The Calculus of Context-aware Ambients (CCA)

How to program: Modelling with ambients

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:

- The concept of ambient
- The ambient hierarchy
- How to model with ambient

What is an ambient?

- An ambient is a conceptual representation of things, whether physical, virtual, mobile or immobile.



What is an ambient (cont'd)?

An **ambient**

- Is a conceptual representation of things
- Can be mobile
- Can sense the presence of other ambients in its environment
- Can exchange message with other ambients
- Can contain other ambients, forming an hierarchy of ambients

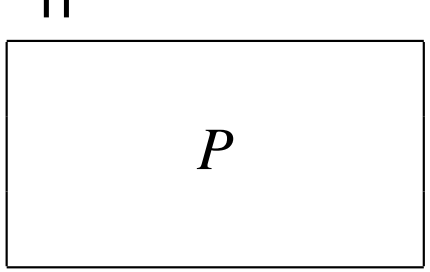
Syntax of ambient

- Textual:

$$n[P]$$

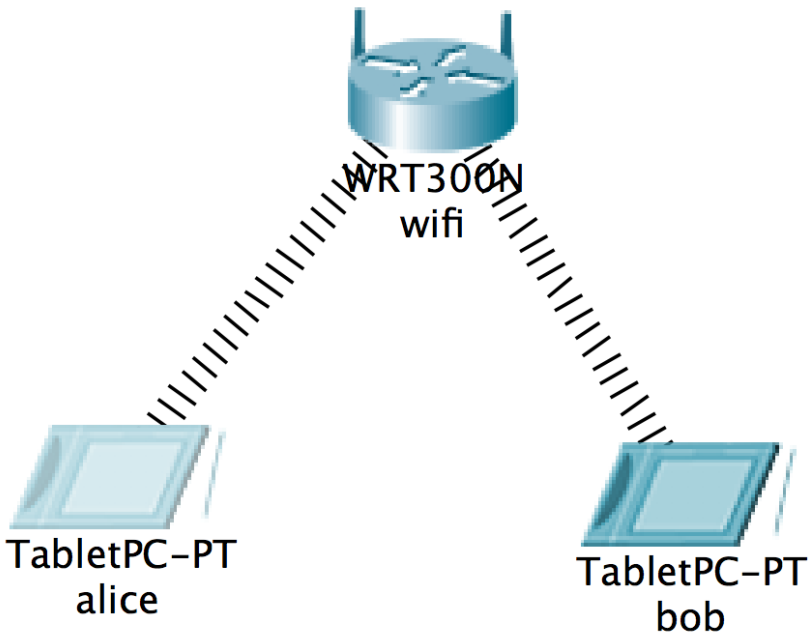
where n is the ambient's name and P is a computation (process) describing the behaviour of this ambient.

- Graphical



Example: a simple wifi access point

- The wifi access point can be modelled as an ambient named wifi.
- The boundary of the ambient denotes the range of wifi access point.
- Therefore a device is in the range of the wifi if that device is a child ambient to wifi.



Example: a simple wifi access point

- wifi ambient

```
wifi[
    !#recv(sender, recipient, msg).if
           <has(recipient)>recipient#send(sender, msg).0
          else sender#send(wifi, error).0
          fi
]
```

- Suppose Alice wants to send a hello message to Bob.

```
alice[
  in wifi.@send(alice, bob, hello).@recv(s, ack).out.0
]
```

- Bob will receive the message if he is in the range the wifi.

```
bob[
 in wifi.@recv(s, ack).@send(bob, s, thanks).out.0
]
```

Example: a simple wifi access point

- The whole system is modelled by the parallel composition of the above three ambients

  ```
  wifi[ ... ] | alice[ ... ] | bob[ ... ]
  ```

