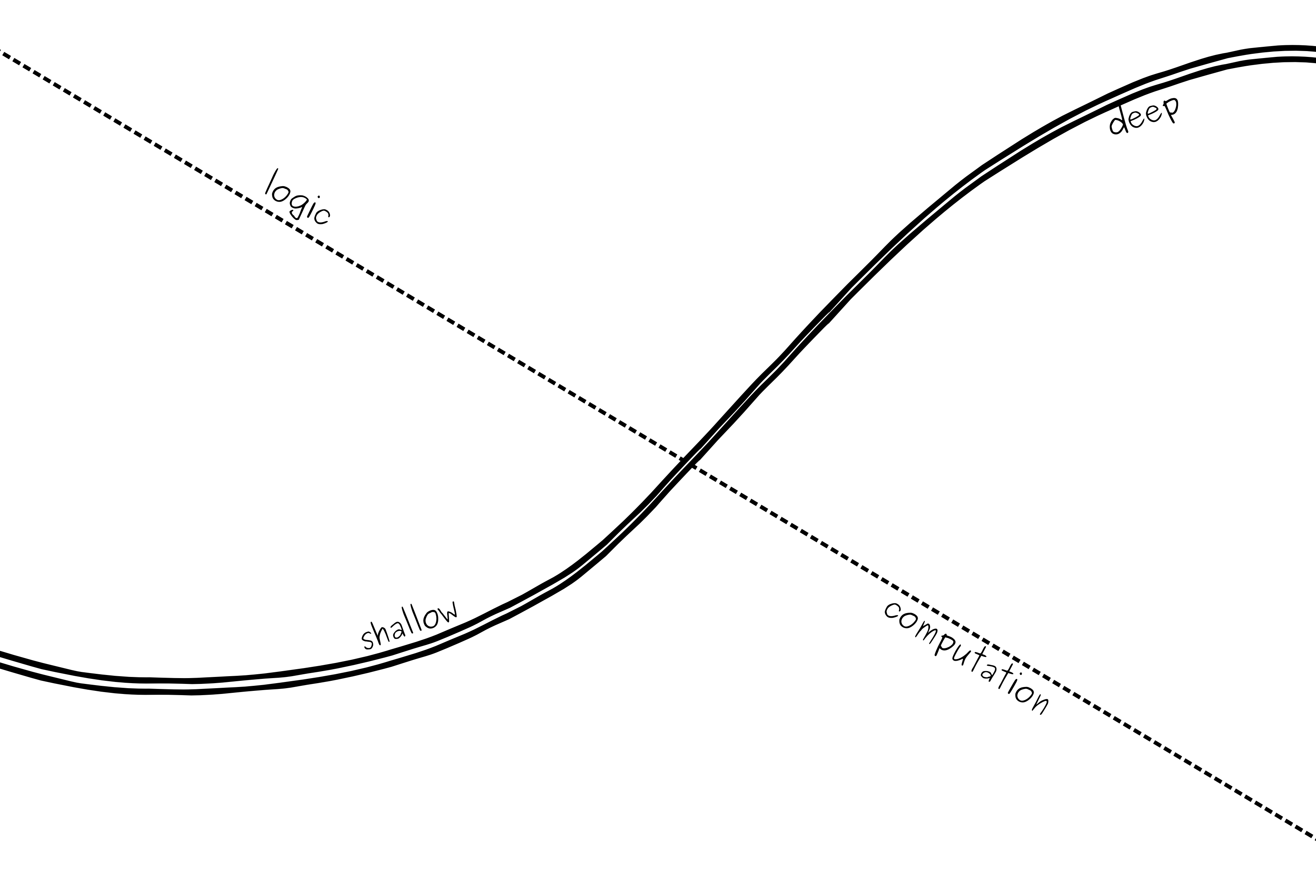
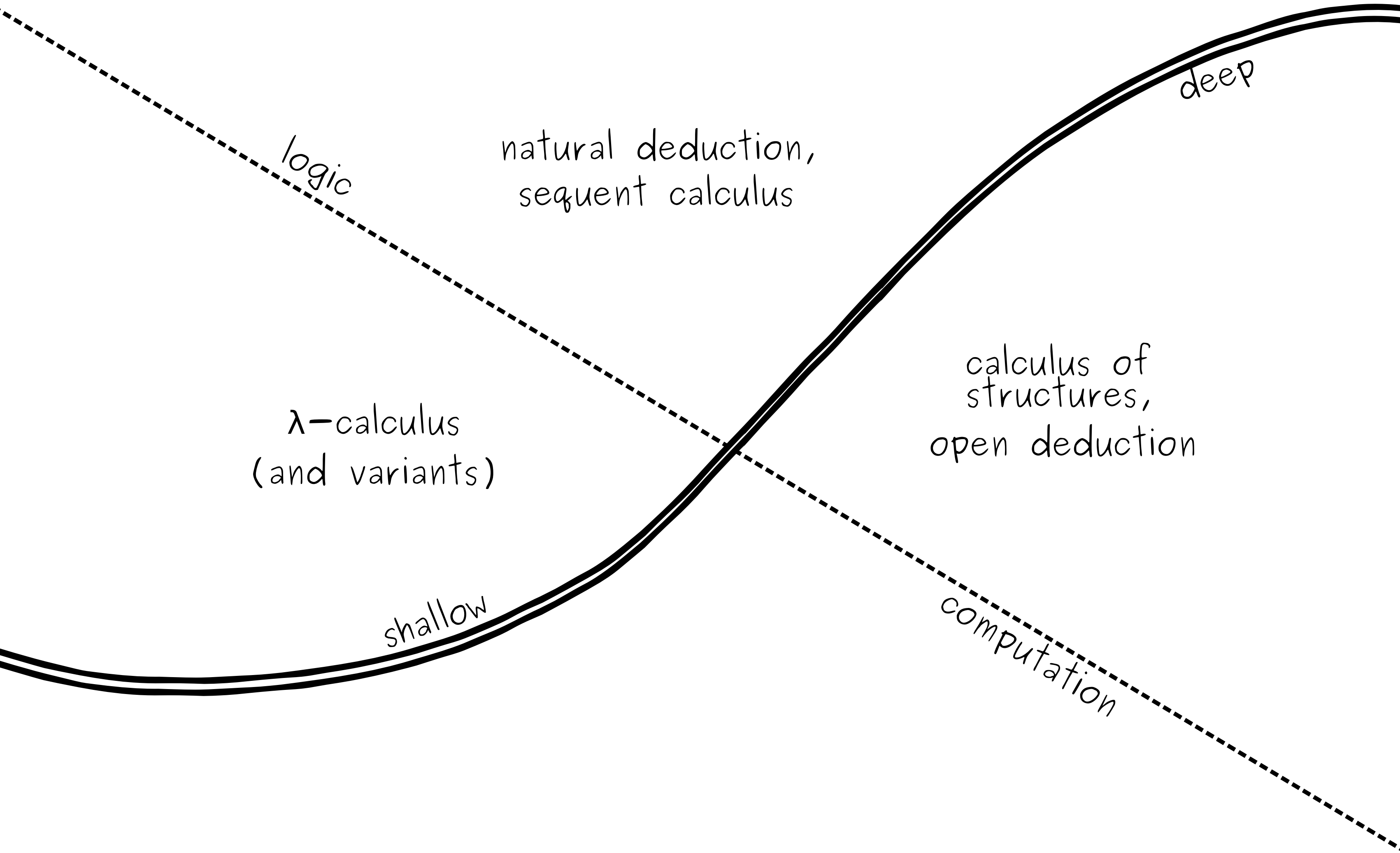
Nested Typing and Communication in the λ-calculus

