The Calculus of Context-aware Ambients (CCA)

How to program: Context-expressions

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

Overview

Learning Outcomes

By the end of this lesson you should know:

- The context model in `ccaPL`
- The syntax and semantics of context-expressions

What is Context?

- Dey (1993):

  Context *is any* ***information*** *that can be used to characterise the* ***situation*** *of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.*

- Important aspects of context include:
  - user location
  - who the user is with
  - nearby people or resources
  - user preferences

Context Model

- In `ccaPL`, a process is described as a hierarchy of nested ambients.
- So, the context of each individual (sub-)process corresponds to the surrounding of that process in the hierarchy.
- This is obtained by replacing the (sub-)process by a ‘hole’ in the hierarchy.
- A hole is represented by the symbol $\odot$ and can be thought of as a place holder.

Examples of Context (cont'd)

- Bob is carrying a mobile device (phone) is represented by the process

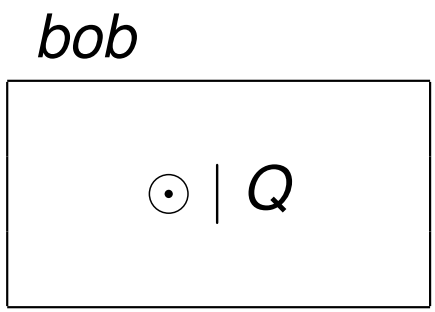
$$bob[phone[P] \mid Q]$$

where P and Q are processes.

- The context of the mobile device carried by Bob is:

$$bob[\odot \mid Q]$$

- Graphically it looks like this:



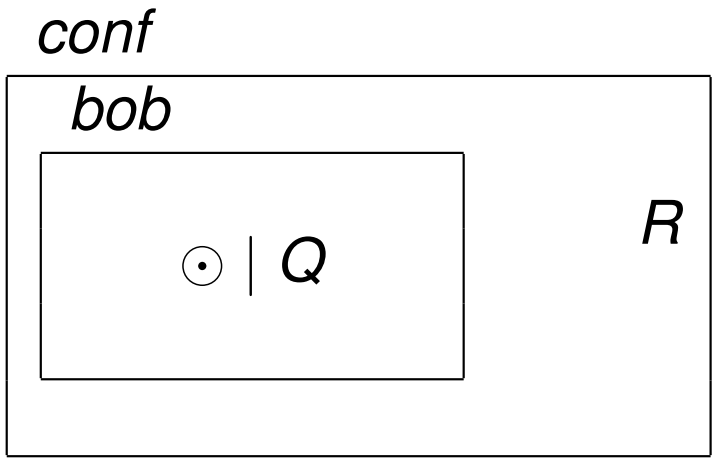
Examples of Context (cont'd)

- Context of a mobile device carried by Bob at the conference room:

$$conf[bob[\odot \mid Q] \mid R]$$

where Q and R are processes.

- Graphically it looks like this:

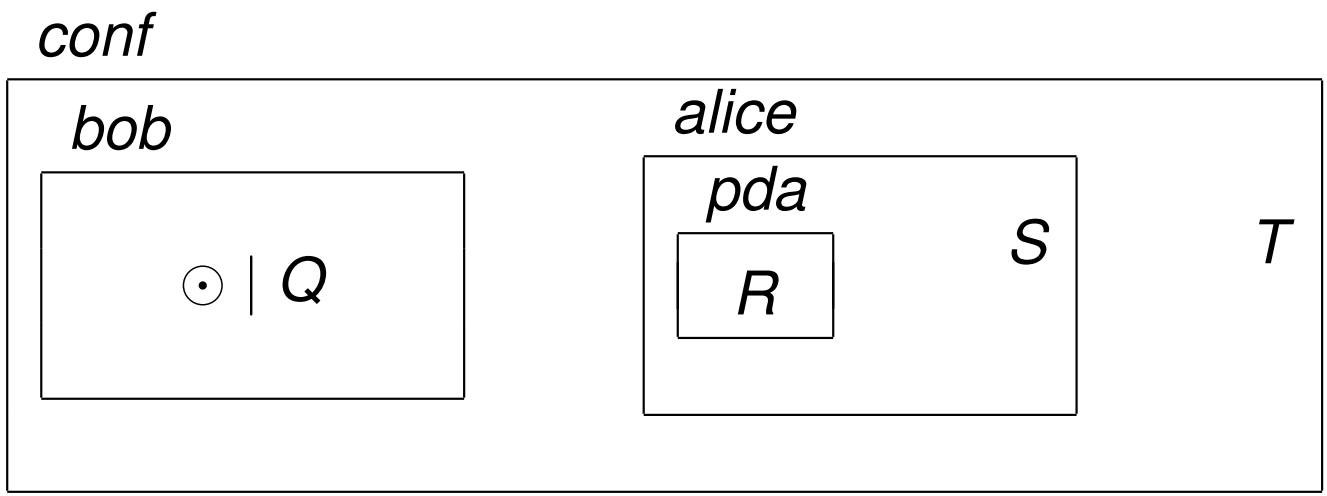


Examples of Context (cont'd)

- Context of a smart device carried by Bob who is with Alice at conference, Alice carrying a pda:

$$conf[bob[\odot \mid Q] \mid alice[pda[R] \mid S] \mid T]$$

- Graphically it looks like this:



Context Model: Syntax

In CCA, a context is described by the following syntax:

C	::=		**Context**
		0	nil
		$\odot$	hole
		$n[C]$	location
		$C \mid P$	parallel composition
		`new` n C	restriction

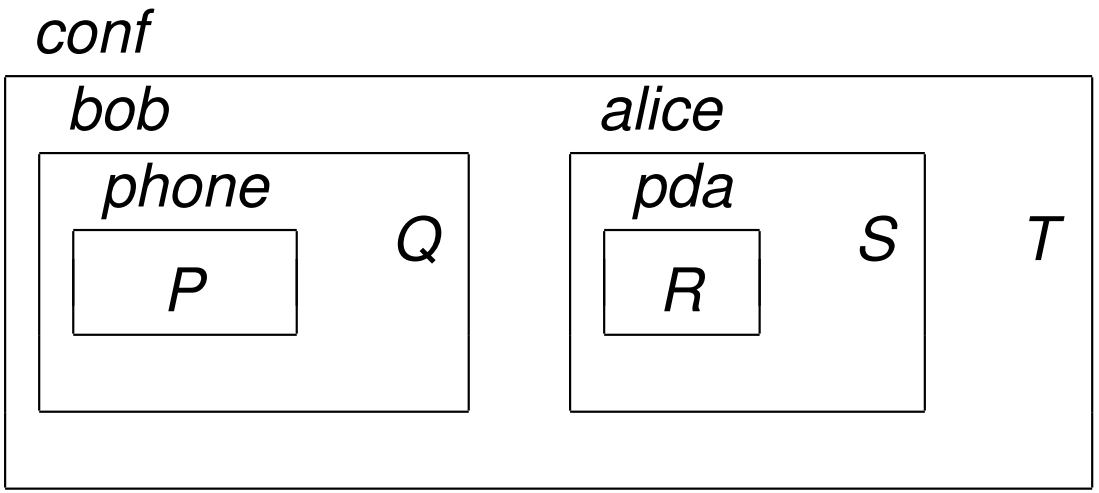
where the symbol C stands for context (environment), n ranges over names and P ranges over processes.

Context Model: Syntax (cont'd)

- **0** is the empty context, also called the **nil** context. It contains no context information.
- $\odot$ denotes the position of a process in that process' context. This is a special context called the **hole** context.
- `new` n C means that the scope of the name n is limited to the context C.
- $n[C]$ means that the internal environment of the ambient n is described by the context C.
- $C \mid P$ says that the process P runs in parallel with the context C.