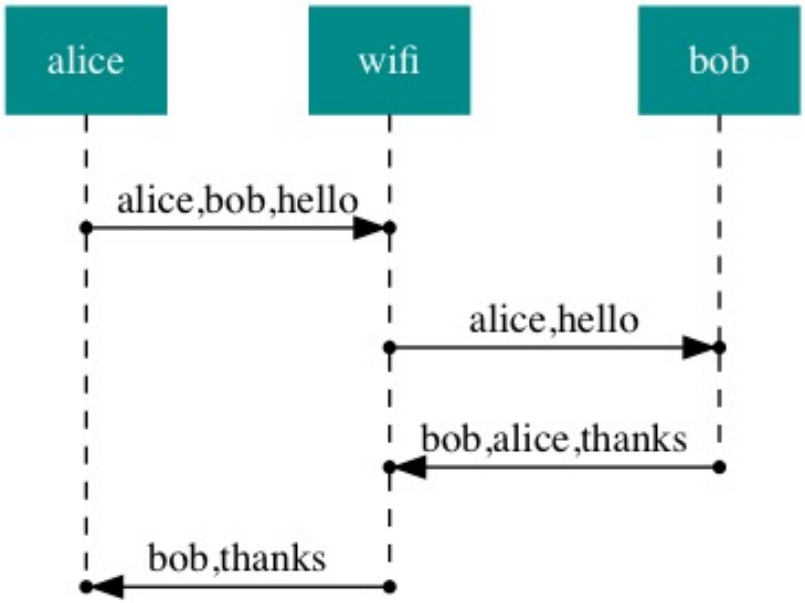
- Download the program wifi.cca from Blackboard
- Execution output

```
Execution mode: interleaving

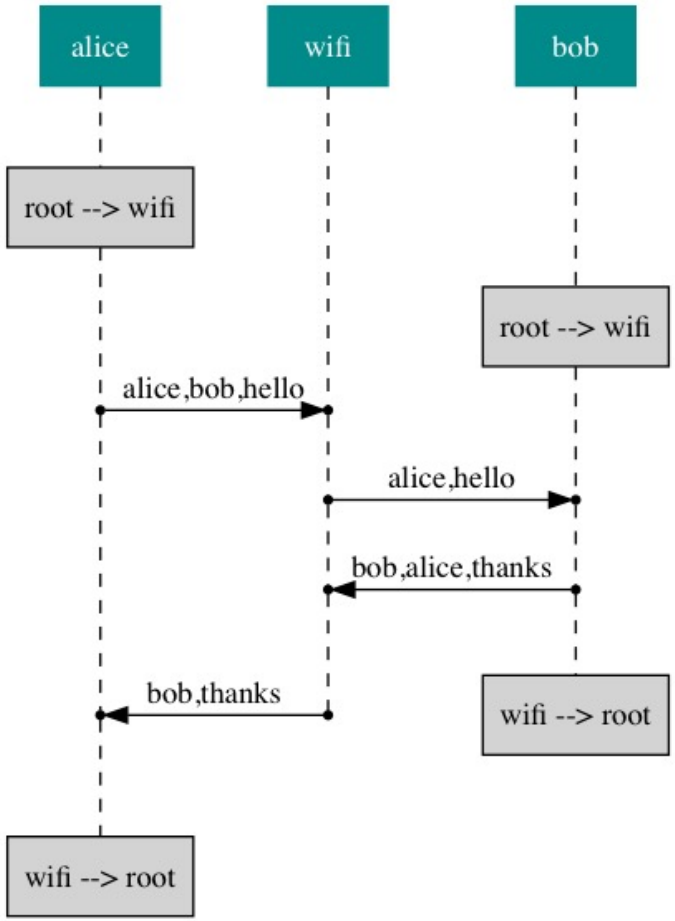
1. ---> {ambient "alice" moves into ambient "wifi"}
2. ---> {ambient "bob" moves into ambient "wifi"}
3. ---> {Child to parent: alice ===(alice,bob,hello)===> wifi}
4. ---> {Parent to child: wifi ===(alice,hello)===> bob}
5. ---> {Child to parent: bob ===(bob,alice,thanks)===> wifi}
6. ---> {Parent to child: wifi ===(bob,thanks)===> alice}
7. ---> {ambient "alice" moves out of ambient "wifi"}
8. ---> {ambient "bob" moves out of ambient "wifi"}
```