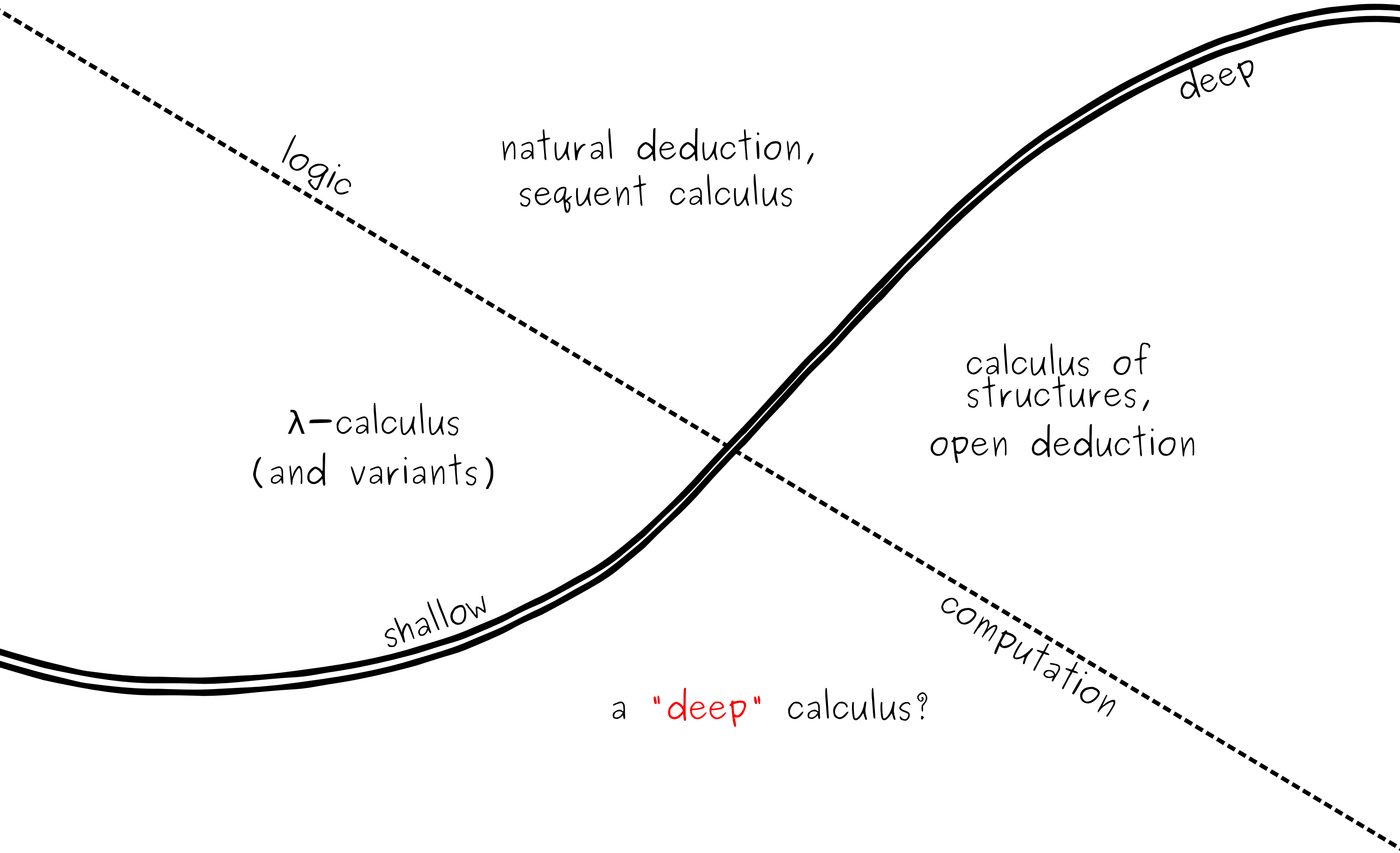
Nicolas Guenot
(ngue@itu.dk)

IT University of Copenhagen

TYPES 2013, Toulouse


logic
deep
shallow
computation


logic
natural deduction,
sequent calculus
deep
λ−calculus
(and variants)
calculus of
structures,
open deduction
shallow
computation



logic
natural deduction,
sequent calculus
deep
λ-calculus
(and variants)
calculus of
structures,
open deduction
shallow
computation
a "deep" calculus?

(a very short introduction to)

Deep Inference and the Calculus of Structures

Consider a proof in intuitionistic natural deduction:

$$\dfrac{\dfrac{}{A \vdash A}\,x \qquad \dfrac{\dfrac{}{A \vdash A}\,x \qquad \dfrac{}{A \to A \to B \vdash A \to A \to B}\,x}{A, (A \to A \to B) \vdash A \to B}\,e}{\dfrac{\dfrac{\dfrac{A, A, (A \to A \to B) \vdash B}{A, (A \to A \to B) \vdash B}\,c}{A \vdash (A \to A \to B) \to B}\,i}{\vdash A \to (A \to A \to B) \to B}\,i}\,e$$

Multisets as formulas, commas are conjunctions:

$$
\mathrm{e}\,\dfrac{\mathrm{x}\,\dfrac{}{A \vdash A} \qquad \mathrm{e}\,\dfrac{\mathrm{x}\,\dfrac{}{A \vdash A} \qquad \mathrm{x}\,\dfrac{}{A \to A \to B \vdash A \to A \to B}}{A \wedge (A \to A \to B) \vdash A \to B}}{\mathrm{i}\,\dfrac{\mathrm{i}\,\dfrac{\mathrm{c}\,\dfrac{A \wedge A \wedge (A \to A \to B) \vdash B}{A \wedge (A \to A \to B) \vdash B}}{A \vdash (A \to A \to B) \to B}}{\vdash A \to (A \to A \to B) \to B}}
$$

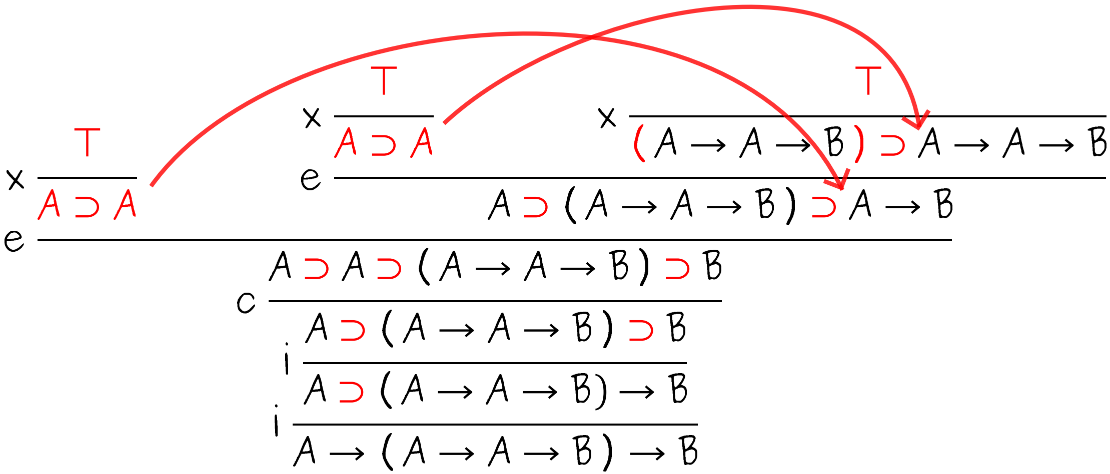
Sequents as formulas, add a meta-level implication:

$$
{\scriptstyle i}\dfrac{{\scriptstyle i}\dfrac{{\scriptstyle c}\dfrac{{\scriptstyle e}\dfrac{{\scriptstyle x}\dfrac{\top}{A \supset A} \qquad {\scriptstyle e}\dfrac{{\scriptstyle x}\dfrac{\top}{A \supset A} \qquad {\scriptstyle x}\dfrac{\top}{(A \to A \to B) \supset A \to A \to B}}{A \wedge (A \to A \to B) \supset A \to B}}{A \wedge A \wedge (A \to A \to B) \supset B}}{A \wedge (A \to A \to B) \supset B}}{A \supset (A \to A \to B) \to B}}{A \to (A \to A \to B) \to B}
$$

Assumptions can be curryfied:

$$
i\,\dfrac{i\,\dfrac{c\,\dfrac{e\,\dfrac{x\,\dfrac{\top}{A \supset A} \qquad e\,\dfrac{x\,\dfrac{\top}{A \supset A} \qquad x\,\dfrac{\top}{(A \to A \to B) \supset A \to A \to B}}{A \supset (A \to A \to B) \supset A \to B}}{A \supset A \supset (A \to A \to B) \supset B}}{A \supset (A \to A \to B) \supset B}}{A \supset (A \to A \to B) \to B}}{A \to (A \to A \to B) \to B}
$$

