CCA: a Calculus of Context-aware Ambients

François Siewe, Antonio Cau and Hussein Zedan
Software Technology Research Laboratory
De Montfort University
Leicester, United Kingdom
Email: {fsiewe, acau, hzedan}@dmu.ac.uk

Abstract

We present a process calculus, CCA, for the modelling and verification of mobile systems that are context-aware. This process calculus is built upon the calculus of mobile ambients and introduces new constructs to enable ambients and processes to be aware of the environment in which they are being executed. This results in a powerful calculus where mobility and context-awareness are first-class citizens. We present the syntax and a formal semantics of the calculus. We show that CCA can encode the π-calculus, and illustrate the pragmatics of the calculus through a case study of a context-aware hospital bed.

Keywords. *Context-awareness, process calculus, mobile ambient, pervasive computing*

1. Introduction

Pervasive computing is a new paradigm for next-generation distributed systems where computers disappear in the background of the users everyday activities. In such a paradigm computation is performed on a multitude of often invisible small devices interconnected through a wireless network. Fundamental to pervasive computing is that any component (including user, hardware and software) can be *mobile* and that computations are *context-aware*. As a result, mobility and context-awareness are important features of any design framework for pervasive computing applications. Context-awareness requires applications to be able to sense aspects of the environment and use this information to adapt their behaviour in response to changing situations. Despite the advances in mobile computing, there is a fundamental lack of linguistic supports and mechanisms for modelling context-awareness in mobile applications.

In order to address these issues we propose the Calculus of Context-aware Ambients (CCA in short), that is built upon the well-defined Calculus of Boxed Ambient (CBA in short) [1] which inherits the mobility model of the Ambient Calculus (AC in short) [2]. We extend CBA with new constructs to enable mobile ambients and processes to be aware of the environment in which they are being executed. This results in a powerful calculus where mobility and context-awareness are first-class citizens. Our contributions are summarised as follows:

- we propose a logical language for expressing properties of the contexts of CCA processes (Sect. 3). We call a formula in this logic a *context expression*. Context expressions are used in CCA to guard capabilities so that they are performed only in specified contexts. A *context-guarded capability* has the form $\kappa?M$ where the guard κ is a context expression and M is a capability. Such a capability is performed only when the environment satisfies its guard. We give the semantics of context expressions in terms of a satisfaction relation with respect to a formal model of contexts based on the hierarchical structure of ambients.
- we give the syntax (Sect. 2) and formal semantics (Sect. 4) of CCA which extends CBA with the following features: (i) *context-guarded capability* as explained above; (ii) *process abstraction* as a mechanism for context provision; and (iii) *process call* as a mechanism for context acquisition.
- we show that CCA can encode the π-calculus (Sect. 5), and illustrate the pragmatics of the calculus through a case study of a context-aware hospital bed (Sect. 6).

2. Syntax of Processes and Capabilities

This section introduces the syntax of the language of CCA. As in the π-calculus [3], [4], the simplest entities of the calculus are *names*. These are used to name ambients, locations, resources and sensors data. We assume a countably-infinite set of names, elements of which are written in lower-case letters, e.g. n, x and y. We let $\tilde{y}$ denote a sequence of names and $|\tilde{y}|$ the arity of such a sequence. We sometimes use $\tilde{y}$ as a set of names where it is appropriate. We distinguish three main syntactic categories: processes P, capabilities M and context expressions κ.

The syntax of processes and capabilities is given in Table 1. The first five process primitives are defined as for CBA [1]; in particular, an ambient has the form $n[P]$ where n is the ambient's name and the process P is the ambient's body. However CCA departs from the Ambient Calculus and other processes calculi such as [5], [6], [2] with the notion of *context-guarded capabilities*, whereby a capability

is guarded by a context-expression which constrains the environment of the executing process. This is done by preceding a capability with a context expression followed by a question mark as in the process $\kappa?M.P$. Such a process waits until the environment satisfies the context expression κ, then performs the capability M and continues like the process P. The process learns about its context (i.e. its environment) by evaluating the guard. In addition to context-guarded capabilities, we introduce two more constructs in CCA: *process abstraction* and *process call*, inspired by Zimmer's notion of *named macro* [5].

