C-Ludics and Natural Language

Christophe Fouqueré

Laboratoire d'Informatique de Paris-Nord
Université de Paris 13 - CNRS 7030

Computational Ludics (K. Terui, 2008)

- **Goal**: Logical reconstruction of computability and complexity theory
- **Why** based on Ludics: monism, interaction, existentialism
  - Monism: One kind of object versus automata/word, program/data, grammar/sentence, ...
  - Interaction:
    - Primitive of computation (cf Curry-Howard), acceptance viewed as orthogonality
    - Symmetry of the process (versus functions)
  - Existentialism: not necessarily closed objects

Computational Ludics: Preliminary Results

- Correspondence with *automata theory*
- Computational presentation of (generalized) Ludics (and generalization of completeness properties)
- Introduction of *design generators* (hence finite presentation of infinite designs)

Ludics vs Automata

Correspondence:

set of Actions	vs	Alphabet
Design	vs	Word
Behaviour	vs	Language
Formula/Type	vs	(eg reg.) Expression
Orthogonality	vs	**Acceptance**

$\mathfrak{D} \perp \mathfrak{R}$ iff $[\![\mathfrak{D}, \mathfrak{R}]\!] = \mathfrak{Dai}$ iff $\mathfrak{D}^*$ is accepted by $\mathfrak{R}^*$

Generalizing Ludics (1)

First of all, Terui defines:

- A **term language** for design presentation.
- A specific language for **generating** designs, hence giving the possibility to speak of, say, regular expressions.

Generalizing Ludics (2)

Second:

- Loci addresses are absolute and reflect a chronicle order
  ➥ In an action, a focus and its ramification are replaced by a **name** together with a set of (distinct) **variables**. Hence, variables may be replaced/reduced by designs during a normalization step.

Generalizing Ludics (3)

Third:

- Designs are cut-free, however a design may be the normalized form between two designs
  ➥ Explicit **cuts** are added to the term language.

- Designs are identity-free
  ➥ As soon as variables are part of the term language, replacing a variable by another one is nothing else but an **identity**.

Term Language

- Girard/Curien's original designs:
  - Normal, η-expanded, linear lambda terms (maybe infinitary).
  - Actions built with ramifications: $I \in \mathcal{P}_f(\mathbb{N})$.
  - (in Desseins) a node is addressed by a locus.
  - (in Desseins) children of negative nodes may have a (maybe infinite) set of negative actions with same focus

- **C-designs**:
  - Arbitrary terms.
  - Actions built from a given signature: $\mathcal{A} = (A, ar)$ where
    - A is a set of names,
    - $ar : A \to \mathbb{N}$ gives an arity to each name.
  - A term is referenced by a name.
  - A specific constant corresponds to undefined positive subnodes, hence divergence: Ω

Term Language: Actions

Actions are polarized:

- A **positive** action is one of the following:

$\maltese$	`daemon`
Ω	`divergence`
$\overline{u}$	`proper positive action,` $u \in A$

- A **negative** action is one of the following:

x	`variable`
$u(x_1, \ldots, x_{ar(u)})$	`proper negative action`

($x_1, \ldots, x_n$ noted $\overrightarrow{x_i}$)

Term Language: C-designs

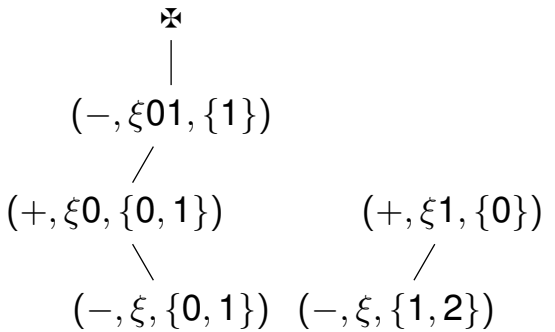
The sets of positive and negative designs are coinductively defined by:

$$
\begin{array}{lcll}
P & ::= & \maltese & \texttt{daemon} \\
 & | & \Omega & \texttt{divergence} \\
 & | & N_0 | \overline{u} \langle \overrightarrow{N_i} \rangle & \texttt{proper action} \\
 & & & \\
N & ::= & x & \texttt{variable} \\
 & | & \sum_u u(\overrightarrow{x_i}).P_u & \texttt{(negative) choices}
\end{array}
$$

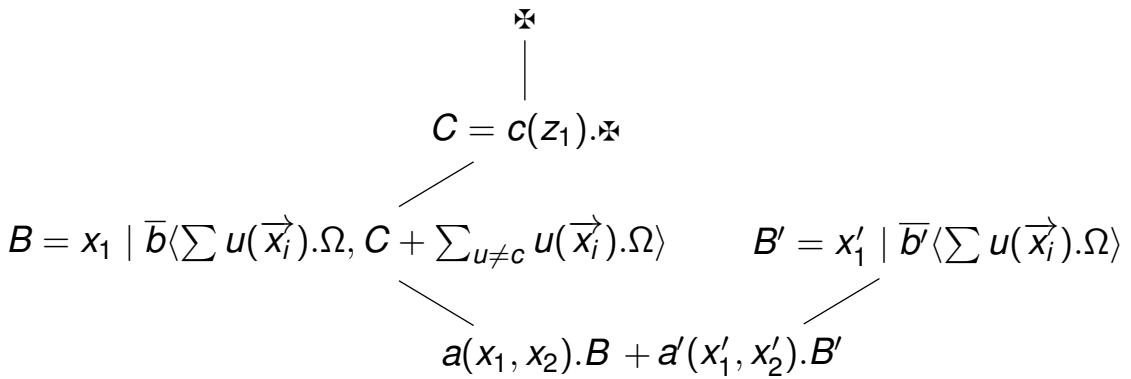
In Ludics:

$$\dfrac{\dfrac{\dfrac{}{\vdash \xi 00}\ \dfrac{\dfrac{}{\vdash \xi 011}\ \maltese}{\xi 01 \vdash}}{\vdash \xi 0, \xi 1}\qquad \dfrac{\xi 10 \vdash \xi 2}{\vdash \xi 1, \xi 2}}{\xi \vdash}$$

or



As a c-design:



or

$$\begin{aligned} & a(x_1,x_2).\left(x_1 \mid \overline{b}\langle \textstyle\sum u(\overrightarrow{x_i}).\Omega \ \ , \ \ c(z_1).\maltese + \textstyle\sum_{u\neq c} u(\overrightarrow{x_i}).\Omega\rangle\right) \\ & + a'(x'_1,x'_2).\left(x'_1 \mid \overline{b'}\langle \textstyle\sum u(\overrightarrow{x_i}).\Omega\rangle\right) \\ & + \textstyle\sum_{u\neq a,a'} u(\overrightarrow{x_i}).\Omega \end{aligned}$$

Term Normalisation: Extension of λ-calculus

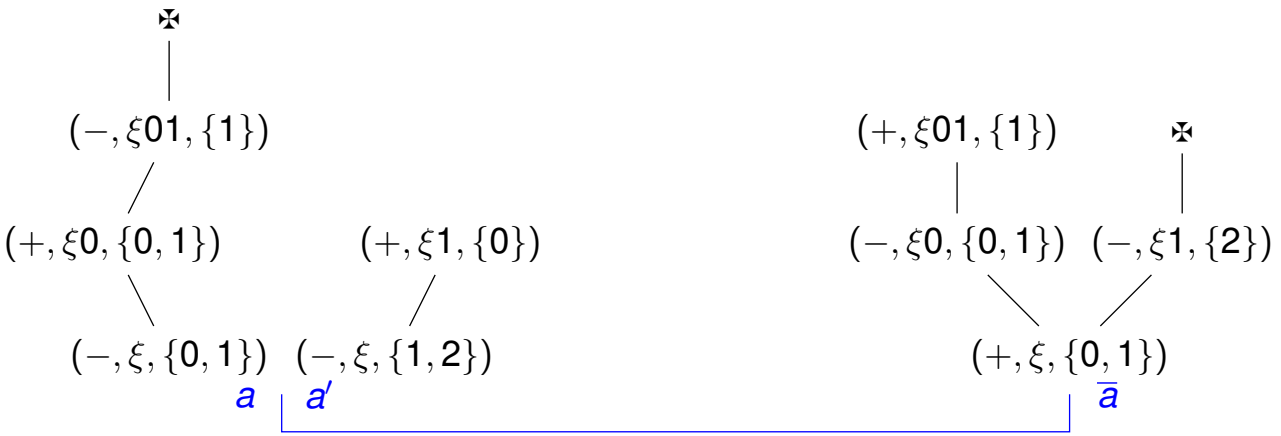
In λ-calculus:

$$(\lambda x_1 \dots x_n.P)N_1 \dots N_n \quad \longrightarrow \quad P[N_1/x_1, \dots, N_n/x_n]$$

In C-Ludics:

- **Reduction rule** of c-designs:
  $$\sum_u u(x_1 \dots x_{ar(u)}).P_u \mid \overline{u_0}\langle N_1 \dots N_{ar(u_0)}\rangle$$
  $$\longrightarrow \quad P_{u_0}[N_1/x_1, \dots, N_{ar(u_0)}/x_{ar(u_0)}]$$
- **Normalization**:
  $P \Downarrow Q$ if $P \longrightarrow^* Q$ and Q is neither a cut nor Ω
  If such a Q exists, it is noted $[\![P]\!]$
- **Orthogonality**: $N \perp P$ if $P[N/x] \Downarrow \maltese$

Term Normalization: a Linear Example



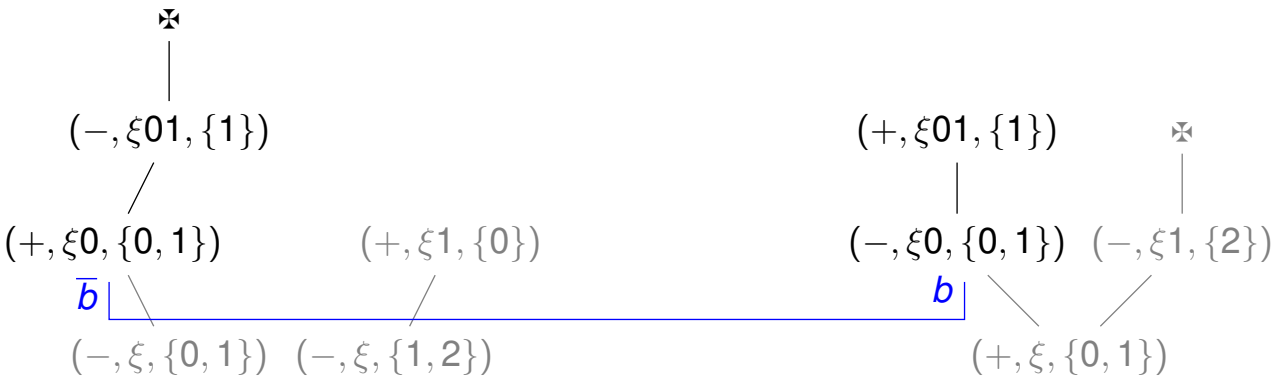
$$\left(a(x_1,x_2).\left(x_1 \mid \overline{b}\langle \sum u(\overrightarrow{x_i}).\Omega \ , \ c(z_1).\maltese + \sum_{u\neq c} u(\overrightarrow{x_i}).\Omega\rangle\right) + a'(x'_1,x'_2).\left(x'_1 \mid \overline{b'}\langle \sum u(\overrightarrow{x_i}).\Omega\rangle\right) + \ldots\right)$$
$$\mid \overline{a}\langle b(y_1,y_2).\left(y_2 \mid \overline{c}\langle \sum u(\overrightarrow{x_i}).\Omega\rangle\right) + \sum_{u\neq b} u(\overrightarrow{x_i}).\Omega \ , \ b''(y''_1).\maltese + \sum_{u\neq b''} u(\overrightarrow{x_i}).\Omega\rangle$$

$$\rightarrow \quad \left(b(y_1,y_2).\left(y_2 \mid \overline{c}\langle \sum u(\overrightarrow{x_i}).\Omega\rangle\right) + \sum_{u\neq b} u(\overrightarrow{x_i}).\Omega\right) \mid \overline{b}\langle \sum u(\overrightarrow{x_i}).\Omega \ , \ c(z_1).\maltese + \sum_{u\neq c} u(\overrightarrow{x_i}).\Omega\rangle$$

$$\rightarrow \quad \left(c(z_1).\maltese + \sum_{u\neq c} u(\overrightarrow{x_i}).\Omega\right) \mid \overline{c}\langle \sum u(\overrightarrow{x_i}).\Omega\rangle$$

$$\rightarrow \quad \maltese$$

Term Normalization: a Linear Example



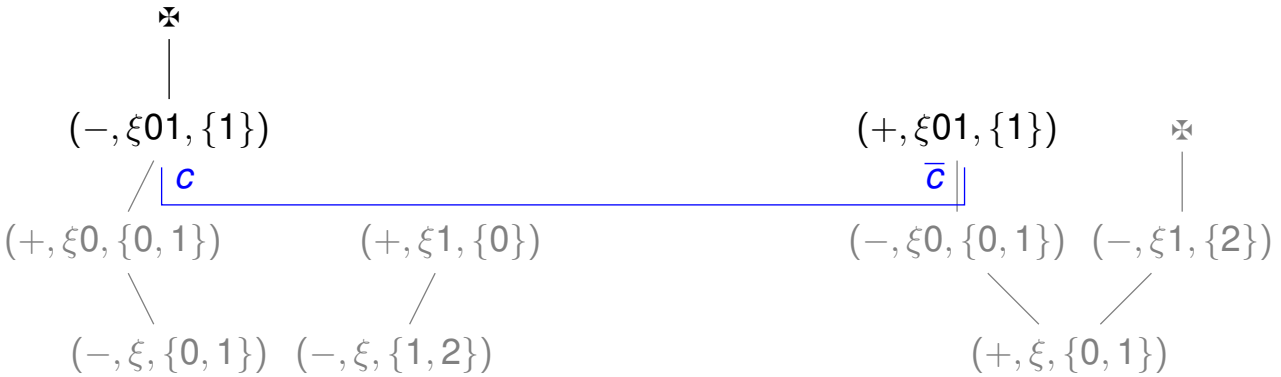
$$\Big(a(x_1,x_2).\Big(x_1 \mid \overline{b}\langle \textstyle\sum u(\overrightarrow{x_i}).\Omega \;,\; c(z_1).\maltese + \sum_{u\neq c} u(\overrightarrow{x_i}).\Omega\rangle\Big) + a'(x'_1,x'_2).\,(x'_1 \mid \overline{b'}\langle \textstyle\sum u(\overrightarrow{x_i}).\Omega\rangle) + \ldots\Big)$$
$$\mid \overline{a}\langle b(y_1,y_2).\,(y_2|\overline{c}\langle \textstyle\sum u(\overrightarrow{x_i}).\Omega\rangle) + \sum_{u\neq b} u(\overrightarrow{x_i}).\Omega \;,\; b''(y''_1).\maltese + \sum_{u\neq b''} u(\overrightarrow{x_i}).\Omega\rangle$$

$$\rightarrow \quad \Big(b(y_1,y_2).\,(y_2|\overline{c}\langle \textstyle\sum u(\overrightarrow{x_i}).\Omega\rangle) + \sum_{u\neq b} u(\overrightarrow{x_i}).\Omega\Big) \mid \overline{b}\langle \textstyle\sum u(\overrightarrow{x_i}).\Omega \;,\; c(z_1).\maltese + \sum_{u\neq c} u(\overrightarrow{x_i}).\Omega\rangle$$

$$\rightarrow \quad \Big(c(z_1).\maltese + \sum_{u\neq c} u(\overrightarrow{x_i}).\Omega\Big) \mid \overline{c}\langle \textstyle\sum u(\overrightarrow{x_i}).\Omega\rangle$$