We can make branching more explicit:



Embedding a proof inside another is possible:

$$
{\scriptstyle i}\,\cfrac{{\scriptstyle i}\,\cfrac{{\scriptstyle c}\,\cfrac{{\scriptstyle e}\,\cfrac{{\scriptstyle x}\,\cfrac{T}{A \supset A} \qquad {\scriptstyle e}\,\cfrac{{\scriptstyle x}\,\cfrac{{\scriptstyle \equiv}\,\cfrac{{\scriptstyle x}\,\cfrac{T}{(A \to A \to B) \supset A \to A \to B}}{(A \to A \to B) \supset (T \supset A) \to A \to B}}{(A \to A \to B) \supset ((A \supset A) \supset A) \to A \to B}}{A \supset (A \to A \to B) \supset A \to B}}{A \supset A \supset (A \to A \to B) \supset B}}{A \supset (A \to A \to B) \supset B}}{A \supset (A \to A \to B) \to B}}{A \to (A \to A \to B) \to B}
$$

This is the calculus of structures:

$$\mathsf{x}\;\frac{\top}{(A\to A\to B)\supset A\to A\to B}$$

$$\equiv\;\frac{(A\to A\to B)\supset A\to A\to B}{(A\to A\to B)\supset(\top\supset A)\to A\to B}$$

$$\mathsf{x}\;\frac{(A\to A\to B)\supset(\top\supset A)\to A\to B}{(A\to A\to B)\supset((A\supset A)\supset A)\to A\to B}$$

$$\mathsf{e}\;\frac{(A\to A\to B)\supset((A\supset A)\supset A)\to A\to B}{A\supset(A\to A\to B)\supset A\to B}$$

$$\equiv\;\frac{A\supset(A\to A\to B)\supset A\to B}{A\supset(A\to A\to B)\supset(\top\supset A)\to B}$$

$$\mathsf{x}\;\frac{A\supset(A\to A\to B)\supset(\top\supset A)\to B}{A\supset(A\to A\to B)\supset((A\supset A)\supset A)\to B}$$

$$\mathsf{e}\;\frac{A\supset(A\to A\to B)\supset((A\supset A)\supset A)\to B}{A\supset A\supset(A\to A\to B)\supset B}$$

$$\mathsf{c}\;\frac{A\supset A\supset(A\to A\to B)\supset B}{A\supset(A\to A\to B)\supset B}$$

$$\mathsf{i}\;\frac{A\supset(A\to A\to B)\supset B}{A\supset(A\to A\to B)\to B}$$

$$\mathsf{i}\;\frac{A\supset(A\to A\to B)\to B}{A\to(A\to A\to B)\to B}$$

situation in the calculus of structures:

situation in the calculus of structures:

- sequents and formulas are the same (no meta-level)

situation in the calculus of structures:

- sequents and formulas are the same (no meta-level)
- it usually involves a congruence on formulas

situation in the calculus of structures:

- sequents and formulas are the same (no meta-level)
- it usually involves a congruence on formulas
- inference is rewriting

situation in the calculus of structures:

- sequents and formulas are the same (no meta-level)
- it usually involves a congruence on formulas
- inference is rewriting
- rewriting can happen deep inside a formula

situation in the calculus of structures:

- sequents and formulas are the same (no meta-level)
- it usually involves a congruence on formulas
- inference is rewriting
- rewriting can happen deep inside a formula
- branching is replaced with nesting: derivations are sequences

situation in the calculus of structures:

- sequents and formulas are the same (no meta-level)
- it usually involves a congruence on formulas
- inference is rewriting
- rewriting can happen deep inside a formula
- branching is replaced with nesting: derivations are sequences
- a proof is a derivation with premise $\top$

Situation in the calculus of structures:

- sequents and formulas are the same (no meta-level)
- it usually involves a congruence on formulas
- inference is rewriting
- rewriting can happen deep inside a formula
- branching is replaced with nesting: derivations are sequences
- a proof is a derivation with premise $\top$

This is a generalisation of natural deduction and sequent calculi

Intuitionistic Logic in the Calculus of Structures (natural deduction style)

We can extract the inference rules for system JD:

$$
\begin{array}{r|c}
 & \top \\ \hline
\mathsf{x} & (A \to A \to B) \supset A \to A \to B \\ \hline
\equiv & (A \to A \to B) \supset (\top \supset A) \to A \to B \\ \hline
\mathsf{x} & (A \to A \to B) \supset ((A \supset A) \supset A) \to A \to B \\ \hline
\mathsf{e} & A \supset (A \to A \to B) \supset A \to B \\ \hline
\equiv & A \supset (A \to A \to B) \supset (\top \supset A) \to B \\ \hline
\mathsf{x} & A \supset (A \to A \to B) \supset ((A \supset A) \supset A) \to B \\ \hline
\mathsf{e} & A \supset A \supset (A \to A \to B) \supset B \\ \hline
\mathsf{c} & A \supset (A \to A \to B) \supset B \\ \hline
\mathsf{i} & A \supset (A \to A \to B) \to B \\ \hline
\mathsf{i} & A \to (A \to A \to B) \to B
\end{array}
$$

$$\mathsf{x}\ \frac{\top}{A \supset A}$$

$$\mathsf{i}\ \frac{A \supset B}{A \to B}$$

$$\mathsf{e}\ \frac{((D \supset A) \supset A) \to B}{D \supset B}$$

$$\mathsf{w}\ \frac{B}{A \supset B}$$

$$\mathsf{c}\ \frac{A \supset A \supset B}{A \supset B}$$

and we use equations $\top \supset A \equiv A$ and $A \supset (B \supset C) \equiv B \supset (A \supset C)$

Proof normalisation in JD is performed as cut elimination in the sequent calculus, with first:

$$\dfrac{\dfrac{((D \supset A) \supset A) \supset B}{((D \supset A) \supset A) \to B}\,\mathrm{i}}{D \supset B}\,\mathrm{e} \quad \longrightarrow \quad \dfrac{((D \supset A) \supset A) \supset B}{D \supset B}\,\mathrm{cut}$$

Proof normalisation in JD is performed as cut elimination in the sequent calculus, with first:

$$\dfrac{\dfrac{((D \supset A) \supset A) \supset B}{((D \supset A) \supset A) \to B}\,i}{D \supset B}\,e \quad\longrightarrow\quad \dfrac{((D \supset A) \supset A) \supset B}{D \supset B}\,cut$$

and cuts are then permuted up until they meet an axiom:

$$\dfrac{\begin{array}{c}\dfrac{T}{(T \supset A) \supset A}\,x \\ P\| \\ ((D \supset A) \supset A) \supset A\end{array}}{D \supset A}\,cut \quad\longrightarrow\quad \begin{array}{c} T \\ P\| \\ D \supset A \end{array}$$

(all detours are turned into cuts and then eliminated)

Nested Typing
for the λ-calculus

For an exact correspondence, we would need combinators, e.g.:

$$P \; ; \; cut(a,b) \; ; \; r \; ; \; Q \longrightarrow P \; ; \; r \; ; \; cut(a,b) \; ; \; Q$$

$$P \; ; \; cut(a,b) \; ; \; R(b) \; ; \; x(a) \; ; \; Q \longrightarrow P \; ; \; R(a) \; ; \; Q$$

$$P \; ; \; cut(a,b) \; ; \; R(b) \; ; \; w(a) \; ; \; Q \longrightarrow P \; ; \; w(\ldots) \; ; \; Q$$

$$\ldots \longrightarrow \ldots$$

For an exact correspondence, we would need combinators, e.g.:

$$P \;;\; \mathrm{cut}(a,b) \;;\; r \;;\; Q \longrightarrow P \;;\; r \;;\; \mathrm{cut}(a,b) \;;\; Q$$

$$P \;;\; \mathrm{cut}(a,b) \;;\; R(b) \;;\; x(a) \;;\; Q \longrightarrow P \;;\; R(a) \;;\; Q$$

$$P \;;\; \mathrm{cut}(a,b) \;;\; R(b) \;;\; w(a) \;;\; Q \longrightarrow P \;;\; w(\ldots) \;;\; Q$$

$$\ldots \longrightarrow \ldots$$

but this would be heavy and difficult to analyse, here we choose to stick to a setting we know better:

the λ-calculus is the interpretation of NJ!

We consider the λ-calculus with explicit substitutions:

$$t, u ::= x \mid \lambda x.t \mid t\,u \mid t[u/x]$$

We consider the λ-calculus with explicit substitutions:

$$t, u ::= x \mid \lambda x.t \mid t\,u \mid t[u/x]$$

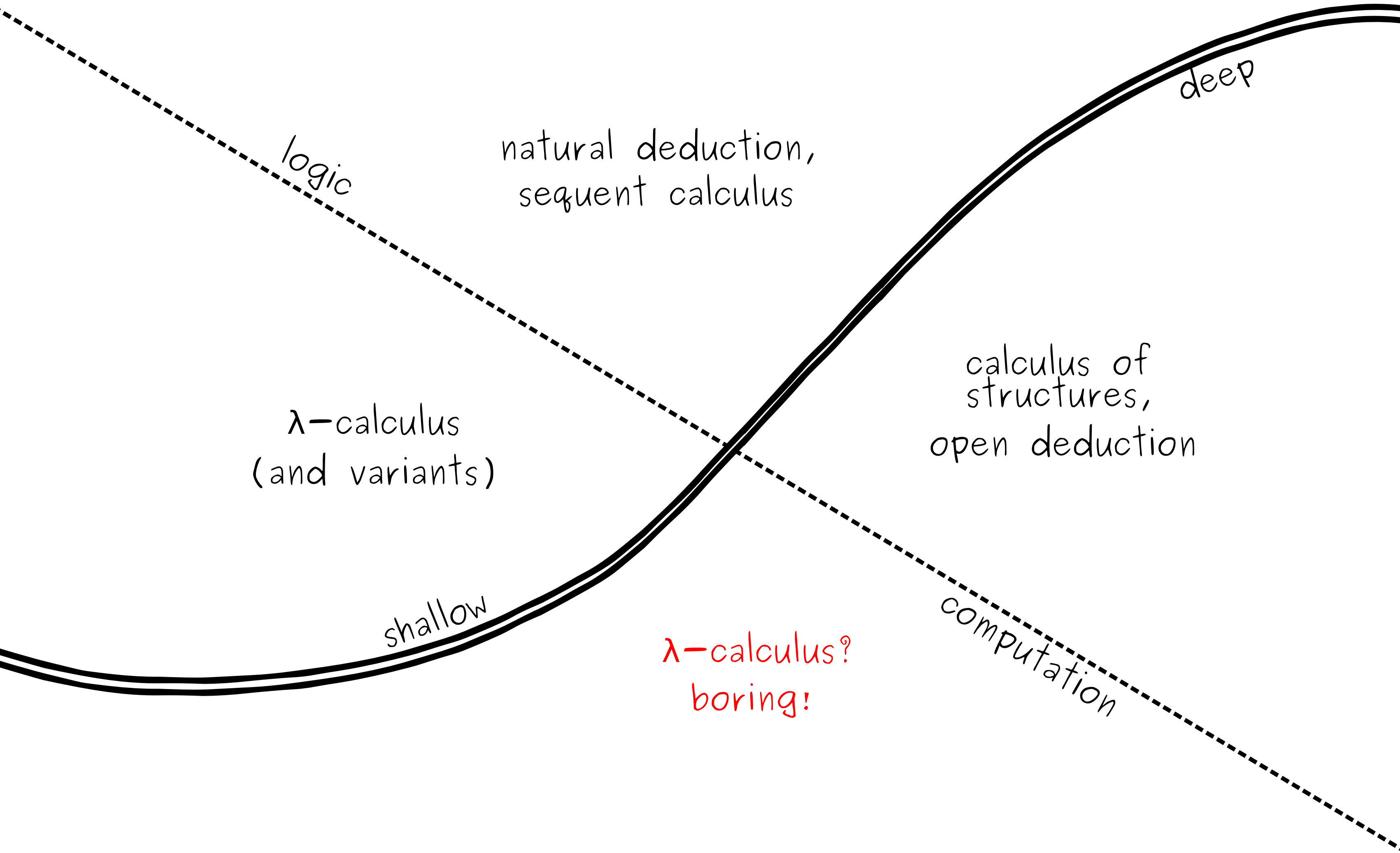
and we can use deep inference rules for nested typing:

$$\text{var}\ \frac{\circ}{x : \{\circ\} \triangleright A \vdash x : A} \qquad \text{lam}\ \frac{G, x : A \vdash t : B}{G \vdash \lambda x.t : A \to B}$$

$$\text{rem}\ \frac{G \vdash t : B}{G, x : A \vdash t : B} \qquad \text{dup}\ \frac{G, x : A, x : A \vdash t : B}{G, x : A \vdash t : B}$$

$$\text{app}\ \frac{G \vdash t : (\{D \vdash u : A\} \triangleright A) \to B}{G, D \vdash t\,u : B} \qquad \text{sub}\ \frac{G, x : \{D \vdash u : A\} \triangleright A \vdash t : B}{G, D \vdash t[u/x] : B}$$

and it yields the reduction of a standard calculus (that's λs)


logic
natural deduction,
sequent calculus
deep
λ-calculus
(and variants)
calculus of
structures,
open deduction
shallow
λ-calculus?
boring!
computation

(and now for some real deep inference...)

The translation of branching is not done this way in CoS:

$$e\ \frac{C_1, \cdots, C_n \vdash A \qquad D_1, \cdots, D_k \vdash A \to B}{C_1, \cdots, C_n, D_1, \cdots, D_k \vdash B}$$

$$\longrightarrow \qquad e\ \frac{D_1 \supset \cdots \supset D_k \supset ((C_1 \supset \cdots \supset C_n \supset A) \to B}{C_1 \supset \cdots \supset C_n \supset D_1 \supset \cdots \supset D_k \supset B}$$

The translation of branching is not done this way in CoS:

$$
e\,\frac{C_1, \cdots, C_n \vdash A \qquad D_1, \cdots, D_k \vdash A \to B}{C_1, \cdots, C_n, D_1, \cdots, D_k \vdash B}
$$

$$
\longrightarrow \qquad e\,\frac{D_1 \supset \cdots \supset D_k \supset ((C_1 \supset \cdots \supset C_n \supset A) \to B}{C_1 \supset \cdots \supset C_n \supset D_1 \supset \cdots \supset D_k \supset B}
$$

but rather:

$$
e\,\cfrac{s\,\cfrac{s\,\cfrac{\cdots}{C_1 \supset \cdots \supset C_{n-1} \supset D_1 \supset \cdots \supset D_k \supset ((C_n \supset A) \supset A) \to B}\;\text{ where }\;\cdots = s\,\cfrac{s\,\cfrac{D_1 \supset \cdots \supset D_k \supset ((C_1 \supset \cdots \supset C_n \supset A) \to B}{C_1 \supset D_1 \supset \cdots \supset D_k \supset ((C_2 \supset \cdots \supset C_n \supset A) \supset A) \to B}}{\cdots}}{C_1 \supset \cdots \supset C_n \supset D_1 \supset \cdots \supset D_k \supset (A \supset A) \to B}}{C_1 \supset \cdots \supset C_n \supset D_1 \supset \cdots \supset D_k \supset B}
$$

The decomposed system JDs is:

$$\mathsf{x}\,\frac{\top}{A \supset A} \qquad \mathsf{i}\,\frac{A \supset B}{A \rightarrow B} \qquad \mathsf{e}\,\frac{(A \supset A) \rightarrow B}{B}$$

$$\mathsf{w}\,\frac{B}{A \supset B} \qquad \mathsf{c}\,\frac{A \supset A \supset B}{A \supset B} \qquad \mathsf{s}\,\frac{((A \supset B) \supset C) \rightarrow D}{A \supset ((B \supset C) \rightarrow D)}$$