A process abstraction $x \triangleright (\tilde{y}).P$ denotes the linking of the name x to the process P where $\tilde{y}$ is a list of *formal parameters*. This linking is local to the ambient where the process abstraction is defined. So a name x can be linked to a process P in one ambient and to a different process Q in another ambient. A call to a process abstraction named x is done by a capability of the form $\alpha\ x\langle\tilde{z}\rangle$ where α specifies the location where the process abstraction is defined and $\tilde{z}$ is the list of *actual parameters*. There must be as many actual parameters as there are formal parameters to the process abstraction being called. The location α can be $\uparrow$ for any parent, $n\uparrow$ for a specific parent n, $\downarrow$ for any child, $n\downarrow$ for a specific child n, :: for any sibling, $n ::$ for a specific sibling n, or ϵ (empty string) for the calling ambient itself. A process call $\alpha\ x\langle\tilde{z}\rangle$ behaves like the process linked to x at location α, in which the formal parameters have been substituted for the actual parameters $\tilde{z}$. A process call can only take place if the corresponding process abstraction is *available* at the specified location.

Table 1. Syntax of CCA processes and capabilities

P, Q	$::=$	$\mathbf{0} \mid P\|Q \mid (\nu n)\ P \mid n[P] \mid !P \mid \kappa?M.P \mid x \triangleright (\tilde{y}).P$
M	$::=$	$\texttt{in}\ n \mid \texttt{out} \mid \alpha\ x\langle\tilde{y}\rangle \mid \alpha\ (\tilde{y}) \mid \alpha\ \langle\tilde{y}\rangle$
α	$::=$	$\uparrow \mid n\uparrow \mid \downarrow \mid n\downarrow \mid :: \mid n :: \mid \epsilon$

For example: i) the capability $\uparrow\ x\langle\tilde{z}\rangle$ waits until the process abstraction x is defined in the parent of the calling ambient, then makes a process call to x; ii) the capability $::\ x\langle\tilde{z}\rangle$ waits until the process abstraction x is defined in one of the siblings of the calling ambient, then makes a process call to x; iii) the capability $n\downarrow x\langle\tilde{z}\rangle$ waits until the process abstraction x is defined in a child ambient named n of the calling ambient, then makes a process call to x; and vi) the capability $x\langle\tilde{z}\rangle$ waits until the process abstraction x is defined in the calling ambient, then makes a process call to x. In CCA, an ambient provides context by (re)defining process abstractions to account for its specific functionality. Ambients can interact with each other by making process call. Because ambients are mobile, the same process call, e.g. $\uparrow x\langle\tilde{z}\rangle$, may lead to different behaviours depending on the location of the calling ambient. So process abstraction is used as a mechanism for context provision while process call is a mechanism for context acquisition.

Ambients exchange messages using the capability $\alpha\ \langle\tilde{z}\rangle$ to send a list of messages $\tilde{z}$ to a location α, and the capability $\alpha\ (\tilde{y})$ to receive a list of messages in $\tilde{y}$ from a location α. The mobility capabilities `in` and `out` are defined as in CBA [1].

3. Context Expressions

3.1. Context Model

In CCA the notion of *ambient* is used to model any entity relevant to the application under consideration: a user, a location, a device or a software agent. As described in Sect. 2, an ambient has a name, a boundary, a collection of local processes and can contain other ambients. Meanwhile, an ambient can move from location to another by performing the mobility capabilities `in` and `out`. It follows that the structure of a CCA process, at any time, is a hierarchy of nested ambients. This hierarchical structure self-reconfigures as the process executes. In such a structure, the context of a sub-process is obtained by replacing in the structure that sub-process by a placeholder '$\odot$' as illustrated in Example 3.1.

Example 3.1: Suppose a application is modelled by the process $P \mid n[Q \mid m[R \mid S]]$. So, the context of the process R in that application is $P \mid n[Q \mid m[\odot \mid S]]$, and that of ambient m is $P \mid n[Q \mid \odot]$.

Our context model is depicted by the grammar in (1), where the symbol C stands for context, n ranges over names and P ranges over processes (as defined in Table 1). The context $\mathbf{0}$ is the empty context, also called the *nil* context. It contains no context information. The position of a process in that process' context is denoted by the symbol $\odot$. This is a special context called the *hole context*. The context $(\nu n)\ C$ means that the scope of the name n is limited to the context C. The context $n[C]$ means that the internal environment of the ambient n is described by the context C. The context $C \mid P$ says that the process P runs in parallel with the context C, and so C is part of process P's context.

$$C ::= \mathbf{0} \mid \odot \mid n[C] \mid C|P \mid (\nu n)\ C \qquad (1)$$

We let $C_1(C_2)$ denote the substitution of C_2 for each occurrence of $\odot$ in C_1. The hole $\odot$ plays an important role in our context model. In fact a context C containing a single hole represents the environment of a process P in the process $C(P)$. An algebraic semantics of contexts is given in Table 2 in terms of equalities, where the set $\mathbf{fn}(C)$ of free names in a context C is defined as for processes. The first three equalities say that parallel composition of contexts has a unit element $\mathbf{0}$, and is commutative and associative, respectively. The equalities (Cont-3) to (Cont-6) are related

to the manipulation of scopes. The last set of equalities state that equality propagates across scopes, parallel composition and ambient nesting, respectively.

Table 2. Algebraic semantics of contexts

$C \mid \mathbf{0} = C$	(Cont-0)
$C_1 \mid C_2 = C_2 \mid C_1$	(Cont-1)
$C_1 \mid (C_2 \mid C_3) = (C_1 \mid C_2) \mid C_3$	(Cont-2)
$(\nu n)\ (\nu m)\ C = (\nu m)\ (\nu n)\ C$	(Cont-3)
$(\nu n)\ C_1 \mid C_2 = (\nu n)\ (C_1 \mid C_2)$ if $n \notin \mathbf{fn}(C_2)$	(Cont-4)
$(\nu n)\ m[C] = m[(\nu n)\ C]$ if $n \neq m$	(Cont-5)
$(\nu n)\ \mathbf{0} = \mathbf{0}$	(Cont-6)
$C_1 = C_2 \Rightarrow (\nu n)\ C_1 = (\nu n)\ C_2$	(Cont-7)
$C_1 = C_2 \Rightarrow C_1 \mid C_3 = C_2 \mid C_3$	(Cont-8)
$C_1 = C_2 \Rightarrow n[C_1] = n[C_2]$	(Cont-9)

In order to navigate through the hierarchical structure of context, we define a spatial reduction relation $\downdownarrows$ for context as follows:

$$C_1 \downdownarrows C_2 \quad \widehat{=} \quad C_1 = n[C_2] \mid C_3, \tag{2}$$

for some name n and some context C_3. The equation (2) says that the context C_1 contains the context C_2 within exactly one level of nesting. Theorem 3.1 asserts that the spatial reduction relation $\downdownarrows$ is closed under equality. Its proof is immediate from (2) and the equalities in Table 2.

Theorem 3.1: $C_1 = C_2, C_2 \downdownarrows C_2', C_2' = C_1' \Rightarrow C_1 \downdownarrows C_1'$.

We let $\downdownarrows^*$ denote the reflexive and transitive closure of the spatial reduction relation $\downdownarrows$.

3.2. Context Expressions

Based on the formal representation of contexts presented in the previous section, we define a logical language for specifying the properties of contexts. Its syntax is given in Table 3 where κ ranges over formulae called *Context Expressions* (CEs in short), n ranges over names and x is a variable symbol which also ranges over names.

Table 3. Syntax of context expressions

$$\kappa \quad ::= \quad \mathbf{T} \mid n = m \mid \bullet \mid \neg\kappa \mid \kappa_1 | \kappa_2 \mid \kappa_1 \wedge \kappa_2 \mid n[\kappa] \mid \mathtt{new}(n, \kappa) \mid \oplus\kappa \mid \diamondsuit\kappa \mid \exists x.\kappa$$

The formal semantics of context expressions is given by the satisfaction relation $\models$ defined in Table 4, where C is universally quantified over the set of all contexts. In Table 4 the notation $C \models \kappa$ states that the context C satisfies the context expression κ. We also denote by $\kappa\{x \leftarrow n\}$ the substitution of n for each free occurrence of x in κ. We now explain the meaning of context expressions. The CE $\mathbf{T}$ holds for all contexts described by the grammar in (1). It stands for the truth value $true$. A CE of the form $n = m$ holds for a context if the names n and m are lexically identical. The CE $\bullet$ holds solely for the hole context $\odot$. This CE is particularly important as it denotes in a context expression the position of the process evaluating that context expression.

First order operators such as negation ($\neg$), conjunction ($\wedge$) and existential quantification ($\exists$) expand their usual semantics to context expressions. A CE $\kappa_1 \mid \kappa_2$ holds for a context if that context is a composition of two contexts C_1 and C_2 such that κ_1 holds for C_1 and κ_2 holds for C_2. A CE $n[\kappa]$ holds for a context if that context is of the form $n[C]$ such that κ holds for the context C. A CE $\mathtt{new}(n, \kappa)$ holds for a context if that context has the form $(\nu n)\ C$ such that κ holds for the context C. A CE $\oplus\kappa$ holds for a context if that context can reduce in one step (with respect to the spatial reduction relation '$\downdownarrows$' defined in (2)) into a context for which κ holds. The operator $\oplus$ is called *spatial next modality*. A CE $\diamondsuit\kappa$ holds for a context if there exists somewhere in that context a sub-context for which κ holds. The operator $\diamondsuit$ is called *somewhere modality*.