Context Model: Exercise

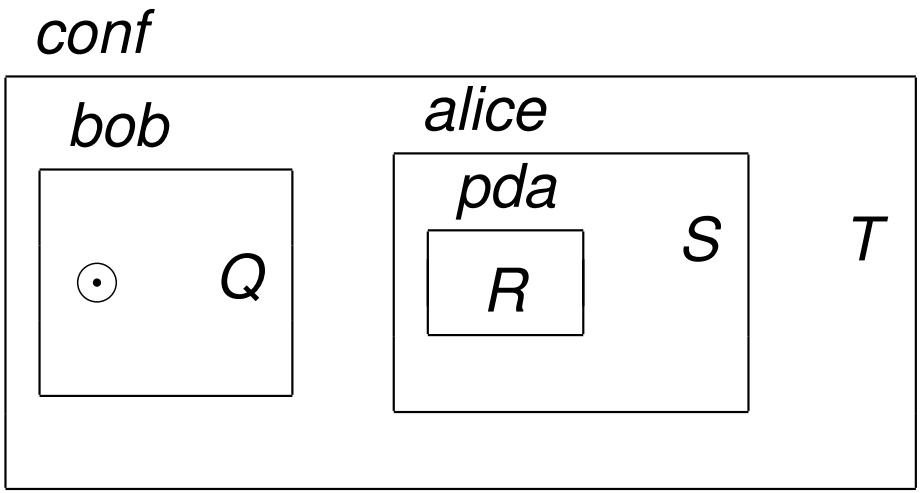
- Consider the following process:



- What is the context of the ambient *phone* in that process?
- What is the context of the ambient *alice* in that process?
- What is the context of the process T in that process?

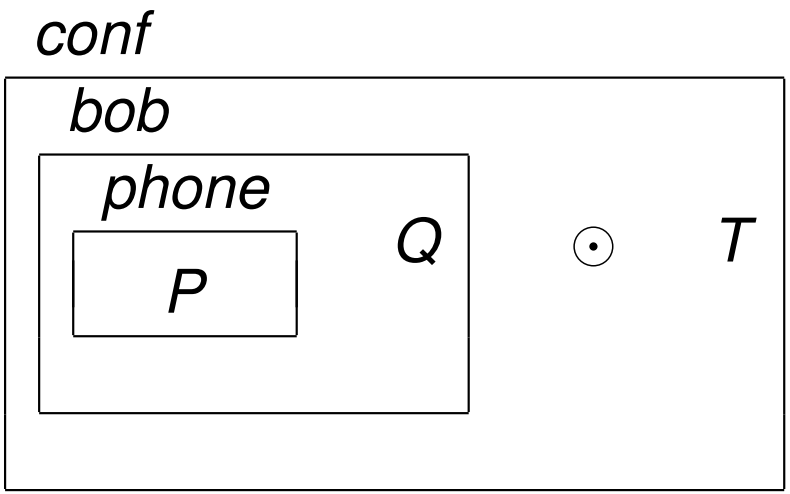
Context Model: Exercise (Cont'd)

- Context of the ambient *phone*:



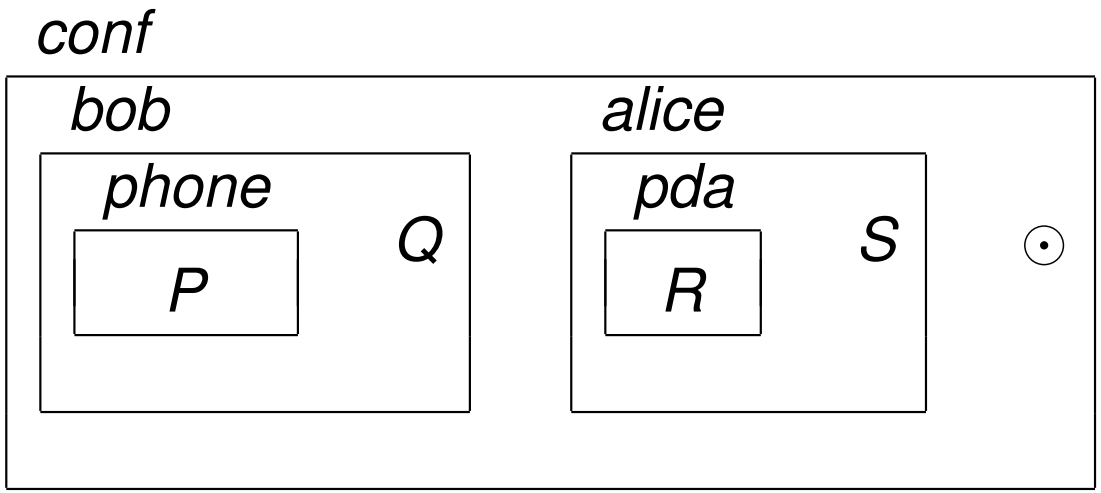
Context Model: Exercise (Cont'd)