(we still use equations $\top \supset A \equiv A$ and $A \supset (B \supset C) \equiv B \supset (A \supset C)$ of JD)

The decomposed system JDs is:

$$x\,\frac{\top}{A \supset A} \qquad i\,\frac{A \supset B}{A \to B} \qquad e\,\frac{(A \supset A) \to B}{B}$$

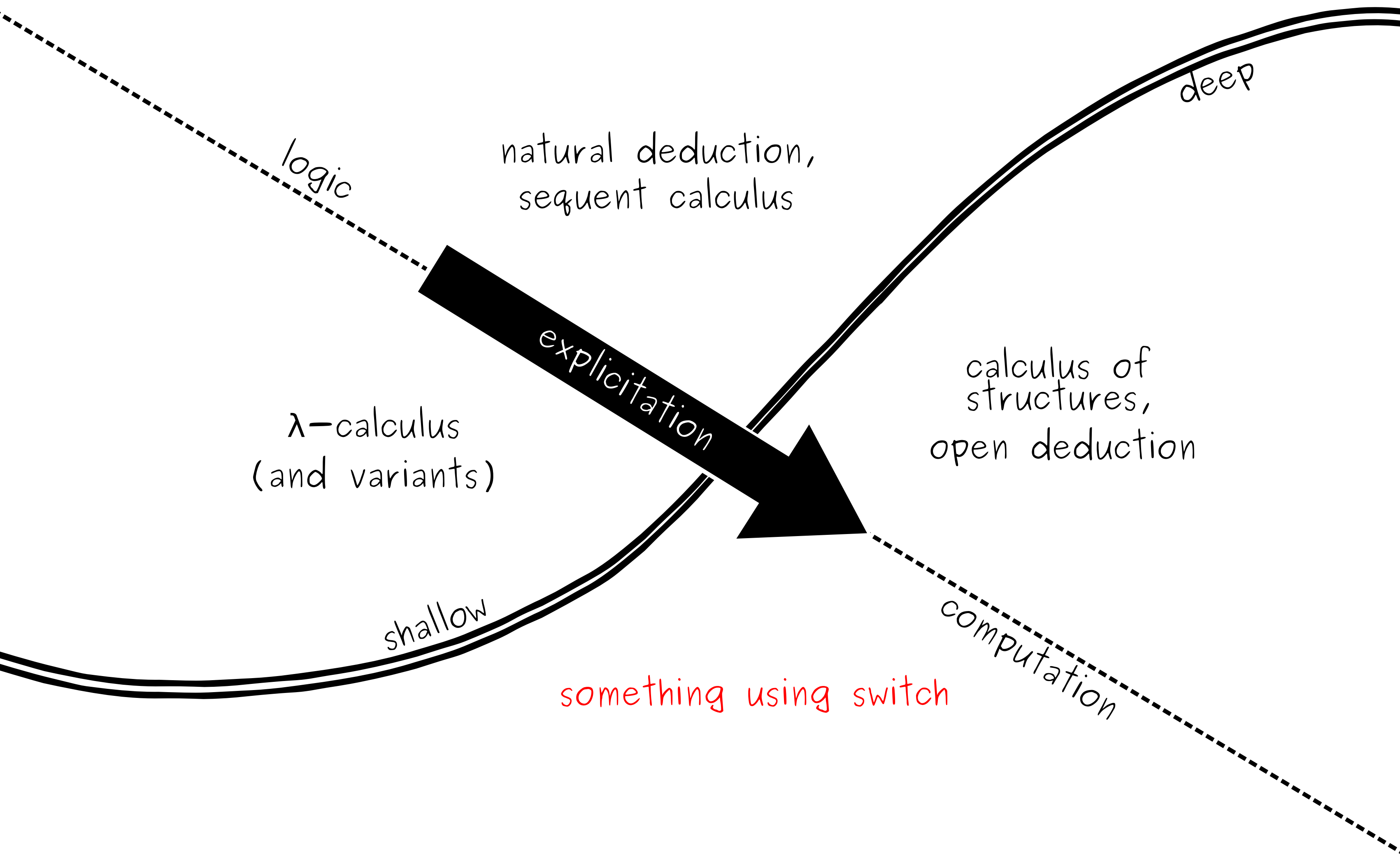
$$w\,\frac{B}{A \supset B} \qquad c\,\frac{A \supset A \supset B}{A \supset B} \qquad s\,\frac{((A \supset B) \supset C) \to D}{A \supset ((B \supset C) \to D)}$$

(we still use equations $\top \supset A \equiv A$ and $A \supset (B \supset C) \equiv B \supset (A \supset C)$ of JD)

In the JDs system, the switch rule decomposes the operation of context splitting in the implication elimination rule

Back to the example proof:

$$
\begin{array}{rc}
 & \top \\ \cline{2-2}
x & (A \to A \to B) \supset A \to A \to B \\ \cline{2-2}
\equiv & (A \to A \to B) \supset (\top \supset A) \to A \to B \\ \cline{2-2}
x & (A \to A \to B) \supset ((A \supset A) \supset A) \to A \to B \\ \cline{2-2}
s & A \supset (A \to A \to B) \supset (A \supset A) \to A \to B \\ \cline{2-2}
e & A \supset (A \to A \to B) \supset A \to B \\ \cline{2-2}
\equiv & A \supset (A \to A \to B) \supset (\top \supset A) \to B \\ \cline{2-2}
x & A \supset (A \to A \to B) \supset ((A \supset A) \supset A) \to B \\ \cline{2-2}
s & A \supset A \supset (A \to A \to B) \supset (A \supset A) \to B \\ \cline{2-2}
e & A \supset A \supset (A \to A \to B) \supset B \\ \cline{2-2}
c & A \supset (A \to A \to B) \supset B \\ \cline{2-2}
i & A \supset (A \to A \to B) \to B \\ \cline{2-2}
i & A \to (A \to A \to B) \to B
\end{array}
$$


logic
natural deduction,
sequent calculus
deep
explicitation
calculus of
structures,
open deduction
λ−calculus
(and variants)
shallow
computation
something using switch

Explicit communication in the λ-calculus

The switch corresponds to a communication operator:

$$\text{com}\ \frac{G \vdash t : (\{D, y : A \vdash u : B\} \triangleright C) \to D}{G, x : A \vdash \bar{a}x.t : (\{D \vdash ay.u : B\} \triangleright C) \to D}$$

where two syntactic constructs are decomposed at the same time, to perform the communication of a typing resource

The switch corresponds to a communication operator:

$$\text{com}\ \frac{G \vdash t : (\{D, y : A \vdash u : B\} \triangleright C) \to D}{G, x : A \vdash \bar{a}x.t : (\{D \vdash ay.u : B\} \triangleright C) \to D}$$

where two syntactic constructs are decomposed at the same time, to perform the communication of a typing resource

The λc–calculus is a refinement of λs:

$$t, u \ ::= \ x \ \mid \ \lambda x.t \ \mid \ t\,u \ \mid \ t[u/x] \ \mid \ \bar{a}x.t \ \mid \ ax.t$$

where the explicit substitutions are resources and communication between subterms is made explicit by operators for input/output of resources (on channels)

Following normalisation, the reduction rules are:

$$
\begin{array}{rll}
(\lambda x.t)\,u & \longrightarrow_{B} & t[u/x] \\
\\
x[u/x] & \longrightarrow_{var} & u \\
t[u/x] & \longrightarrow_{rem} & t \\
t[u/x] & \longrightarrow_{dup} & t_{[y/x]}[u/x][u/y] \\
\\
(\lambda y.t)[u/x] & \longrightarrow_{lam} & \lambda y.t[u/x] \\
(t\ v)[u/x] & \longrightarrow_{app} & t[u/x]\ v \\
\\
(a y.t)[u/x] & \longrightarrow_{get} & a y.t[u/x] \\
(\bar{a} y.t)[u/x] & \longrightarrow_{snd} & \bar{a} y.t[u/x] \\
\\
k\{(\bar{a}x.t)[u/x]\}\,(a y.v) & \longrightarrow_{com} & k\{\bar{a}\tilde{u}.t\}\,(a\tilde{u}.v[u/y])
\end{array}
$$

where k is a spine-context (plus side conditions)