Table 4. Satisfaction relation for context expressions

$C \models \mathbf{T}$			(Sat-true)
$C \models n = n$			(Sat-match)
$C \models \bullet$	iff	$C = \odot$	(Sat-hole)
$C \models \neg\kappa$	iff	$C \not\models \kappa$	(Sat-neg)
$C \models \kappa_1 \mid \kappa_2$	iff	exists C_1, C_2 s.t. $C = C_1 \mid C_2$ and $C_1 \models \kappa_1$ and $C_2 \models \kappa_2$	(Sat-par)
$C \models \kappa_1 \wedge \kappa_2$	iff	$C \models \kappa_1$ and $C \models \kappa_2$	(Sat-and)
$C \models n[\kappa]$	iff	exists C' s.t. $C = n[C']$ and $C' \models \kappa$	(Sat-amb)
$C \models \mathtt{new}(n, \kappa)$	iff	exists C' s.t. $C = (\nu n)\ C'$ and $C' \models \kappa$	(Sat-new)
$C \models \oplus\kappa$	iff	exists C' s.t. $C \downdownarrows C'$ and $C' \models \kappa$	(Sat-next)
$C \models \diamondsuit\kappa$	iff	exists C' s.t. $C \downdownarrows^* C'$ and $C' \models \kappa$	(Sat-sw)
$C \models \exists x.\kappa$	iff	exists n s.t. $C \models \kappa\{x \leftarrow n\}$	(Sat-exist)

Table 5 lists some derived connectives, illustrating properties that can be expressed in the logic. Their informal meaning can be understood in a usual manner as in classical predicate logic. The following theorem is a fundamental

Table 5. Derived connectives

$$\kappa_1 \vee \kappa_2 \mathrel{\widehat{=}} \neg(\neg\kappa_1 \wedge \neg\kappa_2), \qquad \kappa_1 \Rightarrow \kappa_2 \mathrel{\widehat{=}} \neg\kappa_1 \vee \kappa_2$$
$$\mathbf{F} \mathrel{\widehat{=}} \neg\mathbf{T}, \qquad \kappa_1 \Leftrightarrow \kappa_2 \mathrel{\widehat{=}} (\kappa_1 \Rightarrow \kappa_2) \wedge (\kappa_2 \Rightarrow \kappa_1)$$

property of the satisfaction relation $\models$; it states that satisfaction is invariant under equality of contexts. That is, logical formulae can only express properties that are invariant up to equality. The proof of Theorem 3.2 is straightforward by induction on the number of connectives in κ.

Theorem 3.2: **(Satisfaction is up to =)**
$(C \models \kappa \ \wedge \ C = C') \ \Rightarrow \ C' \models \kappa$.

Definition 3.1: **(Validity)** A context expression κ is valid,

and we write $\models \kappa$, if it holds for all contexts, i.e.

$$\models \kappa \text{ iff } \ C \models \kappa, \text{ for all context } C.$$

3.3. Examples of Context Expressions

We now give some examples to show how context expressions can be used to specify the properties of the context of CCA processes. In the following samples of CEs we take the view that a CE is evaluated by the immediate ambient λ say that contains it; the parameters n and m are ambient names.

(a) $\mathtt{has}(n) \mathrel{\widehat{=}} \oplus(\bullet \mid n[\mathbf{T}] \mid \mathbf{T})$ holds if n is located at λ.

(b) $\mathtt{at}(n) \mathrel{\widehat{=}} n[\oplus(\bullet \mid \mathbf{T})] \mid \mathbf{T}$ holds if λ is located at n.

(c) $\mathtt{with}(n) \mathrel{\widehat{=}} n[\mathbf{T}] \mid \oplus(\bullet \mid \mathbf{T})$ holds if λ is co-located with n.

(d) $\mathtt{in\_with}(n) \mathrel{\widehat{=}} \oplus(\bullet \mid \mathbf{T}) \mid \oplus(n[\mathbf{T}] \mid \mathbf{T})$ holds if λ is co-located with the ambient n's parent.

(e) $\mathtt{out\_with}(n) \mathrel{\widehat{=}} n[\mathbf{T}] \mid \oplus\oplus(\bullet \mid \mathbf{T})$ holds if n is co-located with λ's parent.

(f) $\mathtt{out\_at}(n) \mathrel{\widehat{=}} n[\oplus\oplus(\bullet \mid \mathbf{T})] \mid \mathbf{T}$ holds if λ is a grand-child of n.

(g) $\mathtt{near}(n) \mathrel{\widehat{=}} \mathtt{has}(n) \vee \mathtt{at}(n) \vee \mathtt{with}(n) \vee \mathtt{out\_at}(n) \vee \mathtt{in\_with}(n) \vee \mathtt{out\_with}(n)$ holds if n is nearby λ.

(h) $\mathtt{at2}(n,m) \mathrel{\widehat{=}} n[m[\mathbf{T}] \mid \mathbf{T}] \mid \mathbf{T}$ holds if m is located at n.

The context expression $\mathtt{has}(n)$ holds if the ambient λ evaluating the expression has no parent and contains an ambient named n. For example, $\mathtt{has}(pda)$ holds for the context $bob[\odot \mid pda[\mathbf{0}]]$ and we write:

$$bob[\odot \mid pda[\mathbf{0}]] \ \models \ \mathtt{has}(pda). \tag{3}$$

Here the ambient evaluating the CE is bob, i.e. $\lambda = bob$; the ambient bob has no parent and contains an ambient named pda. The formal proof of (3) is given by Table 6, based on the algebraic semantics of context defined in Table 2, the satisfaction relation defined in Table 4 and Theorem 3.2. Note that the CE $\mathtt{has}(pda)$ does not hold for the context

Table 6. Formal proof of (3)

1. $\mathbf{0} \models \mathbf{T}$	{(Sat-true)}
2. $pda[\mathbf{0}] \models pda[\mathbf{T}]$	{1. and (Sat-amb)}
3. $pda[\mathbf{0}] \mid \mathbf{0} \models pda[\mathbf{T}] \mid \mathbf{T}$	{1., 2. and (Sat-par)}
4. $\odot \models \bullet$	{(Sat-hole)}
5. $\odot \mid pda[\mathbf{0}] \mid \mathbf{0} \models \bullet \mid pda[\mathbf{T}] \mid \mathbf{T}$	{3., 4. and (Sat-par)}
6. $(\odot \mid pda[\mathbf{0}] \mid \mathbf{0}) = (\odot \mid pda[\mathbf{0}])$	{(Cont-0)}
7. $\odot \mid pda[\mathbf{0}] \models \bullet \mid pda[\mathbf{T}] \mid \mathbf{T}$	{5., 6. and Theorem 3.2}
8. $bob[\odot \mid pda[\mathbf{0}]] \models \oplus(\bullet \mid pda[\mathbf{T}] \mid \mathbf{T})$	{7. and (Sat-next)}

$conf[bob[\odot \mid pda[\mathbf{0}]]]$ because the evaluating ambient bob has a parent which is the ambient $conf$; viz

$$conf[bob[\odot \mid pda[\mathbf{0}]]] \ \not\models \ \mathtt{has}(pda).$$

But using the spatial operator $\oplus$ to move one step down in that hierarchy, we have:

$$conf[bob[\odot \mid pda[\mathbf{0}]]] \ \models \ \oplus\, \mathtt{has}(pda).$$

Similarly, the CE $\diamond\mathtt{has}(pda)$ holds for any context that has somewhere the context $bob[\odot \mid pda[\mathbf{0}]]$ as sub-context.

4. Semantics of CCA

As customary, the operational semantics of CCA is defined using a structural congruence $\equiv$ and a reduction relation $\rightarrow$. The structural congruence for CCA is the smallest congruence relation on processes that satisfies the axioms in Table 7, where '$\cdots$' refers to the axioms defined in [1] for CBA. The axiom $\mathbf{T}?M.P \equiv M.P$ says that a capability without guard is the same as that capability guarded with $\mathbf{T}$. The next two axioms define the equivalence of context-guarded processes and process abstractions respectively.

Table 7. Structural congruence for processes

$$\begin{array}{l} \mathbf{T}?M.P \equiv M.P \\ P \equiv Q, (\models \kappa \Leftrightarrow \kappa') \Rightarrow \kappa?P \equiv \kappa'?Q \\ P \equiv Q \Rightarrow x \triangleright (\tilde{y}).P \equiv x \triangleright (\tilde{y}).Q \\ \cdots \end{array}$$

The reduction relation of processes is defined in Table 8, where '$\cdots$' refers to the axioms defined in [1] for CBA. The first set of rules (Red Call) gives the semantics of a process call. It states that a process call takes place only if a corresponding process abstraction is defined at the specified location. The rule (Red Guard) links the context model presented in Sect. 3.1 to the reduction relation. It asserts that a context-guarded capability reduces in a context if that context satisfies the guard of that capability.

5. Expressiveness of CCA

The encoding of the asynchronous π-calculus in CCA follows from the encoding of the asynchronous π-calculus in the Calculus of Boxed Ambient [1] which CCA is a conservative extension. Due to the space limit, this encoding is not given here and we refer interested readers to [1] for a detailed presentation of this encoding. We can then conclude that CCA is at least as expressive as the π-calculus.

6. Application to Health Care

We illustrate the usability of CCA through a case study of the *context-aware hospital bed* designed in the *Hospital of the Future* project at the Centre for Pervasive Health Care, Denmark [7], [8]. The bed has an integrated computer and a touch sensitive display which is used by the patients for entertainment purposes (e.g. for watching television) and the

Table 8. Reduction relation for processes

$x \triangleright (\tilde{y}).P \mid x\langle \tilde{z} \rangle$	$\rightarrow$	$x \triangleright (\tilde{y}).P \mid P\{\tilde{y} \leftarrow \tilde{z}\}$	(Red Call Lc)
$n[Q \mid x \triangleright (\tilde{y}).P] \mid m[:: x\langle \tilde{z} \rangle \mid R]$	$\rightarrow$	$n[Q \mid x \triangleright (\tilde{y}).P] \mid m[P\{\tilde{y} \leftarrow \tilde{z}\} \mid R]$	(Red Call S1)
$n[Q \mid x \triangleright (\tilde{y}).P] \mid m[n :: x\langle \tilde{z} \rangle \mid R]$	$\rightarrow$	$n[Q \mid x \triangleright (\tilde{y}).P] \mid m[P\{\tilde{y} \leftarrow \tilde{z}\} \mid R]$	(Red Call S2)
$Q \mid x \triangleright (\tilde{y}).P \mid m[\uparrow x\langle \tilde{z} \rangle \mid R]$	$\rightarrow$	$Q \mid x \triangleright (\tilde{y}).P \mid m[P\{\tilde{y} \leftarrow \tilde{z}\} \mid R]$	(Red Call U1)
$n[Q \mid x \triangleright (\tilde{y}).P \mid m[n \uparrow x\langle \tilde{z} \rangle \mid R]]$	$\rightarrow$	$n[Q \mid x \triangleright (\tilde{y}).P \mid m[P\{\tilde{y} \leftarrow \tilde{z}\} \mid R]]$	(Red Call U2)
$Q \mid \downarrow x\langle \tilde{z} \rangle \mid m[R \mid x \triangleright (\tilde{y}).P]$	$\rightarrow$	$Q \mid P\{\tilde{y} \leftarrow \tilde{z}\} \mid m[R \mid x \triangleright (\tilde{y}).P]$	(Red Call D1)
$Q \mid m \downarrow x\langle \tilde{z} \rangle \mid m[R \mid x \triangleright (\tilde{y}).P]$	$\rightarrow$	$Q \mid P\{\tilde{y} \leftarrow \tilde{z}\} \mid m[R \mid x \triangleright (\tilde{y}).P]$	(Red Call D2)
$C(M.P) \rightarrow C'(P) \quad \Rightarrow \quad C(\kappa?M.P) \rightarrow C'(P)$		if $C \models \kappa$	(Red Guard)
$\cdots$			

clinicians for accessing medical data while working at the bed. The bed is aware of who is using it (i.e. the identity of the patient), and what and who is near it. For example, the bed is aware of the nurse, the patient and the medicine tray. The bed runs a context-aware EPR (Electronic Patient Record) client. Based on the location of the nurse, the patient and the medicine tray, the bed can automatically log in the nurse, find the patient record, display the medicine schema, and in this schema highlight the prescribed medicine which is in the pill container. The nurse is automatically logged out of the computer when she leaves the active zone of the bed. This mechanism of logging in and logging out a user based on its proximity is called *proximity-based user authentication* [7], [8]. The notion of *active zone* is used in the prototype context-aware bed to delimit the range of the bed awareness. The bed is aware of changes that occur within its active zone; for example when a nurse enters or leaves that zone.