- Context of the ambient *alice*:



Context Model: Exercise (Cont'd)

- Context of the process T:



Syntax of Context Expressions

Context expressions are described by the following language:

κ ::=	**Context Expressions**
0	empty context
`true`	true
`false`	false
$n = m$	name match
`this`	hole
`not` κ	negation
$\kappa_1 \mid \kappa_2$	parallel composition
κ_1 `and` κ_2	conjunction
κ_1 `or` κ_2	disjunction
$n[\kappa]$	location
`next` κ	spatial next modality
`somewhere` κ	somewhere modality

Context Expression (**0**)

The context-expression **0** holds only for the empty context **0**.

Example

0 does not hold for any following contexts:

- $\odot$
- $n[\odot]$
- $\texttt{send}().\mathbf{0}$
- $\odot \mid n[\texttt{send}().\mathbf{0}]$

Context Expression (`true`, `false`)

- `true` holds for all contexts.
- `false` holds for no contexts.
- `false` is equivalent to `not true`

Example

`true` holds for the following contexts:

- $\odot$
- $n[\odot]$
- $\mathbf{0}$
- `send`().$\mathbf{0}$
- $\odot \mid n[$`send`().$\mathbf{0}]$

Context Expression ($n = m$)

$n = m$ holds for a context if the names n and m are lexically identical.

Example

- $2 = 2$ evaluates to `true` for all contexts.
- $n = n$ evaluates to `true` for all contexts.
- $2 = 3$ evaluates to `false` for all contexts.
- *alice* $=$ *bob* evaluates to `false` for all contexts.

In addition to the match operator '=', the following predicates can be used:

- '$<$': less than; e.g. $n < m$
- '$<=$': less than or equal ; e.g. $n <= m$
- '$>$': greater than; e.g. $n > m$
- '$>=$': greater than or equal; e.g. $n >= m$

DE MONTFORT UNIVERSITY LEICESTER

Context Expressions (`this`)

`this` holds solely for the hole context $\odot$.

Example

- `this` holds for the context $\odot$
- `this` does not hold for the context $phone[\odot]$
- `this` does not hold for the context $bob[phone[\odot] \mid pda[R] \mid Q]$

Context Expressions ($n[\kappa]$)

$n[\kappa]$ holds for a context if that context is of the form $n[C]$ such that κ holds for the context C.

Example

- *phone*[`this`] holds for the context *phone*[$\odot$]
- *phone*[`true`] holds for the context *phone*[$\odot$]
- *bob*[*phone*[`this`]] holds for the context *bob*[*phone*[$\odot$]]
- *pda*[`this`] does not hold for the context *phone*[$\odot$]
- *phone*[`this`] does not hold for the context $\odot$

Context Expressions ($\kappa_1 \mid \kappa_2$)

$\kappa_1 \mid \kappa_2$ holds for a context C if that context is of the form $C = C_1 \mid C_2$ such that κ_1 holds for C_1 and κ_2 holds for C_2.

Example

- `this` | `true` holds for the context $\odot$ | *pda*[*R*]
- *phone*[`this`] | *pda*[`true`] holds for the context *phone*[$\odot$] | *pda*[*R*]
- *phone*[`this`] | `true` holds for the context *phone*[$\odot$] | *pda*[*R*]
- *bob*[*phone*[`this`] | *pda*[`true`] | `true`] holds for the context *bob*[*phone*[$\odot$] | *pda*[*R*] | *Q*]
- *bob*[`true` | *pda*[`true`] | `true`] holds for the context *bob*[*phone*[$\odot$] | *pda*[*R*] | *Q*]
- `this` | *phone*[`true`] does not hold for the context $\odot$ | *pda*[*R*]

DE MONTFORT UNIVERSITY LEICESTER

Context Expressions (κ_1 and κ_2)

κ_1 and κ_2 holds for a context if κ_1 holds for that context and κ_2 also holds for that context.

Example

- *phone*[this] and *phone*[true] holds for the context *phone*[$\odot$]
- *bob*[*phone*[this] | true] and *bob*[true] holds for the context *bob*[*phone*[$\odot$] | *R*]
- *phone*[this] and *phone*[true] does not hold for the context *phone*[**0**]
- *bob*[*phone*[this] | true] and *bob*[true] does not hold for the context *bob*[*pda*[$\odot$] | *R*]

Context Expressions (κ_1 or κ_2)

κ_1 or κ_2 holds for a context if κ_1 holds for that context or κ_2 holds for that context.