Following normalisation, the reduction rules are:

$$
\begin{array}{rcl}
(\lambda x.t)\,u & \longrightarrow_{B} & t[u/x] \\
x[u/x] & \longrightarrow_{var} & u \\
t[u/x] & \longrightarrow_{rem} & t \\
t[u/x] & \longrightarrow_{dup} & t_{[y/x]}[u/x][u/y] \\
(\lambda y.t)[u/x] & \longrightarrow_{lam} & \lambda y.t[u/x] \\
(t\ v)[u/x] & \longrightarrow_{app} & t[u/x]\ v \\
(ay.t)[u/x] & \longrightarrow_{get} & ay.t[u/x] \\
(\bar{a}y.t)[u/x] & \longrightarrow_{snd} & \bar{a}y.t[u/x] \\
k\{(\bar{a}x.t)[u/x]\}\,(ay.v) & \longrightarrow_{com} & k\{\bar{a}\tilde{u}.t\}\,(a\tilde{u}.v[u/y])
\end{array}
$$

where k is a spine-context (plus side conditions)

Following normalisation, the reduction rules are:

$$
\begin{array}{rcl}
(\lambda x.t)\,u & \longrightarrow_{B} & t[u/x] \\
\\
x[u/x] & \longrightarrow_{var} & u \\
t[u/x] & \longrightarrow_{rem} & t \\
t[u/x] & \longrightarrow_{dup} & t_{[y/x]}[u/x][u/y] \\
\\
(\lambda y.t)[u/x] & \longrightarrow_{lam} & \lambda y.t[u/x] \\
(t\;v)[u/x] & \longrightarrow_{app} & t[u/x]\;v \\
\\
(a y.t)[u/x] & \longrightarrow_{get} & a y.t[u/x] \\
(\bar{a} y.t)[u/x] & \longrightarrow_{snd} & \bar{a} y.t[u/x] \\
\\
k\{(\bar{a}x.t)[u/x]\}\,(a y.v) & \longrightarrow_{com} & k\{\bar{a}\tilde{u}.t\}\,(a\tilde{u}.v[u/y])
\end{array}
$$

where k is a spine-context (plus side conditions)

In λc, there are synchronisation points:

$$(\bar{a}x.t)[u/x]\,(ay.v) \quad \longrightarrow_{com} \quad (\bar{a}u.t)\,(au.v[u/y])$$

so that in $(\bar{a}x.t)[u/x]$, the substitution cannot be pushed, and it disappears only when this term is applied to an argument of the appropriate shape:

there are deadlocks in the untyped λc–calculus!

In λc, there are synchronisation points:

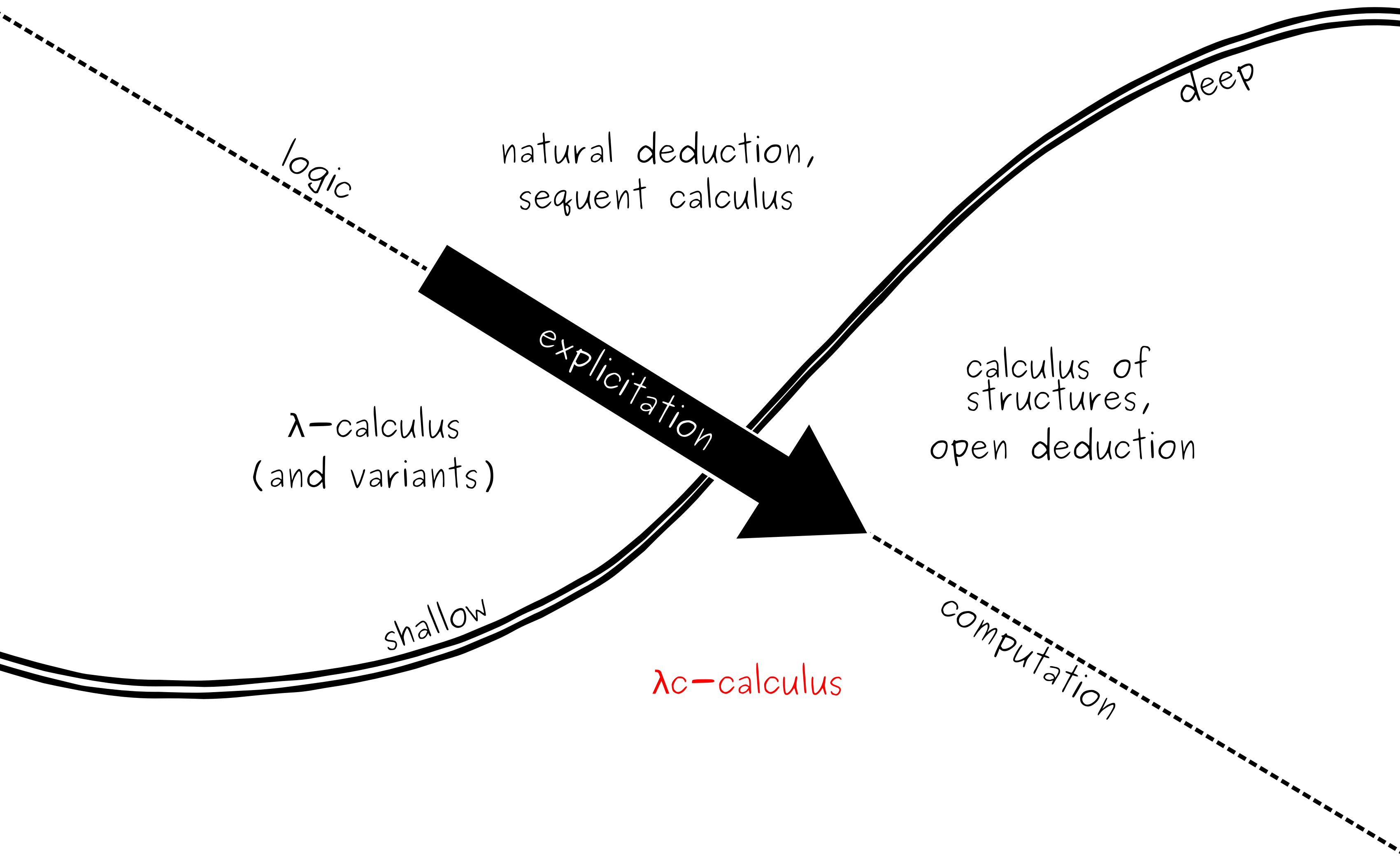
$$(\bar{a}x.t)[u/x]\,(ay.v) \quad \longrightarrow_{com} \quad (\bar{a}u.t)\,(au.v[u/y])$$

so that in $(\bar{a}x.t)[u/x]$, the substitution cannot be pushed, and it disappears only when this term is applied to an argument of the appropriate shape:

there are deadlocks in the untyped λc–calculus!

Notice that:

- well–typed terms admit deadlock–free reduction
- a plain λ–term can usually not reduce properly in λc
- synchronisation points partially force a reduction strategy


logic
natural deduction,
sequent calculus
deep
explicitation
calculus of
structures,
open deduction
λ−calculus
(and variants)
shallow
λc−calculus
computation

The correspondence is not perfect here, there are more typing derivations than terms because of trivial permutations:

$$\text{sub}\ \frac{G, x : \{ \vdash u : A \} \triangleright A \vdash t : B}{G \vdash t[u/x] : B}$$

go on typing t or u?