We consider six entities: the bed, the patient, the nurse, the medicine tray, the pill container and the active zone. Each of these entities is represented as an ambient in CCA. Initially, the bed is located in its active zone, the patient is inside the bed, and the nurse and medicine tray containing the pill containers are outside the active zone of the bed as depicted in Fig. 1. This is modelled by the following process:

$$\begin{array}{ll} & nurse[P_{\mathrm{n}}] \\ \mid & tray[P_{\mathrm{t}} \mid con_1[P_1] \mid \cdots \mid con_k[P_k]] \\ \mid & zone[bed[P_{\mathrm{b}} \mid patient\_001[P_{\mathrm{p}}]]] \end{array}$$

where P_{x} is a process specifying the behaviour of its host ambient.

Nurse Ambient. The nurse can enter and leave the active zone as often as needed in the course of her work activities. This is done by the corresponding ambient performing the capability in $zone$ to enter the zone and the capability out to leave the zone. Once in the active zone, the nurse can access the patient EPR using the touch screen embedded in the bed. We assume that the nurse bring with her the medicine tray in and out the active zone. This requires a synchronisation between the ambient representing the nurse and the ambient modelling the medicine tray. Such a synchronisation is modelled using handshake message passing communication primitives. So the behaviour of the nurse can be modelled as follows:

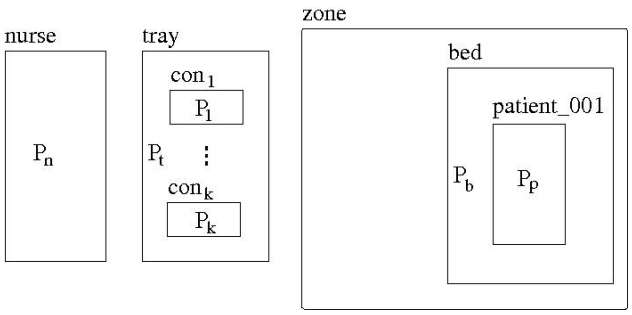

Figure 1. Context-aware hospital bed system

$$\begin{array}{rcll} P_{\mathrm{n}} & \widehat{=} & ! \ \mathtt{with}(zone)?tray :: \langle\rangle.\mathtt{in}\ zone.\mathbf{0} & (i) \\ & \mid & ! \ \mathtt{at}(zone)?bed :: \mathtt{epr}\langle\rangle.\mathbf{0} & (ii) \\ & \mid & ! \ \mathtt{at}(zone)?tray :: \langle\rangle.\mathtt{out}.\mathbf{0} & (iii) \end{array}$$

The first component (i) in P_{n} says that when the nurse ambient is next to the active zone (see CE '$\mathtt{with}(\cdot)$' on page 4) and is willing to enter the zone, it sends an empty message to the tray ambient to signal its intention to move in the active zone. When the tray ambient responds by receiving this message, the nurse ambient enters the zone by performing the capability in $zone$. Once in the active zone (see CE '$\mathtt{at}(\cdot)$' on page 4), the nurse ambient can access the patient record by calling the process abstraction epr located in the bed ambient. This is modelled by the second component (ii). The last component (iii) says that the nurse ambient signals its intention to leave the active zone to the tray ambient by sending an empty message to it. At the receipt of the message, the nurse ambient leaves the zone by performing the capability out.

Pill Container Ambient. A pill container is aware of the patient and reveals itself when near the patient by lighting the name of the patient. For example, if the pill container con_i, for some i such that $1 \leq i \leq k$, contains the medicine of the patient named $parent\_001$, then the behaviour of the pill container is specified as follows:

$$P_i \quad \widehat{=} \quad ! \ \diamond\mathtt{near}(patient\_001)? \uparrow \langle patient\_001 \rangle.\mathbf{0}$$

So the pill container con_i lights the patient name by sending the name to its parent ambient, which will eventually propagate the message.

Tray Ambient. It is assumed that the tray ambient follows the nurse ambient in and out the active zone to supply patient medicine stored in pill containers. So the tray ambient communicates with the nurse ambient to know when to move in and out the active zone. This is modelled as follows and the explanation is similar to that of the nurse ambient.

$$\begin{aligned} P_{\mathsf{t}} \;\widehat{=}\;\; & !\,\mathtt{with}(zone)?nurse :: ().\mathtt{in}\ zone.\mathbf{0} \\ \mid\; & !\,\mathtt{at}(zone)?nurse :: ().\mathtt{out}.\mathbf{0} \\ \mid\; & !\downarrow(msg). :: \langle msg\rangle.\mathbf{0} \end{aligned}$$

The tray ambient is able to receive messages from the pill container ambients it contains (by performing the capability '$\downarrow(msg)$') and forward them to the ambients around it such as the nurse ambient and the bed ambient (by performing the capability '$:: \langle msg\rangle$').