$$\rightarrow \quad \maltese$$

Term Normalization: a Linear Example



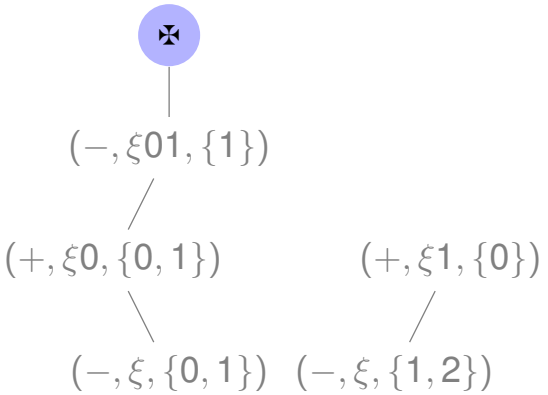
$$\left(a(x_1,x_2).\left(x_1 \mid \overline{b}\langle \sum u(\overrightarrow{x_i}).\Omega \ , \ c(z_1).\maltese + \sum_{u\neq c} u(\overrightarrow{x_i}).\Omega\rangle\right) + a'(x'_1,x'_2).\left(x'_1 \mid \overline{b'}\langle \sum u(\overrightarrow{x_i}).\Omega\rangle\right) + \ldots\right)$$
$$\mid \overline{a}\langle b(y_1,y_2).\left(y_2 | \overline{c}\langle \sum u(\overrightarrow{x_i}).\Omega\rangle\right) + \sum_{u\neq b} u(\overrightarrow{x_i}).\Omega \ , \ b''(y''_1).\maltese + \sum_{u\neq b''} u(\overrightarrow{x_i}).\Omega\rangle$$

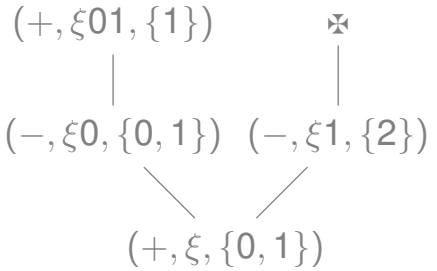
$$\rightarrow \quad \left(b(y_1,y_2).\left(y_2 | \overline{c}\langle \sum u(\overrightarrow{x_i}).\Omega\rangle\right) + \sum_{u\neq b} u(\overrightarrow{x_i}).\Omega\right) \mid \overline{b}\langle \sum u(\overrightarrow{x_i}).\Omega \ , \ c(z_1).\maltese + \sum_{u\neq c} u(\overrightarrow{x_i}).\Omega\rangle$$

$$\rightarrow \quad \left(c(z_1).\maltese + \sum_{u\neq c} u(\overrightarrow{x_i}).\Omega\right) \mid \overline{c}\langle \sum u(\overrightarrow{x_i}).\Omega\rangle$$

$$\rightarrow \quad \maltese$$

Term Normalization: a Linear Example





$$\left(a(x_1,x_2).\left(x_1 \mid \overline{b}\langle \sum u(\overrightarrow{x_i}).\Omega \ , \ c(z_1).\maltese + \sum_{u\neq c} u(\overrightarrow{x_i}).\Omega\rangle\right) + a'(x'_1,x'_2).\left(x'_1 \mid \overline{b'}\langle \sum u(\overrightarrow{x_i}).\Omega\rangle\right) + \ldots\right)$$
$$\mid \overline{a}\langle b(y_1,y_2).\left(y_2 | \overline{c}\langle \sum u(\overrightarrow{x_i}).\Omega\rangle\right) + \sum_{u\neq b} u(\overrightarrow{x_i}).\Omega \ , \ b''(y''_1).\maltese + \sum_{u\neq b''} u(\overrightarrow{x_i}).\Omega\rangle$$

$$\rightarrow \quad \left(b(y_1,y_2).\left(y_2 | \overline{c}\langle \sum u(\overrightarrow{x_i}).\Omega\rangle\right) + \sum_{u\neq b} u(\overrightarrow{x_i}).\Omega\right) \mid \overline{b}\langle \sum u(\overrightarrow{x_i}).\Omega \ , \ c(z_1).\maltese + \sum_{u\neq c} u(\overrightarrow{x_i}).\Omega\rangle$$

$$\rightarrow \quad \left(c(z_1).\maltese + \sum_{u\neq c} u(\overrightarrow{x_i}).\Omega\right) \mid \overline{c}\langle \sum u(\overrightarrow{x_i}).\Omega\rangle$$

$$\rightarrow \quad \maltese$$

Term Normalization in the Large

In the previous example:

- Variables are used linearly.
- They specify an immediate sub-cut.

The framework of c-ludics is more general:

- Variables may appear several times.
- Cuts may be put for later usage.

As in the following example...

