doi: 10.17234/INFUTURE.2015.21 Review

Modelling states of a computing system aware of an aspect of context

Krunoslav Peter
Andrija Stampar Teaching Institute of Public Health
Mirogojska 16, Zagreb, Croatia
kruno.peter@stampar.hr

Summary

The paper describes an approach to the modelling adaptive behaviour of a computing system aware of a single aspect of context. It senses states of an aspect of context and changes its state accordingly. Its adaptation to context can be modelled by using a finite-state automaton. It is possible to implement its decision-making subsystem by translating the state-transition table of the finite automaton to the set of decision rules of the subsystem.

Keywords: context, sensor, context-aware system, adaptable system, decision-making subsystem, finite-state automaton, state-transition table, decision rule

Introduction

Modern computing systems are "responsive to their contexts and environments" (Ornellas, 2014). For example, a system aware of light senses light conditions of its environment and adjusts its display's backlight for optimum visibility and lower power consumption. The system adapts to context automatically "without distracting from the user's task" (Chalmers, 2011, 3) and "reduces the burden of excessive user involvement" (Loke, 2007, 7). Any adaptation is a change of system's state. Such behaviour can be modelled by using a general computation model called the finite automaton or finite-state automaton. Additionally, the state-transition table of a finite automaton may be useful for implementation of the decision-making subsystem of a context-aware system. It can be translated to the set of decision rules of such subsystem.

The goal of this article is to present a simple and technology-dependent approach to the problem of implementing the thinking subsystem. Other approaches are mentioned in the article. They are more complex than rule-based approaches but effective in certain domains.

Context

Context of an adaptable system includes information such as network connectivity, communication costs, light conditions, location and time (Schilit et al, 1994). It is information about system's environment and system itself. Such a piece of information is called the *aspect of context* (Chalmers, 2011, 68).

"*Sensors*, biological or nonbiological, provide a means to acquire" such information (Loke, 2007, 15).
Some aspects of context are *discrete* (Loke, 2007, 20), for example, network connectivity. A system is connected or disconnected to a wireless network during a particular period of time. A connected system adapts to context by executing its firewall (it does not bother a user to execute "a strange program called Firewall"); a disconnected system will automatically stop and unload the firewall to free main memory. The states of network connectivity are "connected" and "disconnected"; they are also 1 (true) and 0 (false) – the Boolean values. Those states can be thought as elements of an *alphabet of states*. They change arbitrarily during the time. A sequence of those discrete states forms a *word* of some *regular language* (Abiteboul et al, 1995, 14), for example disconnected, connected and disconnected (010). It is a *formal language* that can be described by using a *regular expression* and accepted by a particular finite-state automaton. The previous language is specified by the regular expression *disconnected (connected disconnected)**.
Noise is an example of a *continuous* aspect of context (Loke, 2007, 20). A noise-aware system might set louder sound in a noisy environment. To implement such system, its developer might set the *thresholds* for sound pressure. For example, a noise-aware system senses "normal" and "loud" noise. Sound pressure in a quiet office is about 40 dB (Wikipedia, 2015); a system considers it "normal". Traffic on a busy street exceeds 80 dB (Wikipedia, 2015); it is sensed as "loud" noise. A noise-aware system recognizes a regular language of the "normal" state (< 80 dB) and the "loud" state (≥ 80 dB): *(normal + loud)**. They are the elements of an alphabet of noise states and they change arbitrarily.

Context-aware system

Besides main functionality, a *context-aware system* is able to utilize "contextual information about the physical world" (Loke, 2007, 5). It senses context and processes contextual information (Ferreira et al, 2015). Such system "reacts to changing context" (Schilit et al, 1994) and behaves adaptable; it does so automatically. Using of perceptual information about the environment distinguishes *context-aware computing* from traditional computing (Loke, 2007, 7; Chalmers, 2011, 67).
The principle of "separating the acquisition and representation of context from the use of context" is important for developing context-aware systems (Ferreira et al, 2015). Therefore, there are three additional functionalities of a context-aware system: *sensing*, *reasoning* and *acting* (Loke, 2007, 15). These functionalities can be realized in a centralized or a distributed architecture (Schilit et al, 1994; Loke, 2007, 15). According to these functionalities, an abstract architecture of context-aware system has three subsystems (Loke, 2007, 25): the *sensing subsystem*, the *thinking subsystem* and the *acting subsystem*.

The first subsystem acquires data about system's environment; the second subsystem performs reasoning about these data. Finally, the third subsystem performs an adaptation to the environment.
The thinking or decision-making subsystem "uses input from software and hardware sensors to decide how, when, and where to adapt the system" (McKinley et al, 2004). This subsystem might be a simple rule-based system or an arbitrary complex reasoning system (Loke, 2007, 23; McKinley et al, 2004). If the thinking subsystem is a rule-based system whose states changes according to the states of context, then it is possible to model and implement it like a finite-state automaton.
The paper presents two examples of general approaches to implementing software adaptation. Setting system's sound loudness in a noisy context is an example of *parameter adaptation*; it modifies system's parameters that determine adaptive behaviour. On the other hand, *compositional adaptation* exchanges system's components with others that improve system's adaptability to its context or add new behaviour (McKinley et al, 2004), e.g., to run the firewall of a connected system. Thinking subsystems for both software adaptations might be modelled by using finite automata.

Finite-state automata

As is previously stated, an appropriate model for a system aware of an aspect of context is the *finite-state automaton* (FSA). An FSA (Sipser, 2013, 35; Abiteboul et al, 1995, 13) is a 5-tuple $(S, \Sigma, \delta, s_0, F)$, where

S – a finite set of *states*;
Σ – an *alphabet*;
δ – the *state-transition function* $\delta: S \times \Sigma \rightarrow S$;
s_0 – the *start state*;
F – a set of *accepting states*.

The state-transition function is often represented as a *state-transition table* (Dovedan, 2003, 65).
The FSA M accepts the language $L(M)$; it is a *regular language*. It can be specified by writing an FSA accepting it (Abiteboul et al, 1995, 14) or by using the regular expression (Dovedan, 2003, 19).
"A regular expression over Σ is written using the symbols in Σ and the operations concatenation, * and +" (Abiteboul et al, 1995, 14). * stands for set of all words over Σ and + stands for union.

Specifying adaptive behaviour

Adaptive behaviour of a system aware of the aspect C of context can be specified like an FSA $(S, \Sigma, \delta, s_0, F)$:

S – a finite set of system's states;
Σ – a finite set of states of the aspect C of context;
δ – the state-transition function $\delta: S \times \Sigma \rightarrow S$;
s_0 – the initial state
F – a set of accepting states.

Here is an example of a simple context-aware system (Example 1):

Example 1
The system M_1 that is aware of noise conditions. It senses noise conditions every second; if noise is louder than a specified threshold, the system's sound will also be louder.
The system recognizes language $L(M_1)$ – an arbitrary sequence of conditions *normal* and *loud*[1]:

$(normal + loud)^*$

The system is defined as the FSA M_1 ($\{ volume_1, volume_2 \}, \{ normal, loud \}, \delta, \{ volume_1 \}, \{ volume_1, volume_2 \}$), with a state-transition table (Table 1)

Table 1: The state-transition table of the FSA M_1

δ	*normal*	*loud*
$volume_1$	$volume_1$	$volume_2$
$volume_2$	$volume_1$	$volume_2$

Example 2 shows a bit complicated system:

Example 2
A system M_2 is aware of noise conditions. It has three states. It is not allowed the change of system's state from the "normal" state to the loudest of all states. A user of system must not be shocked by the loudness change.
The system recognizes language L (M_2) – an arbitrary sequence of conditions *normal*, $loud_1$ and $loud_2$:

$(normal + loud_1 + loud_2)^*$

It is defined as the FSA M_2 ($\{ volume_1, volume_2, volume_3 \}, \{ normal, loud_1, loud_2 \}, \delta, \{ volume_1 \}, \{ volume_1, volume_2, volume_3 \}$), with a state-transition table (Table 2)

[1] One can use characters or numbers instead *normal* and *loud*.