The correspondence is not perfect here, there are more typing derivations than terms because of trivial permutations:

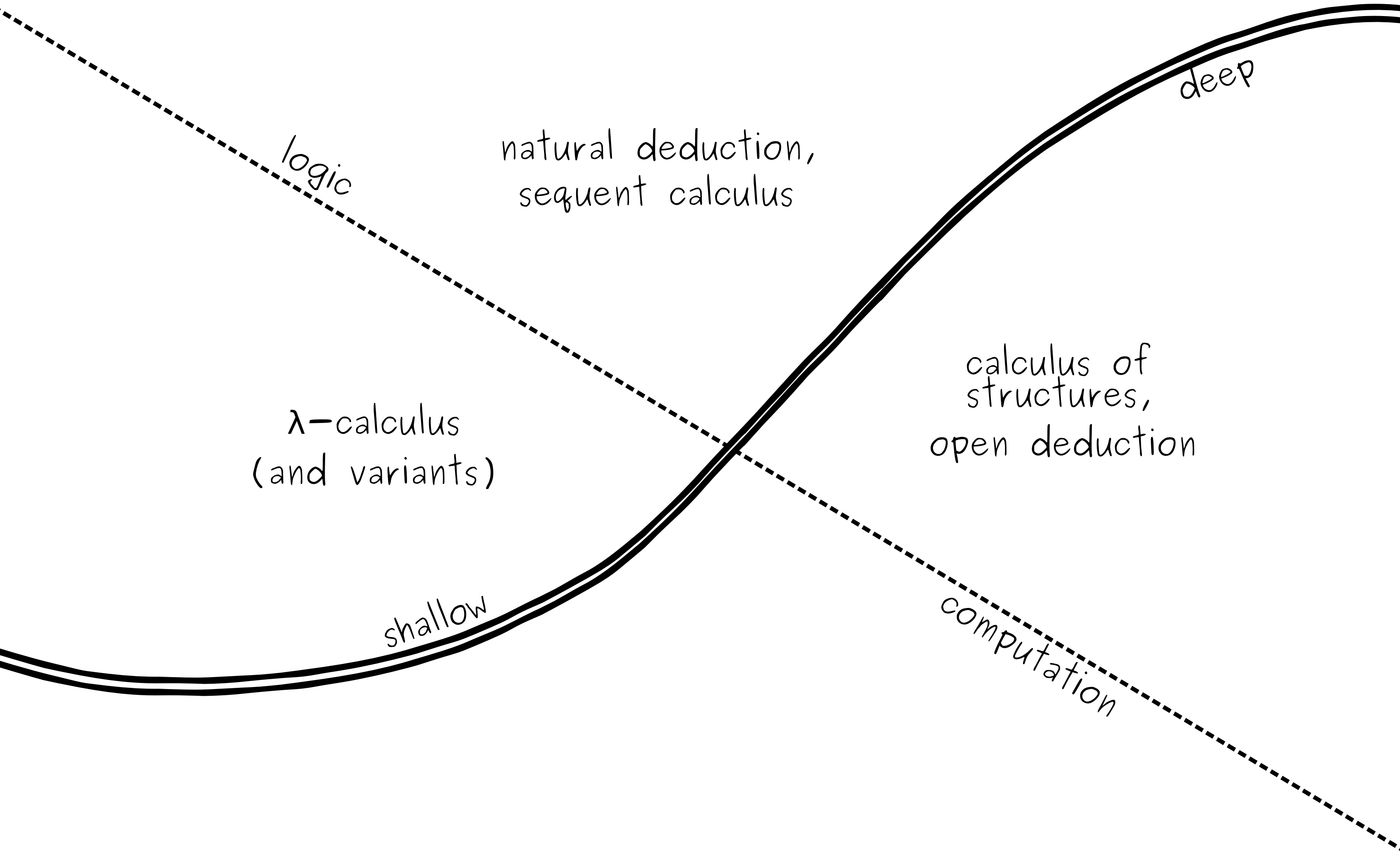
$$\text{sub}\ \frac{G, x : \{\vdash u : A\} \rhd A \vdash t : B}{G \vdash t[u/x] : B}$$

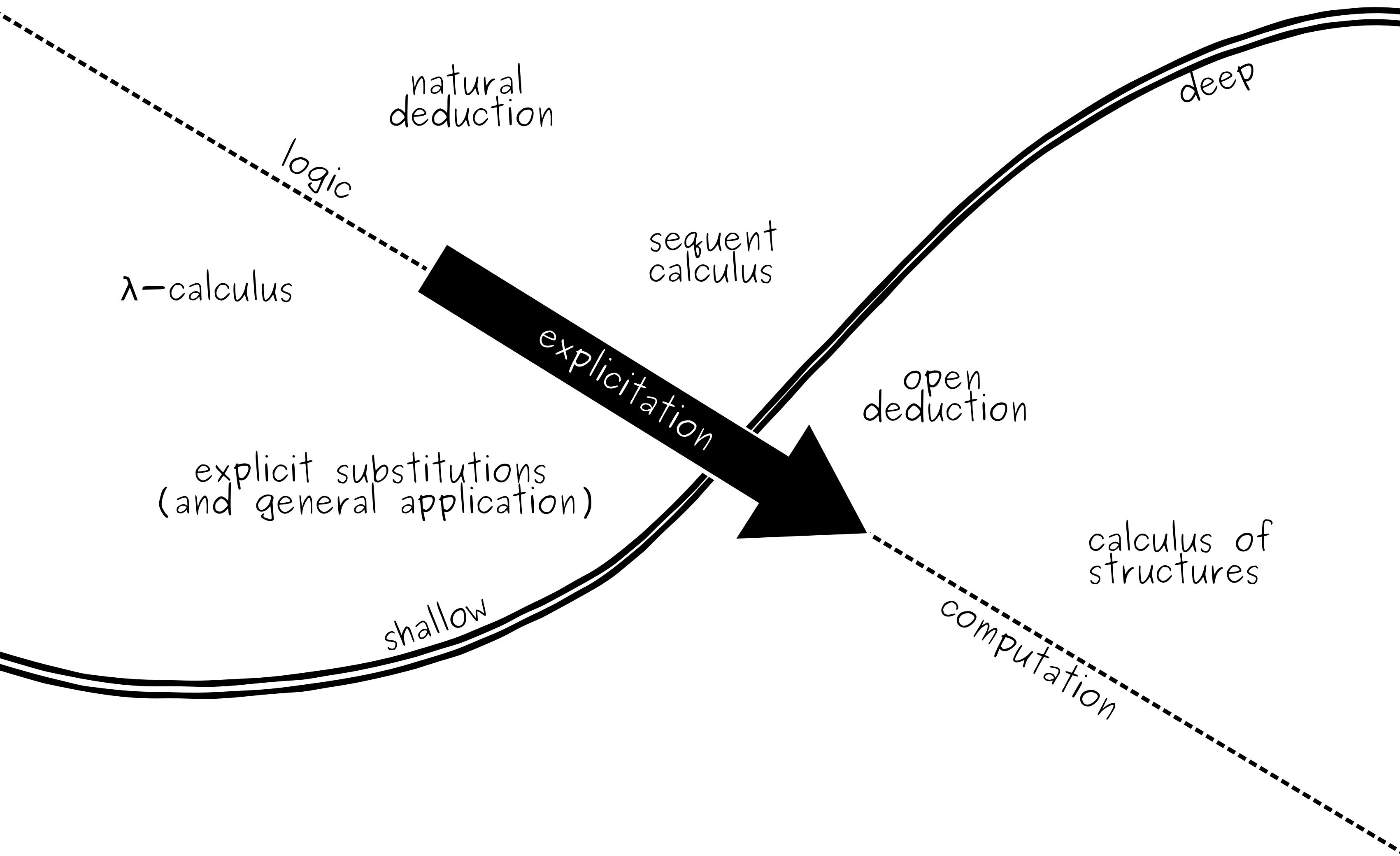
go on typing t or u?