Term Normalization: a Non-linear Example

$$\left(a(x).\left(x \mid \overline{b}\langle c().\left(x \mid \overline{b}\langle c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\rangle\right)\rangle\right)\right) \mid \overline{a}\langle b(y).(y \mid \overline{c}\langle\rangle)\rangle$$

$$\rightarrow \quad (b(y).(y \mid \overline{c}\langle\rangle)) \mid \overline{b}\langle c().\left((b(y).(y \mid \overline{c}\langle\rangle)) \mid \overline{b}\langle c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\rangle\right)$$

$$\rightarrow \quad c().\left((b(y).(y \mid \overline{c}\langle\rangle)) \mid \overline{b}\langle c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\rangle\right) \mid \overline{c}\langle\rangle$$

$$\rightarrow \quad b(y).(y \mid \overline{c}\langle\rangle) \mid \overline{b}\langle c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\rangle$$

$$\rightarrow \quad \left(c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\right) \mid \overline{c}\langle\rangle$$

$$\rightarrow \quad \maltese$$

Term Normalization: a Non-linear Example

$$\left(a(x).\left(x \mid \overline{b}\langle c().\left(x \mid \overline{b}\langle c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\rangle\right)\rangle\right)\right) \mid \overline{a}\langle b(y).(y \mid \overline{c}\langle\rangle)\rangle$$

$$\rightarrow \quad (b(y).(y \mid \overline{c}\langle\rangle)) \mid \overline{b}\langle c().\left((b(y).(y \mid \overline{c}\langle\rangle)) \mid \overline{b}\langle c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\rangle\right)$$

$$\rightarrow \quad c().\left((b(y).(y \mid \overline{c}\langle\rangle)) \mid \overline{b}\langle c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\rangle\right) \mid \overline{c}\langle\rangle$$

$$\rightarrow \quad b(y).(y \mid \overline{c}\langle\rangle) \mid \overline{b}\langle c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\rangle$$

$$\rightarrow \quad \left(c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\right) \mid \overline{c}\langle\rangle$$

$$\rightarrow \quad \maltese$$

Term Normalization: a Non-linear Example

$$\left(a(x).\left(x \mid \overline{b}\langle c().\left(x \mid \overline{b}\langle c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\rangle\right)\rangle\right)\right) \mid \overline{a}\langle b(y).(y \mid \overline{c}\langle\rangle)\rangle$$

$$\rightarrow \quad (b(y).(y \mid \overline{c}\langle\rangle)) \mid \overline{b}\langle c().\left((b(y).(y \mid \overline{c}\langle\rangle)) \mid \overline{b}\langle c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\rangle\right)$$

$$\rightarrow \quad c().\left((b(y).(y \mid \overline{c}\langle\rangle)) \mid \overline{b}\langle c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\rangle\right) \mid \overline{c}\langle\rangle$$

$$\rightarrow \quad b(y).(y \mid \overline{c}\langle\rangle) \mid \overline{b}\langle c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\rangle$$

$$\rightarrow \quad \left(c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\right) \mid \overline{c}\langle\rangle$$

$$\rightarrow \quad \maltese$$

Term Normalization: a Non-linear Example

$$\left(a(x).\left(x \mid \overline{b}\langle c().\left(x \mid \overline{b}\langle c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\rangle\right)\rangle\right)\right) \mid \overline{a}\langle b(y).(y \mid \overline{c}\langle\rangle)\rangle$$

$$\rightarrow \quad (b(y).(y \mid \overline{c}\langle\rangle)) \mid \overline{b}\langle c().\left((b(y).(y \mid \overline{c}\langle\rangle)) \mid \overline{b}\langle c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\rangle\right)$$

$$\rightarrow \quad c().\left((b(y).(y \mid \overline{c}\langle\rangle)) \mid \overline{b}\langle c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\rangle\right) \mid \overline{c}\langle\rangle$$

$$\rightarrow \quad b(y).(y \mid \overline{c}\langle\rangle) \mid \overline{b}\langle c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\rangle$$

$$\rightarrow \quad \left(c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\right) \mid \overline{c}\langle\rangle$$

$$\rightarrow \quad \maltese$$

Term Normalization: a Non-linear Example

$$\left(a(x).\left(x \mid \overline{b}\langle c().\left(x \mid \overline{b}\langle c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\rangle\right)\rangle\right)\right) \mid \overline{a}\langle b(y).(y \mid \overline{c}\langle\rangle)\rangle$$