Table 2: The state-transition table[2] of the FSA M_2

δ	***normal***	***loud₁***	***loud₂***
$volume_1$	$volume_1$	$volume_2$	$volume_2$
$volume_2$	$volume_1$	$volume_2$	$volume_3$
$volume_3$	$volume_2$	$volume_2$	$volume_3$

Implementing a system aware of single aspect of context

In the case of a continuous aspect of context, it is possible that the sensing subsystem delivers discrete values according to defined thresholds. For instance, if sound pressure exceeds 80 dB, the sensing subsystem delivers the "loud" value. The thinking subsystem may store this value in a *context variable*. Additionally, a value of the *state variable* represents system's state; it is an input of the acting subsystem. The state-transition table is implemented by the set of *if-then* rules (Schilit et al, 1994) of the thinking subsystem. The syntax for the decision rule ***if*** *(State and Context)* ***then*** *State* corresponds to the state-transition function $\delta: S \times \Sigma \rightarrow S$. Example 3 shows decisions rules for the thinking subsystem defined by the state-transition table of the FSA M_1.

Example 3
The state-transition table of the FSA M_1 might be directly translated to the following decision rules[3]:

Input:
state
context

if *(state == "volume₁" and context == "normal")* ***then*** *state = "volume₁".*
if *(state == "volume₁" and context == "loud")* ***then*** *state = "volume₂".*
if *(state == "volume₂" and context == "normal")* ***then*** *state = "volume₁".*
if *(state == "volume₂" and context == "loud")* ***then*** *state = "volume₂".*

Output:
state

Assigning a value to the state variable that is different from the previous value triggers the acting subsystem in this implementation of the noise-aware system.

In the system aware of many aspects of context, there could be the state-transition tables for any aspects of context.

[2] Shades of grey in the state-transition table visually emphasise distinct values.

[3] In the syntax of decision rules, == is the equality operator and = is the assignment operator.

Other approaches

Developers of the decision-making subsystems also apply "first-order logic-based formalisms to represent context and situations, and rules that map situations to required actions" (Loke, 2007, 23). There is an approach that is "inspired by biological processes, such as the human nervous system" (McKinley et al, 2004). Decision makers might even learn about and adapt to dynamical context and user behaviour (Schilit et al, 1994; McKinley et al, 2004).

Conclusion

Modelling adaptive behaviour of a general computing system aware of an aspect of context like a finite-state automaton helps a developer of a system to define context's and system's states. Adaptation to an aspect of context is specified in the form of the state-transition table of a finite automaton; this table might be translated to the set of decision rules of the thinking subsystem.

Literature

Abiteboul, Serge; Hull, Richard; Vianu, Victor. *Foundation of Databases*. Upper Saddle River: Addison-Wesley, 1995.

Chalmers, Dan. Sensing and Systems in Pervasive Computing: Engineering Context Aware Systems. London: Springer Verlag, 2011.

Dovedan, Zdravko. *Formalni jezici – sintaksna analiza*. Zagreb: Zavod za informacijske studije, 2003.

Ferreira, Denzil; Kostakos, Vassilis; Dey, Anind K. *AWARE: mobile context instrumentation framework*. 20 April 2015. http://journal.frontiersin.org/article/10.3389/fict.2015.00006/full (15 May 2015)

Loke, Seng. *Context-Aware Pervasive Systems: Architecture for a New Breed of Applications*. Boca Raton: Auerbach Publications, 2006.

McKinley, Philip; Sadjadi Seyed, Masoud; Kasten, Eric; Cheng, Betty. *Composing adaptive software*. // *IEEE Computer* 37 (7), 2004. p. 56-64.

Ornellas, Abigail. *2015 is about IoT, Smart Washing Machines and Context-Aware Technology*. 2 December 2014. http://info.groveis.com/blog/2015-is-about-iot-smart-washing-machines-and-context-aware-technology (26 May 2015)

Schilit, Bill N.; Adams, Norman; Want, Roy. *Context-Aware Computing Applications*. // *IEEE Workshop on Mobile Computing Systems and Applications*, December 8-9 1994.

Sipser, Michael. *Introduction to the Theory of Computation, 3rd Edition*. Boston: Cengage Learning, 2013.

Sound pressure. 15 May 2015. http://en.wikipedia.org/wiki/Sound_pressure (15 May 2015)