temporal first-order logic formula that is easy to understand. Second, they can reason with absence of information, which is not possible with event-based languages for distributed systems, that are based on finite-state machines [6]. As a result, AESL supports an increased level of expressiveness that is closer to human intuition.

1.1.1. Example

Using the definitions and nomenclature proposed in [5,6] we can define applications that link abstract state predicates to *action predicates*. In this paper, the former are always prefixed with *H_(High-level)*, *H_UserInLocation*(*uid, role, rid, rattr, t*).

For example, in the formula: