The Calculus of Context-aware Ambients (CCA)

How to program: Context-aware Processes

François Siewe

Software Technology Research Laboratory (STRL)
Cyber Technology Institute (CTI)
De Montfort University
Leicester, UK
fsiewe@dmu.ac.uk or fsiewe@yahoo.fr
http://www.cse.dmu.ac.uk/∼fsiewe/

November 2022

DE MONTFORT UNIVERSITY LEICESTER

Overview

Learning Outcomes

By the end of this lesson you should know:

- Context-aware processes
- How to model context-aware IoT systems
- Other modelling formalisms

DE MONTFORT UNIVERSITY LEICESTER

Context-aware Processes

A context-aware process is a process that contains a process of one of the following forms

- $<\kappa>M.P$ Context-guarded prefix
- $\texttt{if}\ \ <\kappa_1>M_1.P_1\ \ldots\ <\kappa_\ell>M_\ell.P_\ell\ \ \texttt{fi}$ Selection (if)
- $\texttt{if}\ <\kappa_1>M_1.P_1\ \ldots\ <\kappa_\ell>M_\ell.P_\ell\ \texttt{else}\ P\ \texttt{fi}$ (if-else)
- $\texttt{find}\ \ y_1, y_2, \ldots, y_\ell : \kappa\ \ \texttt{for}\ \ P$ Search

Context-guarded Prefix ($<\kappa>M.P$)

$<\kappa>M.P$

- waits until the environment satisfies the context expression κ, then performs the capability M and continues like the process P.
- We write $M.P$ in place of $<\texttt{true}>M.P$.

Example

- $<\texttt{at}(conf)>\texttt{send}(hello).\mathbf{0}$
- $<\texttt{with}(bob)>bob :: \texttt{send}(hello).\mathbf{0}$
- $<\texttt{at}(conf)>\texttt{recv}(x).\mathbf{0}$
- $<x = 2>\texttt{send}(hello).\mathbf{0}$

Definition: A process $<\kappa>M.P$ is enabled in a context if the condition κ holds and the capability M can be executed in that context.

DE MONTFORT UNIVERSITY LEICESTER

Lab Session 6 – Try it out (Context-guarded Prefix)

1. $\texttt{recv}(x, y).\{$
   $\quad < x > y > \texttt{send}(\textit{Greater_than}).\mathbf{0} \mid$
   $\quad < x = y > \texttt{send}(\textit{Equal}).\mathbf{0} \mid$
   $\quad < x < y > \texttt{send}(\textit{Less_than}).\mathbf{0}$
   $\} \mid \texttt{send}(0, 5).\texttt{recv}(z).\mathbf{0}$
2. $< \texttt{this} > \texttt{skip.skip}.\mathbf{0}$
3. $< \texttt{this} > \texttt{skip.skip}.\mathbf{0} \mid \texttt{send}().\mathbf{0}$
4. $< \texttt{this} \mid \texttt{true} > \texttt{skip.skip}.\mathbf{0} \mid \texttt{send}().\mathbf{0}$
5. $< \texttt{this} \mid \texttt{true} > \texttt{skip.skip}.\mathbf{0} \mid \texttt{send}().\mathbf{0} \mid n[\mathbf{0}]$
6. $n[< \texttt{this} > \texttt{skip.skip}.\mathbf{0}]$
7. $n[< \texttt{next this} > \texttt{skip.skip}.\mathbf{0}]$
8. $m[n[< \texttt{next this} > \texttt{skip.skip}.\mathbf{0}]]$
9. $m[n[< \texttt{somewhere this} > \texttt{skip.skip}.\mathbf{0}]]$

DE MONTFORT UNIVERSITY LEICESTER

Selection (`if`)

`if`

$<\kappa_1>M_1.P_1$

$<\kappa_2>M_2.P_2$

$\ldots$

$<\kappa_\ell>M_\ell.P_\ell$

`fi`

- waits until one of the branches, j say, becomes enabled (i.e. can be executed) and proceeds like $<\kappa_j>M_j.P_j$
- if many branches are enabled at the same time, one of them is chosen non-deterministically and executed.