DE MONTFORT UNIVERSITY LEICESTER

Example: a simple wifi access point



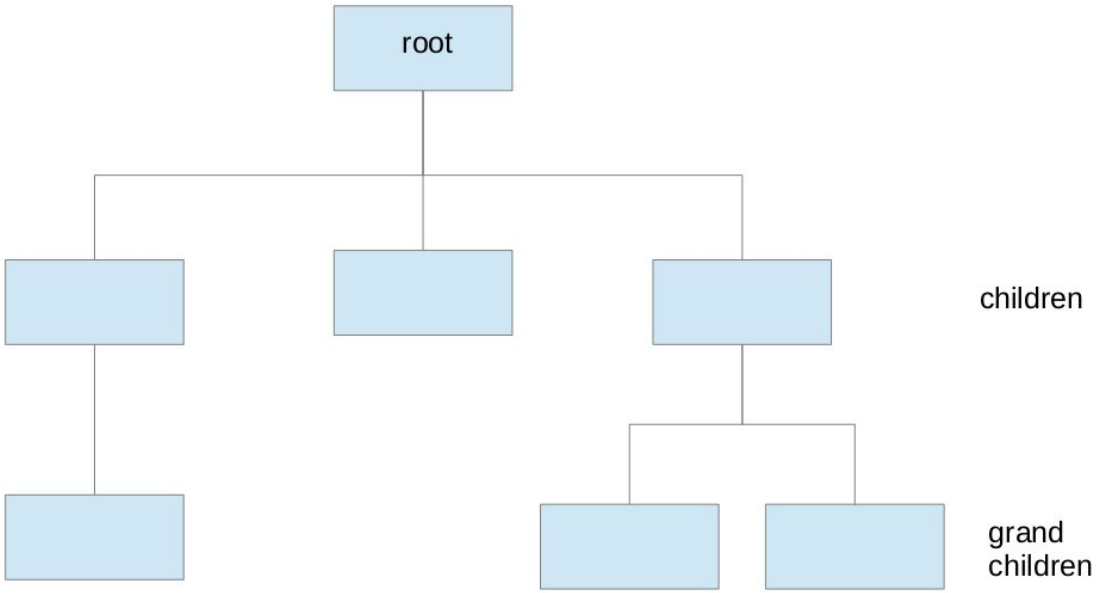
Communication graph



Behaviour graph

Ambient hierarchy

- In `ccaPL`, the top ambient is called the **root** ambient.
- Any ambient is a descendant of the **root** ambient.



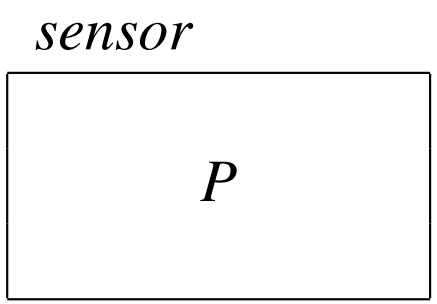
Modelling with ambients: a sensor node

- A sensor node can be modelled as an ambient:

$$sensor[P]$$

where P describes the behaviour of the sensor node.

- Graphically it looks like this:

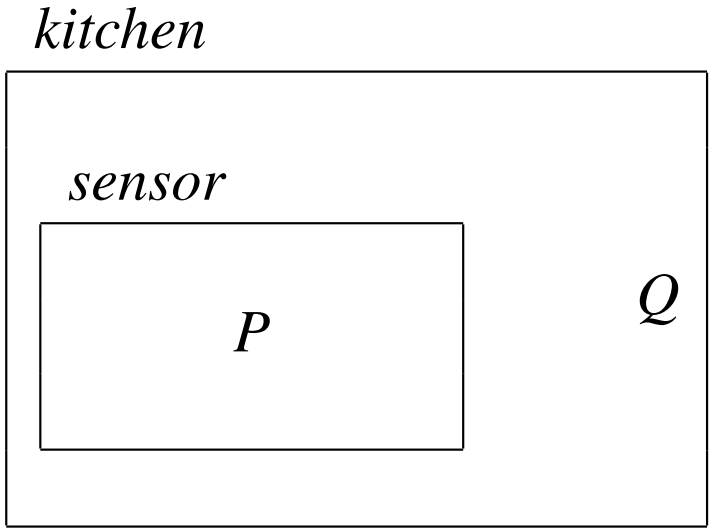


Modelling with ambients: location

- The location, $kitchen$, of a sensor node can also be modelled as an ambient:

$$kitchen[sensor[P] \mid Q]$$

  Q here represents the other things in the kitchen.
- Graphically it looks like this:



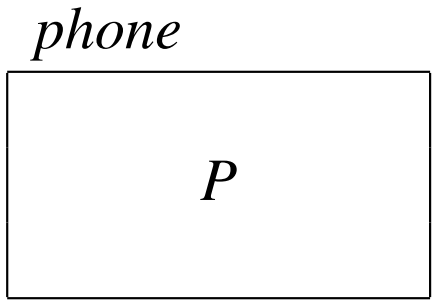
Modelling with ambients: smart devices and users

- A smart mobile phone **device** can be modelled as an ambient:

$$phone[P]$$

where P encompasses the functionality of the smart phone.

- Graphically it looks like this:



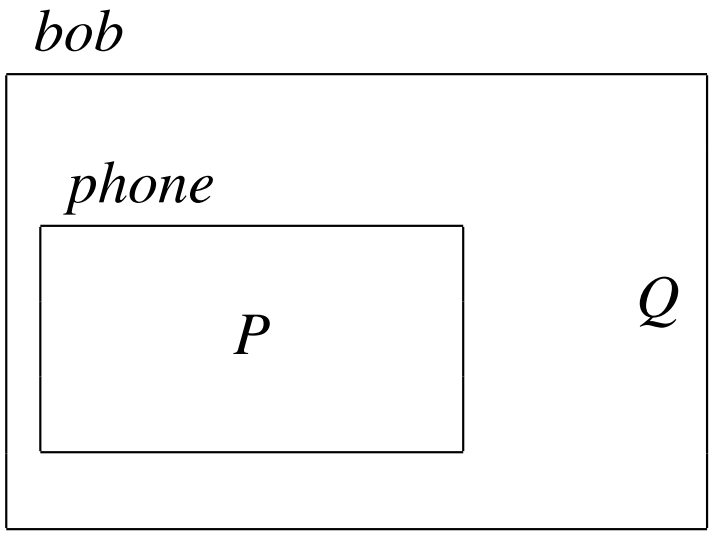
Modelling with ambients: smart devices and users (1)

- the **user**, bob, carrying the phone can also be modelled as an ambient:

$$bob[phone[P] \mid Q]$$

  Q here characterises the behaviour of the user, bob.

- Graphically it looks like this:

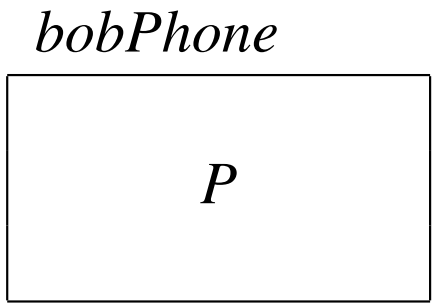


Modelling with ambients: smart devices and users (2)

- the **user** can also be modelled by naming the device after the user:

$$bobPhone[P]$$

- Graphically it looks like this:

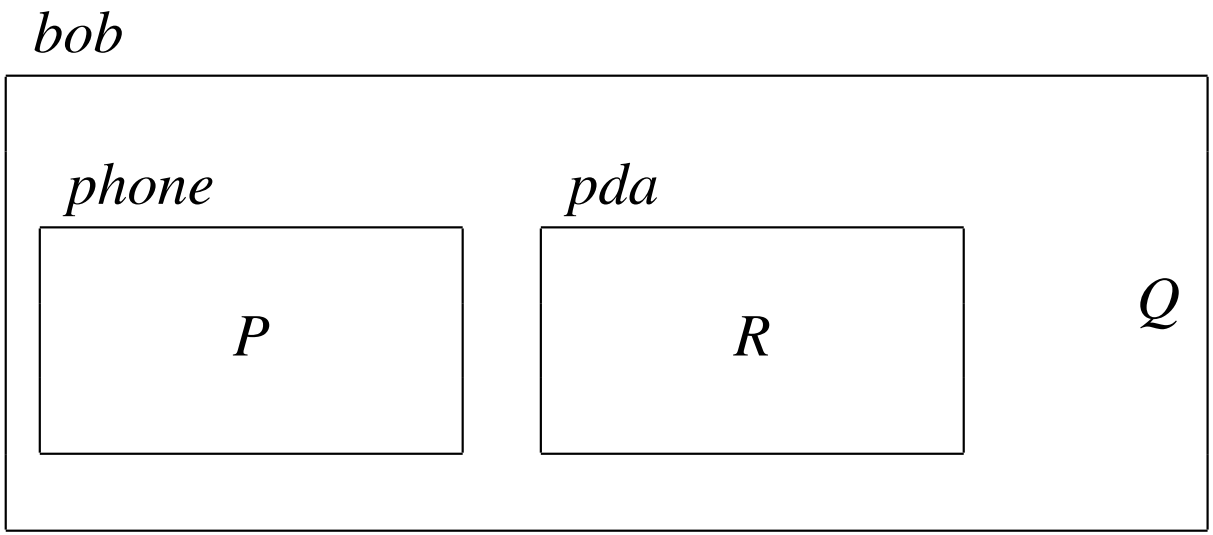


Modelling with ambients: smart devices and users

- Bob might carry more than one device: a phone and a PDA

$$bob[phone[P] \mid pda[R] \mid Q]$$

- Graphically it looks like this:

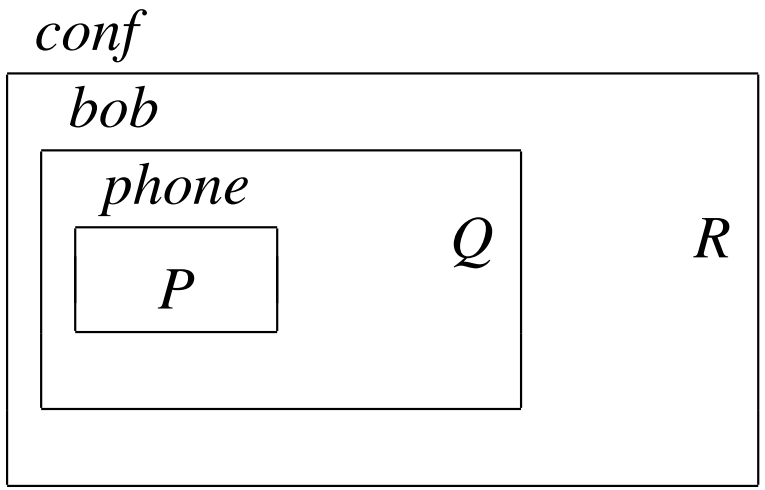


Modelling with ambients: user's location

- The **location** of the user can be modelled as an ambient
- E.g. Bob is carrying a phone at the conference room

$$conf[bob[phone[P] \mid Q] \mid R]$$

- Graphically it looks like this:

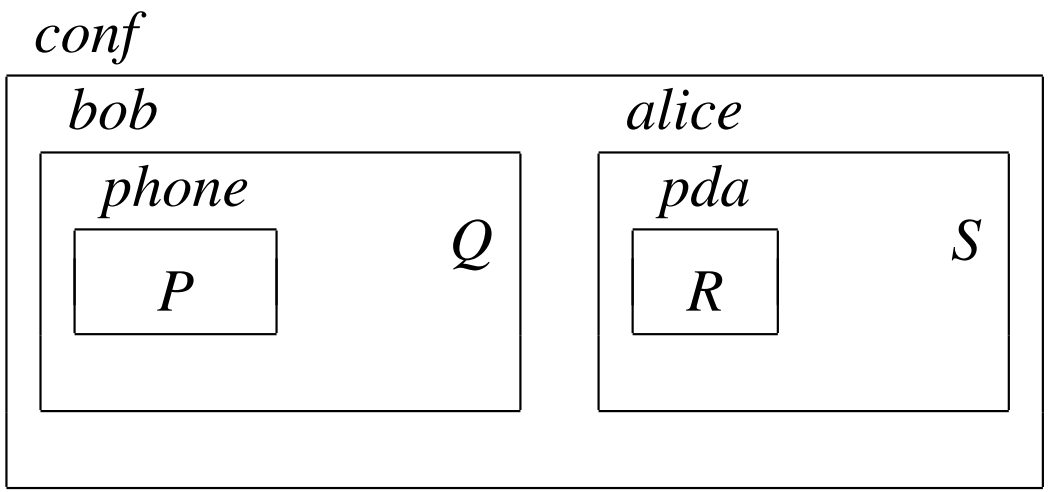


Modelling with ambients: user is with an other person

- Bob **is with** Alice at the conference room, Bob carrying a mobile phone while Alice holding a pda

$$conf[bob[phone[P] \mid Q] \mid alice[pda[R] \mid S]]$$

- Graphically it looks like this:

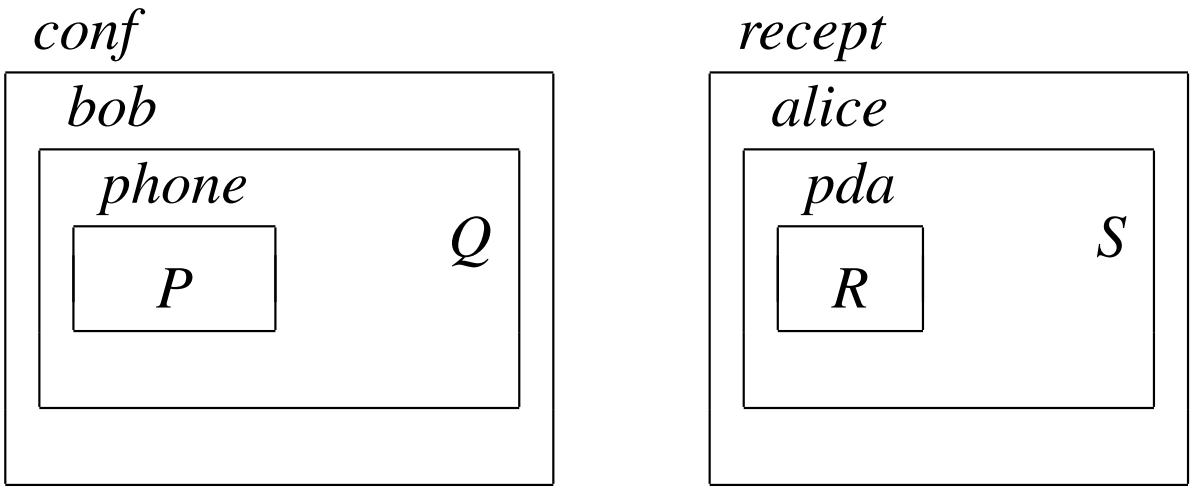


Modelling with ambients: user/device environment

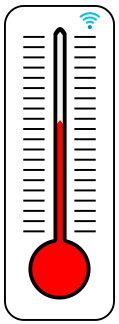
- the conference room **is next to** the reception room
- Bob **is at** the conference room, **carrying** a mobile phone and Alice **is at** the reception room, **carrying** a pda

$$conf[bob[phone[P] \mid Q]] \mid recept[alice[pda[R] \mid S]]$$

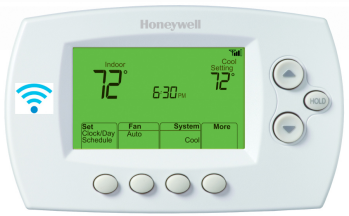
- Graphically it looks like this:



Example: a Simple HVAC System









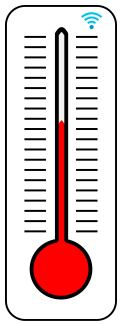


T_Sensor Thermostat Heating

- Temperature sensor (assumptions)
  - if heating on, the temperature increases.
  - if heating off, the tempearture decreases.
- Thermostat (requirements)
  - if heating on and temp above 21^{o}C, turn the heating off.
  - if heating off and temp below 15^{o}C, turn the heating on.