Example

- *phone*[this] or *phone*[true] holds for the context *phone*[$\odot$]
- *bob*[*phone*[this] | true] or *bob*[true] holds for the context *bob*[*phone*[$\odot$] | *R*]
- *phone*[this] or *phone*[true] holds for the context *phone*[**0**]
- *bob*[*phone*[this] | true] or *bob*[true] holds for the context *bob*[*pda*[$\odot$] | *R*]
- *phone*[this] or *phone*[true] does not hold for the context $\odot$
- *bob*[*phone*[this] | true] or *bob*[true] does not hold for the context *alice*[*phone*[$\odot$] | *R*]

DE MONTFORT UNIVERSITY LEICESTER

Context Expressions (next κ)

next κ holds for a context C if that context is of the form $C = n[C']$ and κ holds for C'.
The operator next is called the spatial next modality.

Example

- next this holds for the context *phone*[$\odot$]
- next (*phone*[true] | this) holds for the context *bob*[*phone*[*R*] | $\odot$]
- next (true | this) holds for the context *bob*[*phone*[*R*] | $\odot$]
- next ((next true) | this) holds for the context *bob*[*phone*[*R*] | $\odot$]
- next this dose not hold for the context *bob*[*phone*[*R*] | $\odot$]
- next this does not hold for the context *bob*[*phone*[$\odot$]]
- next next this holds for the context *bob*[*phone*[$\odot$]]

DE MONTFORT UNIVERSITY LEICESTER

Context Expression (somewhere κ)

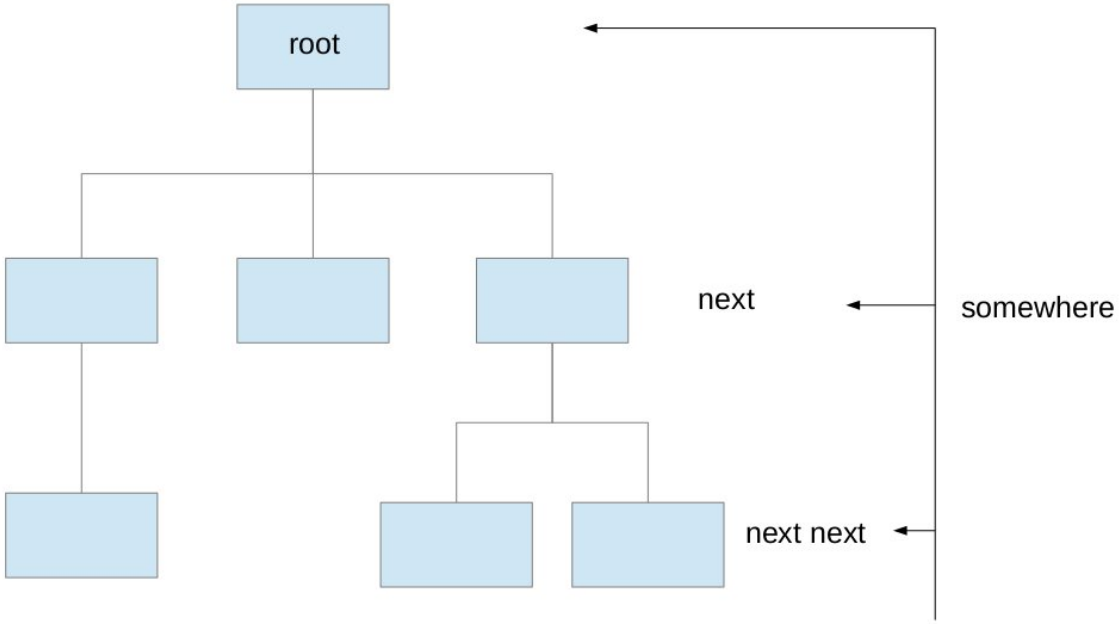
somewhere κ holds for a context if there exists somewhere in that context a sub-context for which κ holds.