The interleaving of branches in CoS is not expressed in the tree structure of a λ-term:

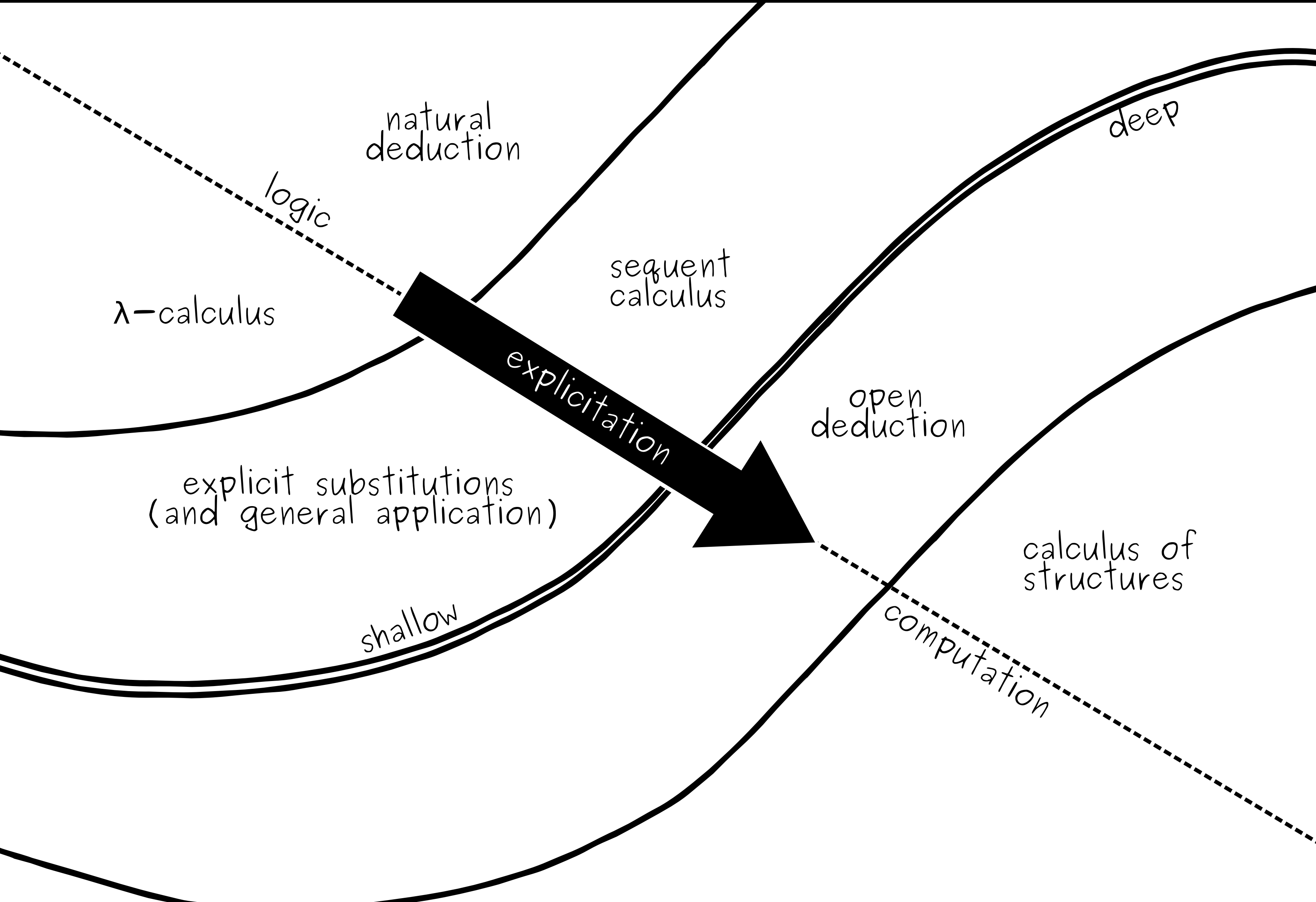
The λc-calculus should provide a Curry-Howard interpretation for an open deduction variant of JDs

(in OD, derivations can appear in parallel, in a graph structure)


logic
natural deduction,
sequent calculus
deep
λ-calculus
(and variants)
calculus of
structures,
open deduction
shallow
computation

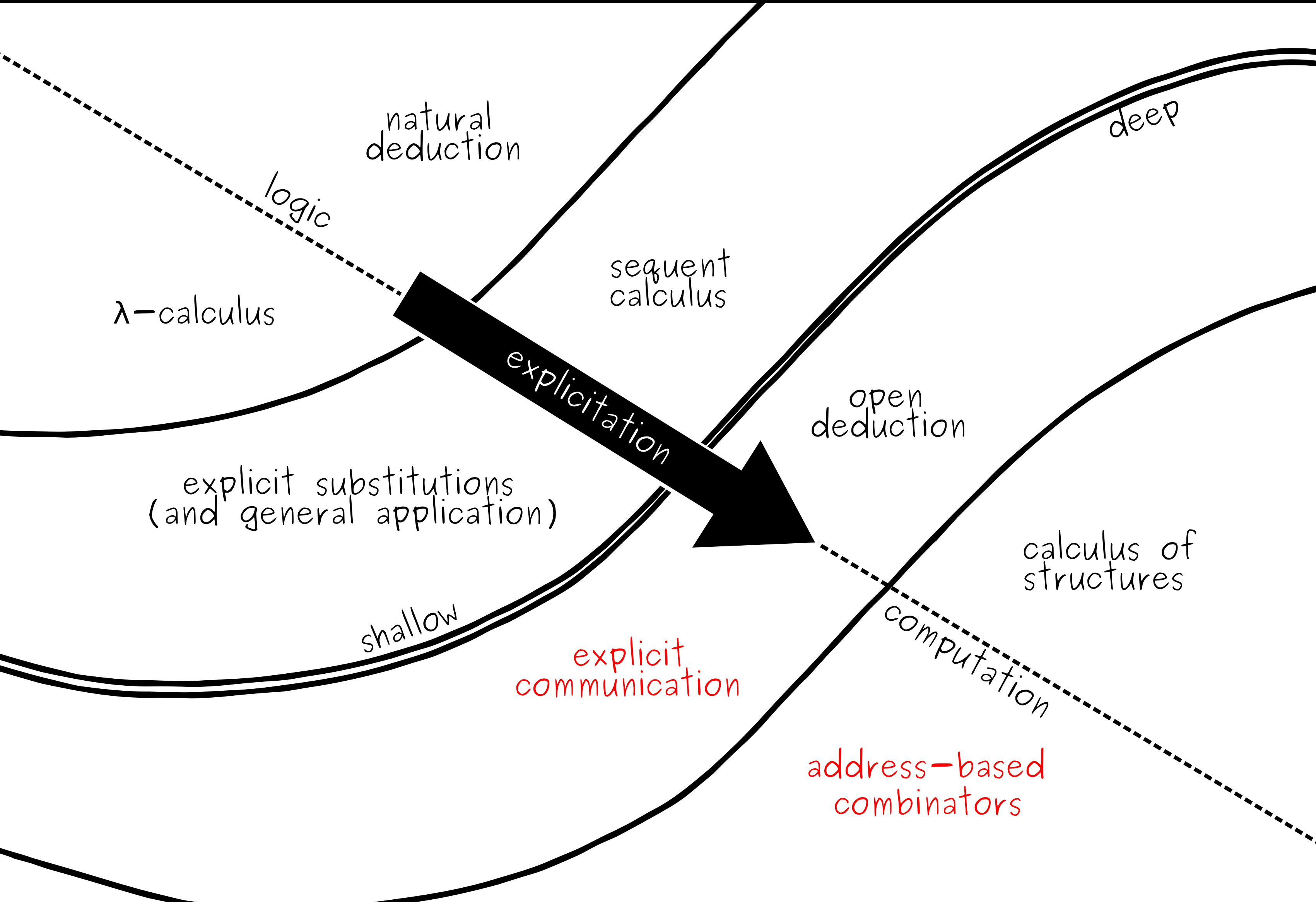


natural deduction
logic
sequent calculus
λ−calculus
explicitation
deep
open deduction
explicit substitutions (and general application)
calculus of structures
shallow
computation


natural deduction
logic
sequent calculus
λ−calculus
explicitation
deep
open deduction
explicit substitutions (and general application)
shallow
calculus of structures
computation

Future work:

- prove properties of typed/untyped λc
- establish a precise correspondence with open deduction
- what part of the strategy is embedded in synchronisations?
- explore a correspondence with (address-based) combinators


natural deduction
logic
λ-calculus
sequent calculus
deep
explicitation
open deduction
explicit substitutions (and general application)
calculus of structures
shallow
computation
explicit communication
address-based combinators