Example: a Simple HVAC System

- We assume the context-expression state(device, value) tells if a device is in a given state. Its formal definition is given later. The temperature sensor is modelled below.
- The initial temperature is 18^{o}C.



$$
\begin{array}{l}
T_Sensor \\
\quad !\mathtt{recv}(v).\mathtt{if} \\
\qquad < true > \ Thermostat :: \mathtt{send}(v).\mathtt{send}(v).\mathbf{0} \\
\qquad < state(heating, on) > \ \mathtt{let}\ t = v + 2\ \mathtt{in}\ \mathtt{send}(t).\mathbf{0} \\
\qquad < state(heating, off) > \ \mathtt{let}\ t = v - 1\ \mathtt{in}\ \mathtt{send}(t).\mathbf{0} \\
\quad \mathtt{fi} \mid \mathtt{send}(18).\mathbf{0}
\end{array}
$$

Example: a Simple HVAC System

- $state(device, value) =$
  $\{\texttt{somewhere}\ (device[value[0] \mid \texttt{true}] \mid \texttt{true})\}$
- the heating device is an actuator. Initially the heating is off.



$heating$

$$
\begin{array}{l}
\quad !\ Thermostat :: \texttt{recv}(v).\texttt{if} \\
\qquad\quad < v = 0 > \ \texttt{del}\ on.off[0] \\
\qquad\quad < v = 1 > \ \texttt{del}\ off.on[0] \\
\qquad \texttt{fi} \\
\mid off[0]
\end{array}
$$

Example: a Simple HVAC System

- Thermostat device is a controller



$$
\begin{array}{l}
\mathit{Thermostat} \\
\quad !\texttt{recv}().\mathit{T_sensor} :: \texttt{recv}(t).\texttt{if} \\
\qquad < t > 21 > \ \mathit{heating} :: \texttt{send}(0).\texttt{send}().\mathbf{0} \\
\qquad < t < 15 > \ \mathit{heating} :: \texttt{send}(1).\texttt{send}().\mathbf{0} \\
\quad \texttt{else send}().\mathbf{0} \\
\quad \texttt{fi} \mid \texttt{send}().\mathbf{0}
\end{array}
$$

Example: a Simple HVAC System

T_sensor

$$!\texttt{recv}(v).\texttt{if}\ <\mathit{true}>\ \mathit{Thermostat} :: \texttt{send}(v).\texttt{send}(v).\mathbf{0}\ <\mathit{state}(\mathit{heating},\mathit{on})>\ \texttt{let}\ t = v+2\ \texttt{in}\ \texttt{send}(t).\mathbf{0}\ <\mathit{state}(\mathit{heating},\mathit{off})>\ \texttt{let}\ t = v-1\ \texttt{in}\ \texttt{send}(t).\mathbf{0}\ \texttt{fi} \mid \texttt{send}(18).\mathbf{0}$$

Thermostat

$$!\texttt{recv}().T_sensor :: \texttt{recv}(t).\texttt{if}\ <t>21>\ \mathit{heating} :: \texttt{send}(0).\texttt{send}().\mathbf{0}\ <t<15>\ \mathit{heating} :: \texttt{send}(1).\texttt{send}().\mathbf{0}\ \texttt{else send}().\mathbf{0}\ \texttt{fi} \mid \texttt{send}().\mathbf{0}$$

heating

$$!\ \mathit{Thermostat} :: \texttt{recv}(v).\texttt{if}\ <v=0>\ \texttt{del}\ \mathit{on}.\mathit{off}[0]\ <v=1>\ \texttt{del}\ \mathit{off}.\mathit{on}[0]\ \texttt{fi} \mid \mathit{off}[0]$$

Example: a Simple HVAC System

- The whole system is modelled by the parallel composition of the above three ambients

  `T_sensor[ ... ] | Thermostat[ ... ] | heating[ ... ]`

- Download the program hvac.cca from Blackboard
- Textual output.