Patient Ambient. The patient ambient selects an entertainment program by sending a request to the bed ambient as follows:

$$P_{\mathsf{p}} \;\widehat{=}\; !\,(\nu req)\ bed \uparrow \langle req\rangle.\mathbf{0}$$

The restriction operator 'ν' models here the *freshness* of a request.

Bed Ambient. The bed ambient is located in its active zone which delimits the context of the bed. One of the most important context-awareness properties of the bed is the ability of logging the nurse in when she enters the active zone and logging her out when she leaves that zone, automatically. This is modelled as follows:

$$\begin{aligned} P_{\mathsf{b}} \;\widehat{=}\;\; & (\diamond\mathtt{at2}(zone, nurse))?\mathtt{login}\langle nurse\rangle.\mathbf{0} \quad (a) \\ \mid\; & !\,(\diamond\neg\mathtt{at2}(zone, nurse))?\mathtt{logout}\langle\rangle. \\ & (\diamond\mathtt{at2}(zone, nurse))?\mathtt{login}\langle nurse\rangle.\mathbf{0} \quad (b) \end{aligned}$$

The context-guarded capability in (a) says that when the nurse enter the bed's active zone, the action '$\mathtt{login}\langle nurse\rangle$' is taken to log the nurse in the EPR system. The nurse is logged out when she leaves the active zone as specified in (b). The replication operator '!' in (b) means that the sequence '(a) followed by (b)' is repeated forever. Due to space limit, the process abstractions `login`, `logout` and `epr` are not detailed here.

7. Related Work

Zimmer [5] proposes a context-awareness calculus which features a hierarchical structure similar to mobile ambients, and a generic multi-agent synchronisation mechanism inspired from the join-calculus. This work has been extended in [6] by Bucur and Nielson to enable ambients to publish context information upwards in the hierarchy of ambients. In both work, a piece of primary context information is a capability modelled by a named macro, similar to the notion of process call in `CCA`. Unlike their approach and to preserve the autonomy of ambients, we do not allow an ambient to remotely create a process abstraction in another ambient. Moreover, none of these calculi enable context-guarded capabilities. Roman [9] builds a language-based model of context-aware systems where contexts are provided through exposed variables as opposed to private variables.

8. Conclusion and Future Work

In this paper we have presented `CCA`, a calculus of context-aware mobile ambients. The syntax, formal semantics and expressiveness of `CCA` have been presented in this paper. A case study of a context-aware hospital bed is utilised to illustrate the usability of the calculus.

In future work, we will investigate a theory of behavioural equivalence for context-aware mobile ambients based upon the technique developed in the Ambient Calculus [2] using Morris-style contextual equivalence. We will also investigate type systems for verifying key properties of context-aware mobile applications such as safety, security and privacy.

References

[1] M. Bugliesi, G. Castagna, and S. Crafa, "Access Control for Mobile Agents: The Calculus of Boxed Ambients," *ACM Trans. on Programming Languages and Systems*, vol. 26, no. 1, pp. 57–124, January 2004.

[2] L. Cardelli and A. Gordon, "Mobile Ambients," *Theoretical Computer Science*, vol. 240, pp. 177–213, 2000.

[3] R. Milner, *Communication and Mobile Systems: The π-Calculus*. Cambridge University Press, 1999.

[4] D. Sangiorgi and D. Walker, *The π-calculus: A Theory of Mobile Processes*. Cambridge University Press, 2001.

[5] P. Zimmer, "A Calculus for Context-awareness," BRICS, Tech. Rep., 2005.

[6] D. Bucur and M. Nielsen, "Secure Data Flow in a Calculus for Context Awareness," in *Concurrency, Graphs and Models*, ser. Lecture Notes in Computer Science, vol. 5065. Springer, 2008, pp. 439–456.

[7] J. Bardram, "Hospitals of the Future–Ubiquitous Computing Support for Medical Work," in *Hospitals Workshop Ubihealth 2003*, 2003.

[8] ——, "Aplications of Context-aware Computing in Hospital Work–Examples and Design Principles," in *Proceedings of ACM Symposium on Applied Computing*, March 2004, pp. 1574–1579.

[9] G. Roman, C. Julien, and J. Payton, "A Formal Treatment of Context-Awareness," in *FASE*, ser. LNCS, no. 2984. Springer, 2004, pp. 12–36.