DE MONTFORT UNIVERSITY LEICESTER

Selection (`if`) – Examples

Example

- Deterministic choice

  `if`

  $<x = 2>\texttt{send}(3).\mathbf{0}$

  $<\texttt{not}\ (x = 2)>\texttt{send}(x).\mathbf{0}$

  `fi`

- Non-deterministic choice

  `if`

  $<\texttt{true}>\texttt{send}(1).\mathbf{0}$

  $<\texttt{true}>\texttt{send}(2).\mathbf{0}$

  $<\texttt{true}>\texttt{send}(3).\mathbf{0}$

  $<\texttt{true}>\texttt{send}(4).\mathbf{0}$

  `fi`

DE MONTFORT UNIVERSITY LEICESTER

Lab Session 7 – Try it out (`if`)

1. $\texttt{send}(5).\texttt{recv}(x).\mathbf{0} \mid$
   $\texttt{recv}(x).\texttt{if}$
   $\quad < x = 2 > \texttt{send}(3).\mathbf{0}$
   $\quad < \texttt{not}\ (x = 2) > \texttt{send}(x).\mathbf{0}$
   $\texttt{fi}$
2. $\texttt{recv}(x).\mathbf{0} \mid$
   $\texttt{if}$
   $\quad < \texttt{true} > \texttt{send}(0).\mathbf{0}$
   $\quad < \texttt{true} > \texttt{send}(1).\mathbf{0}$
   $\quad < \texttt{true} > \texttt{send}(3).\mathbf{0}$
   $\quad < \texttt{true} > \texttt{send}(4).\mathbf{0}$
   $\quad < \texttt{true} > \texttt{send}(5).\mathbf{0}$
   $\quad < \texttt{true} > \texttt{send}(6).\mathbf{0}$
   $\texttt{fi}$

DE MONTFORT UNIVERSITY LEICESTER

Selection (if − else)

if
  $<\kappa_1>M_1.P_1$
  $<\kappa_2>M_2.P_2$
  ...
  $<\kappa_\ell>M_\ell.P_\ell$
else P
fi

- If one of the branches, j say, is enabled (i.e. can be executed) then it proceeds like $<\kappa_j>M_j.P_j$
- If many branches are enabled at the same time, one of them is chosen non-deterministically and executed.
- If none of the branches is enabled, then it proceeds like P.

DE MONTFORT UNIVERSITY LEICESTER

Selection (`if − else`) – Examples

Example

- Simple

  `if`
  $<x = 2>$`send`$(3).\mathbf{0}$
  `else` `send`$(x).\mathbf{0}$
  `fi`

- Multiple

  `if`
  $<at(home)>$`send`$(home).\mathbf{0}$
  $<at(conf)>$`send`$(conference).\mathbf{0}$
  $<at(recept)>$`send`$(reception).\mathbf{0}$
  `else` `send`$(away).\mathbf{0}$
  `fi`

DE MONTFORT UNIVERSITY LEICESTER

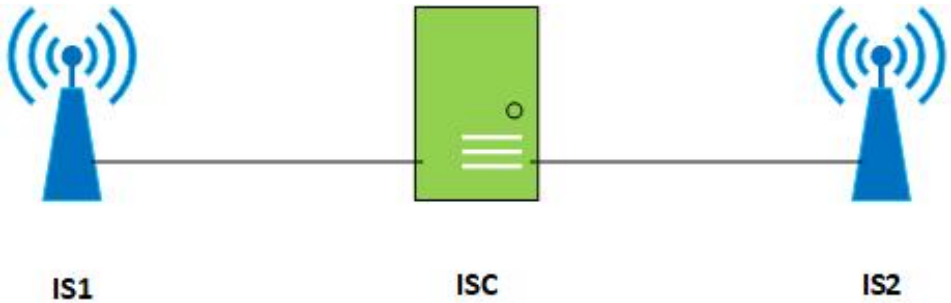
Lab Session 8 – Try it out (`if – else`)

1. $\texttt{send}(5).\texttt{recv}(x).\mathbf{0} \mid$
   $\texttt{recv}(x).\texttt{if}$
   $\quad < x < 2 > \texttt{send}(0).\mathbf{0}$
   $\texttt{else} \quad \texttt{send}(x).\mathbf{0}$
   $\texttt{fi}$
2. $\mathit{home}[\ \mathit{bob}[\texttt{recv}(x).\mathbf{0} \mid$ // change home to conf, recept, ...
   $\quad \texttt{if}$
   $\quad\quad <\mathit{at}(\mathit{home})>\texttt{send}(\mathit{home}).\mathbf{0}$
   $\quad\quad <\mathit{at}(\mathit{conf})>\texttt{send}(\mathit{conference}).\mathbf{0}$
   $\quad\quad <\mathit{at}(\mathit{recept})>\texttt{send}(\mathit{reception}).\mathbf{0}$
   $\quad \texttt{else} \quad \texttt{send}(\mathit{away}).\mathbf{0}$
   $\quad \texttt{fi}\ ]$
   $]$

DE MONTFORT UNIVERSITY LEICESTER

Example: Infostation-based communication system 1

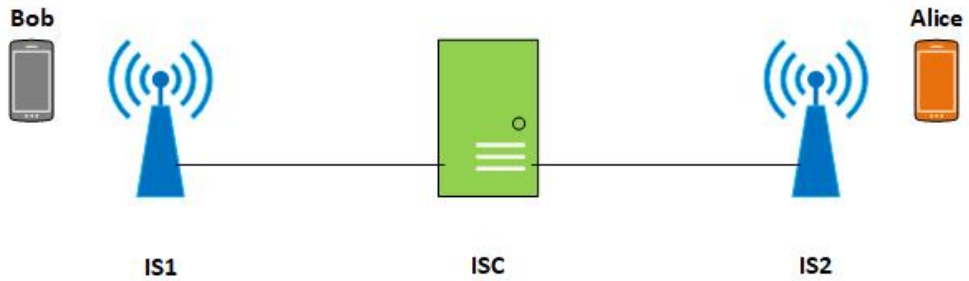
- An infostation system comprises one central infostation centre (ISC) and many wireless access points, called infostations (ISs), deployed at key locations to maximise coverage.
- The ISs are physically connected to the ISC via network cables.
- The ISC is aware of all the ISs connected to it and can communicate with them via the network cables.



DE MONTFORT UNIVERSITY LEICESTER

Example: Infostation-based communication system 2

- An IS is aware of any mobile device within its range and is able to interact with the device.
- A user within the range of an IS can send or receive text messages using a mobile device.
- In this figure Bob is in the range of IS1 while Alice is in the range of IS2.
- Remember that the user is mobile, and can move in/out of the range of the ISs.



Example: Infostation-based communication system 3

Specification of the IS

- When IS receives a text message, it sends the message to the recipient if the latter is within range. Otherwise it forwards the message to ISC.
- An IS can be modelled as an ambient, whose boundary delimits the range of the IS.
- A device is in the range of an IS if the device is a child ambient of the IS.

```
is1[
    !#recv(s,r,msg).if
        < has(r) > r#send(s,r,msg).0
    else isc::send(s,r,msg).0
    fi
  |
    !isc::recv(s,r,msg).if
        < has(r) > r#send(s,r,msg).0
    else skip.0 //drop the message
    fi
]
```

DE MONTFORT UNIVERSITY LEICESTER

Example: Infostation-based communication system 4

Specification of an ISC

- When ISC receives a text message from an IS, it looks for the IS where the recipient is located and forwards the message to that IS.
- The IS where the recipient is located then forwards the message to the recipient mobile device.
- If the recipient is not found, the message is dropped.

```
isc[
    !::recv(s,r,msg).if
        < at(is1,r) > is1::send(s,r,msg).0
        < at(is2,r) > is2::send(s,r,msg).0
    else skip.0 //drop the message
    fi
]
```

Example: Infostation-based communication system 5

Specification of the Infostation system

The whole system is specified as the parallel composition of the ISs and the ISC, i.e.

```
isc[
          ... // as above
]
|
is1[
         ... // as above
]
|
is2[
         ... // as above
]
```

Specification of the user devices

- A user device can receive or/and send messages.
- The user and its device can be modelled as a single ambient named for example after the user.
- Suppose Bob wants to send the message hi to alice.
- Bob's device must have the capability to send a message.

```
bob[
    @send(bob,alice,hi).0
]
```

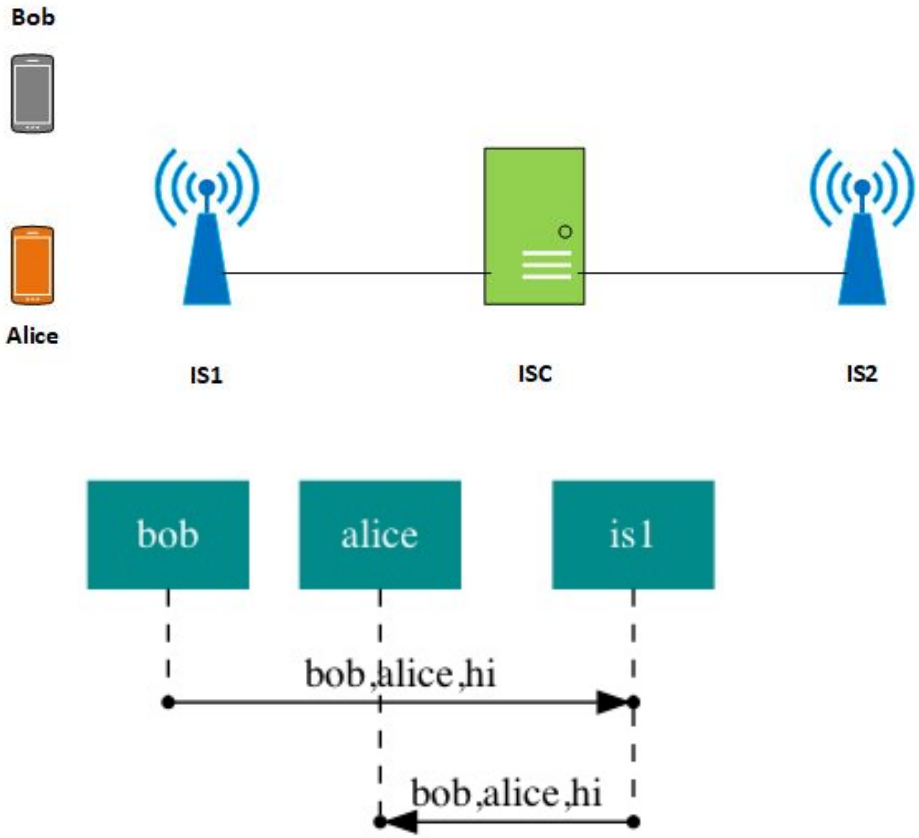
- Alice's device must have the capability to receive a message.

```
alice[
    @recv(s,r,msg).0
]
```

Simulation – Scenario 1

- Suppose Bob and Alice are both in the range of IS1, i.e.

```
is1[
        ... // as above
|  bob[
     @send(bob,alice,hi).0
   ]
|  alice[
     @recv(s,r,msg).0
   ]
]
```



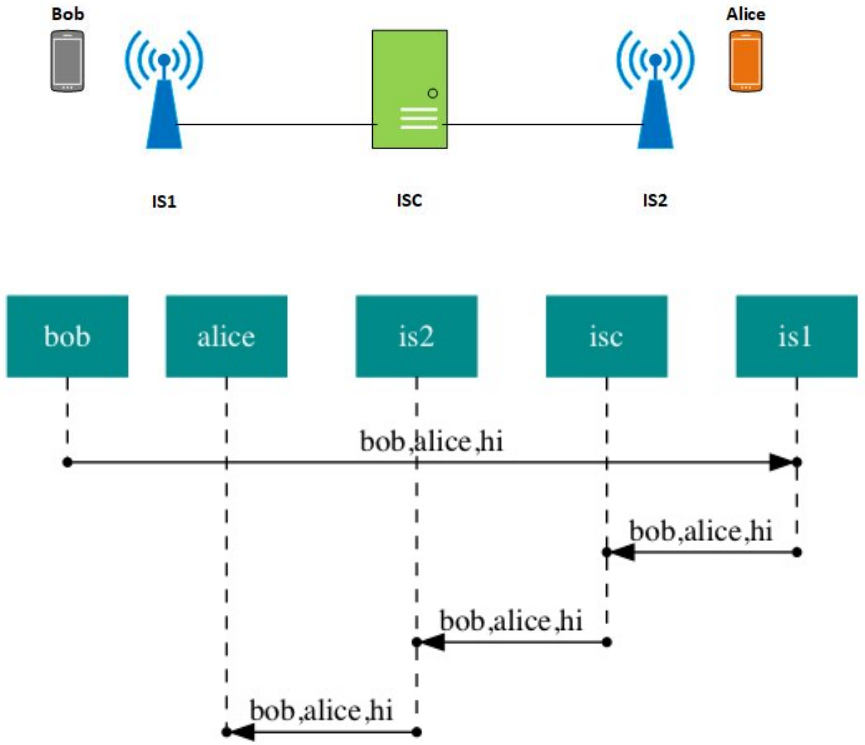
Simulation – Scenario 2

- Bob in the range of IS1

```
is1[
      ... // as above
| bob[
   @send(bob,alice,hi).0
 ]
]
```