Examples — -bash ▸ gedit — 107×52

```
GWHXTECHA05526S:Examples fsiewe00$ java -jar ccaPL501.jar -g hvac.cca
CCA Parser Version 5.01:  Reading from file hvac.cca . . .
Please report any error messages to fsiewe@dmu.ac.uk or fsiewe@yahoo.fr . . .
CCA Parser Version 5.01:  CCA program parsed successfully.

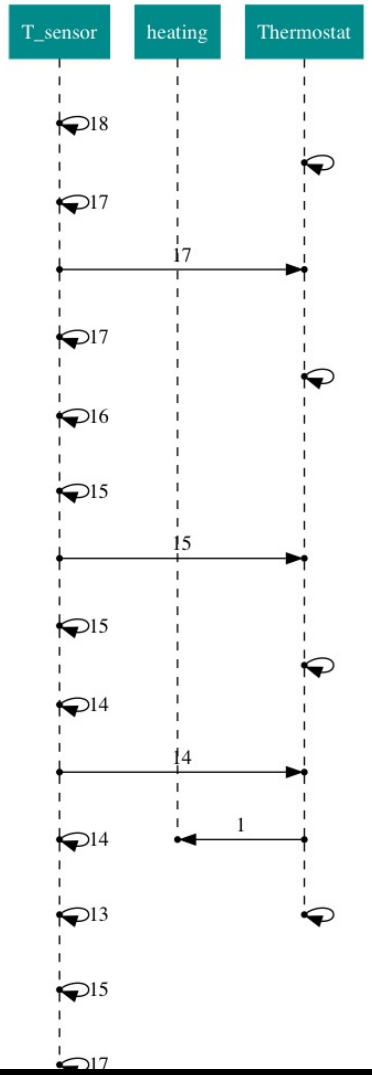
Execution mode: interleaving


1. --->  {Local: T_sensor ===(18)===> T_sensor}
2. --->  {Local: Thermostat ===()===> Thermostat}
3. --->  {Local: T_sensor ===(17)===> T_sensor}
4. --->  {Sibling to sibling: T_sensor ===(17)===> Thermostat}
5. --->  {Local: T_sensor ===(17)===> T_sensor}
6. --->  {Local: Thermostat ===()===> Thermostat}
7. --->  {Local: T_sensor ===(16)===> T_sensor}
8. --->  {Sibling to sibling: T_sensor ===(16)===> Thermostat}
9. --->  {Local: T_sensor ===(16)===> T_sensor}
10. --->  {Local: Thermostat ===()===> Thermostat}
11. --->  {Local: T_sensor ===(15)===> T_sensor}
```

DE MONTFORT UNIVERSITY LEICESTER

Example: a Simple HVAC System

- Communication graph.



Summary

Learning Outcomes

In this lesson you have learned:

- The concept of ambient
- The ambient hierarchy
- How to model with ambient

Quiz

- What is an ambient?
- How is an ambient represented in CCA?
- How do ambients communicate?
- Answer by "true" or "false"
  - Ambient can be mobile.
  - An ambient can contain another ambient.
- Give the graphical representation of these ambients
  1. $table[P] \mid chair[Q]$
  2. $room[tv[P] \mid john[Q] \mid jack[R]]$
  3. $house[living[P] \mid dining[john[Q]] \mid bedroom[jack[R]]]$
  4. $wifi[P] \mid laptop[Q] \mid printer[R]$
- Ambient m is a child ambient of n and the sibling of the ambient w. Give the graphical and textual representations of these ambients.

DE MONTFORT UNIVERSITY LEICESTER

References

1. http://www.cse.dmu.ac.uk/~fsiewe/CCA/cca.html
2. F. Siewe et al. (2009). CCA: a Calculus of Context-aware Ambients. *IEEE 23rd International Conference on Advanced Information Networking and Applications (AINA-09)*, University of Bradford, Bradford, UK, May 26-29.
3. F. Siewe et al. (2010). The Calculus of Context-aware Ambients. *Journal of Computer and Systems Sciences*, 77(4), 597-620.
4. F. Siewe et al. (2016). Analysing Use Case Diagrams in a Calculus of Context-aware Ambients. *International Journal of Intelligent Computing Research, Infonomics Society*, 7(1), 655-662.
5. F. Siewe et al. (2016). Privacy Protection by Typing in Ubiquitous Computing Systems. *Journal of Systems and Software*, 120, 133-153.

DE MONTFORT UNIVERSITY LEICESTER