Example

- somewhere this holds for the context $\odot$
- somewhere this holds for the context *phone*[$\odot$]
- somewhere this holds for the context *bob*[*phone*[$\odot$] | *R*]
- somewhere this holds for the context *conf*[*bob*[*phone*[$\odot$] | *R*] | *Q*]
- somewhere this does not hold for the context **0**
- somewhere this does not hold for the context *phone*[**0**]
- somewhere this does not hold for the context *conf*[*bob*[*phone*[**0**]]]

Context Expression (somewhere κ)

The context expression (somewhere κ) holds if the context expression κ holds at any level of the ambient hierarchy.



Context Expression (`not` κ)

`not` κ holds for a context if κ does not hold for that context.

Example

- `not this` holds for the context *phone*[$\odot$]
- `not next true` holds for the context $\odot$
- `not this` does not hold for the context $\odot$
- `not next true` does not hold for the context *phone*[$\odot$]

Formalising Common Context Predicates in CCA

- Context expressions (CEs) are commonly used inside an ambient, in the form:

$$\lambda[< \kappa > M.P \mid Q]$$

  where
  - κ is a CE
  - M is a capability, and
  - P and Q are processes.
- We say that λ is the ambient evaluating the context expression κ.
- In the following slides, we assume that CEs are used in this form, i.e. inside some ambient that we refer to as λ.

The Predicate has

- has(n) holds if the ambient λ evaluating the expression a child ambient named n:

$$\texttt{has}(n) = \texttt{somewhere}\ (\texttt{this} \mid n[\texttt{true}] \mid \texttt{true})$$

- has(n, m)holds if m is a child ambient of the ambient n:

$$\texttt{has}(n, m) = \texttt{somewhere}\ (n[m[\texttt{true}] \mid \texttt{true}] \mid \texttt{true})$$

Example

- has(pda) holds for the context $conf[bob[\odot \mid pda[\mathbf{0}]]]$.
- has($building, conf$) holds for the context $building[recept[\mathbf{0}] \mid conf[bob[\odot \mid pda[\mathbf{0}]]]]$.

The Predicat at

- at(n) holds for a context if the ambient λ evaluating the expression is located in the ambient n:

$$\texttt{at}(n) = \texttt{somewhere}\ (n[\texttt{next}\ (\texttt{this} \mid \texttt{true})] \mid \texttt{true})$$

- at(n, m) holds for a context if the ambient λ evaluating the expression is located in a root ambient named n (same as has(n, m)):

$$\texttt{at}(n, m) = \texttt{somewhere}\ (n[m[\texttt{true}] \mid \texttt{true}] \mid \texttt{true})$$

Example

- at(*conf*) holds for the context $conf[bob[\odot \mid pda[\mathbf{0}]]]$.
- at(*conf*) holds for the context $building[recept[\mathbf{0}] \mid conf[bob[\odot \mid pda[\mathbf{0}]]]]$.

The Predicate `with`

- `with`(n) holds for a context if λ and n are siblings in that context:

$$\texttt{with}(n) = \texttt{somewhere}\ (n[\texttt{true}] \mid \texttt{next}\ (\texttt{this} \mid \texttt{true}) \mid \texttt{true})$$

- `with`(n, m) holds for a context if n and m are siblings in that context:

$$\texttt{with}(n, m) = \texttt{somewhere}\ (n[\texttt{true}] \mid m[\texttt{true}] \mid \texttt{true})$$

Example

- `with`(*alice*) holds for the context $alice[\mathbf{0}] \mid bob[\odot \mid pda[\mathbf{0}]]$.
- `with`(*bob*, *alice*) holds for the context $conf[alice[\mathbf{0}] \mid bob[\odot \mid pda[\mathbf{0}]]]$.

Other Predicates

- Note that the above predicates are not implemented in ccaPL. If you want to use any of them, you will have to define them in the declaration section of your program.
- You can define your own predicates in a similar manner.

Summary

Learning Outcomes

In this lesson you have learned:

- The context model in `ccaPL`
- The syntax and semantics of context-expressions

Quiz

Answer by "true" or "false"

- The hole $\odot$ is a process.
- The keyword this denotes a process in ccaPL.
- The keyword this denotes a context ccaPL.
- The hole $\odot$ is a context.
- The keyword this denotes a context-expression.
- The hole $\odot$ is a context-expression.
- The root ambient is the ancestor of all ambient.