- Alice in the range of IS2

```
is2[
     ... // as above
| alice[
   @recv(s,r,msg).0
 ]
]
```



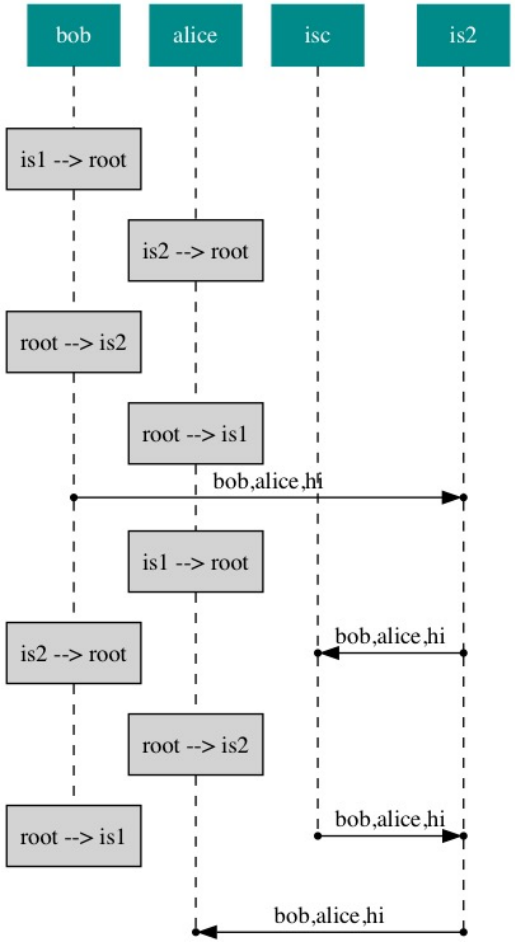
Simulation – Scenario 3

- Bob can move to IS2 and back

```
is1[    ... // as above
| bob[
    @send(bob,alice,hi).0
  |out.in is2.out.in is1.0
  ]
]
```

- Alice can move to IS1 and back

```
is2[    ... // as above
| alice[
    @recv(s,r,msg).0
  |out.in is1.out.in is2.0
  ]
]
```



Search ($\texttt{find}\ y_1, \ldots, y_n : \kappa\ \texttt{for}\ P$)

$\texttt{find}\quad y_1, \ldots, y_n : \kappa\quad \texttt{for}\quad P$

- searches for a list of names $z_1, \ldots, z_n$ such that the environment satisfies the context expression $\kappa\{y_1 \leftarrow z_1, \ldots, y_n \leftarrow z_n\}$ and then continues like $P\{y_1 \leftarrow z_1, \ldots, y_n \leftarrow z_n\}$.

Example

- Find current location

$$\texttt{find}\ y : \texttt{at}(y)\ \texttt{for}\ \texttt{send}(y).\mathbf{0}$$

- Find a neighbor and say hello

$$\texttt{find}\ y : \texttt{with}(y)\ \texttt{for}\ y :: \texttt{send}(\mathit{hello}).\mathbf{0}$$

DE MONTFORT UNIVERSITY LEICESTER

Lab Session 9 – Try it out (Search)

1. Find something I have

   $bob[\ \texttt{find}\ y : has(y)\ \texttt{for}\ \texttt{send}(y).\mathbf{0} \mid \texttt{recv}(x).\mathbf{0} \mid$
   $\quad key[\mathbf{0}]\ ]$

2. Find my current location

   $home[\ bob[$
   $\quad \texttt{find}\ y : at(y)\ \texttt{for}\ \texttt{send}(y).\mathbf{0}$
   $\quad \mid \texttt{recv}(x).\mathbf{0}\ ]$
   $]$

3. Find a neighbor and say hello

   $bob[$
   $\quad \texttt{find}\ y : with(y)\ \texttt{for}\ y :: \texttt{send}(hello, y).\mathbf{0}$
   $]$
   $\mid alice[:: \texttt{recv}(x, t).\mathbf{0}\ ]$

Specification of an ISC – revisited

- When ISC receives a text message from an IS, it looks for the IS where the recipient is located and forwards the message to that IS.
- The IS where the recipient is located then forwards the message to the recipient mobile device.
- If the recipient is not found, the message is saved in the ISC until delivery.

```
isc[
    !::recv(s,r,msg).
          find x: at(x, r) for x::send(s,r,msg).0
]
```

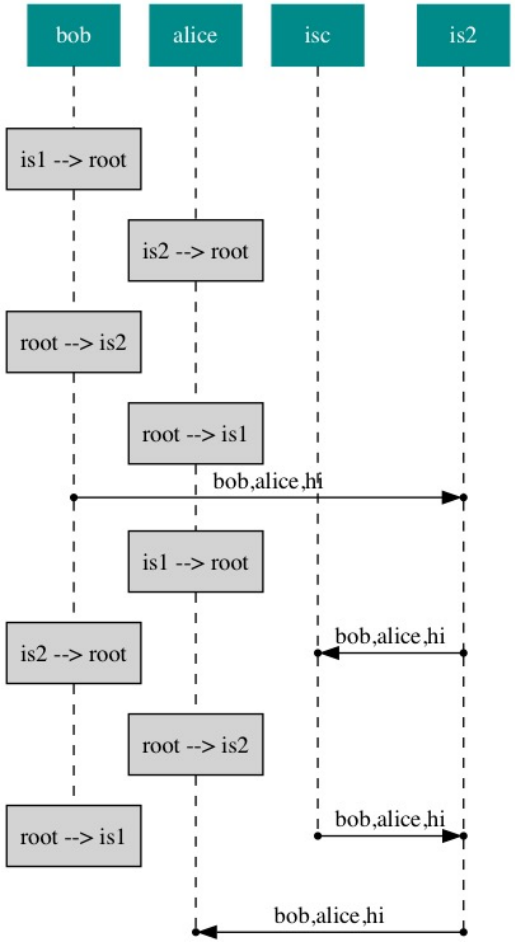
Simulation – Scenario 4

- Bob can move to IS2 and back

```
is1[    ... // as above
| bob[
    @send(bob,alice,hi).0
  |out.in is2.out.in is1.0
  ]
]
```

- Alice can move to IS1 and back

```
is2[    ... // as above
| alice[
    @recv(s,r,msg).0
  |out.in is1.out.in is2.0
  ]
]
```



Other Related Process Calculi

- π-calculus (Milner et al., 1992)
  - Milner(1992). A calculus of mobile processes. *Information and Computation* 100 (100): 1–40
  - http://en.wikipedia.org/wiki/Pi-calculus
- Ambient Calculus (Cardelli & Gordon, 2000)
  - Cardelli, L., Gordon, A. (2000). Mobile Ambients. *Theoretical Computer Science* 240, 1777–213.

DE MONTFORT UNIVERSITY LEICESTER

Other Related Process Calculi (cont'd)

- Bigraphs (Milner, 2006)
  - Milner, R. (2006). Pure bigraphs: structure and dynamics. *Information and Computation* 204(1):60–122.
  - http://www.itu.dk/research/pls/wiki/index.php/A_Brief_Introduction_To_Bigraphs
- Calculus for Context-Awareness (Zimmer 2005, Bucur 2008)
  - Zimmer, P. (2005). A Calculus for Context-awareness. *Tech. Rep.*, BRICS.
  - Bucur, D. Nielsen, M. (2008). Secure Data Flow in a Calculus for Context Awareness. *Lecture Notes in Computer Science* Vol. 5065. Springer, PP. 439–456.

DE MONTFORT UNIVERSITY LEICESTER

Learning Outcomes

In this lesson you have learned:

- Context-aware processes
- How to model context-aware IoT systems
- Other modelling formalisms

Quiz

Download the program infostation.cca from Blackboard.

- Execute the scenario 1.
- Execute the scenario 2.
- Execute the scenario 3.
- Execute the scenario 4.
- Create your own scenarios and execute them.
- Add a third infostation is3 to the system and test the new system.

DE MONTFORT UNIVERSITY LEICESTER