$$\rightarrow \quad (b(y).(y \mid \overline{c}\langle\rangle)) \mid \overline{b}\langle c().\left((b(y).(y \mid \overline{c}\langle\rangle)) \mid \overline{b}\langle c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\rangle\right)$$

$$\rightarrow \quad c().\left((b(y).(y \mid \overline{c}\langle\rangle)) \mid \overline{b}\langle c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\rangle\right) \mid \overline{c}\langle\rangle$$

$$\rightarrow \quad b(y).(y \mid \overline{c}\langle\rangle) \mid \overline{b}\langle c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\rangle$$

$$\rightarrow \quad \left(c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\right) \mid \overline{c}\langle\rangle$$

$$\rightarrow \quad \maltese$$

Term Normalization: a Non-linear Example

$$\left(a(x).\left(x \mid \overline{b}\langle c().\left(x \mid \overline{b}\langle c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\rangle\right)\rangle\right)\right) \mid \overline{a}\langle b(y).(y \mid \overline{c}\langle\rangle)\rangle$$

$$\rightarrow \quad (b(y).(y \mid \overline{c}\langle\rangle)) \mid \overline{b}\langle c().\left((b(y).(y \mid \overline{c}\langle\rangle)) \mid \overline{b}\langle c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\rangle\right)$$

$$\rightarrow \quad c().\left((b(y).(y \mid \overline{c}\langle\rangle)) \mid \overline{b}\langle c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\rangle\right) \mid \overline{c}\langle\rangle$$

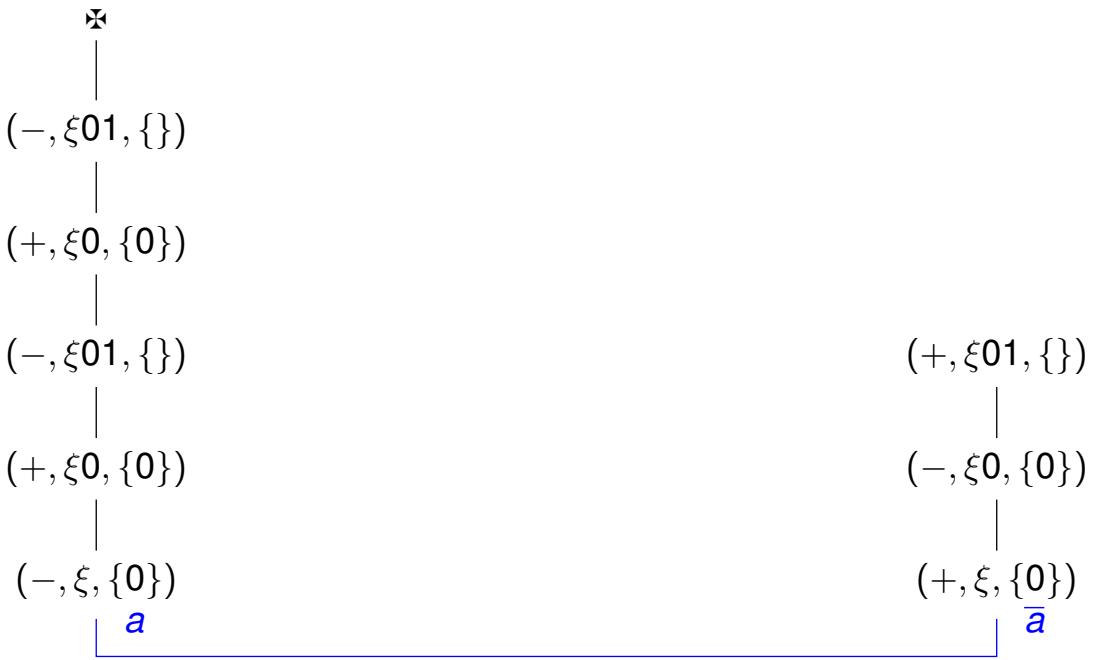
$$\rightarrow \quad b(y).(y \mid \overline{c}\langle\rangle) \mid \overline{b}\langle c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\rangle$$

$$\rightarrow \quad \left(c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\right) \mid \overline{c}\langle\rangle$$

$$\rightarrow \quad \maltese$$

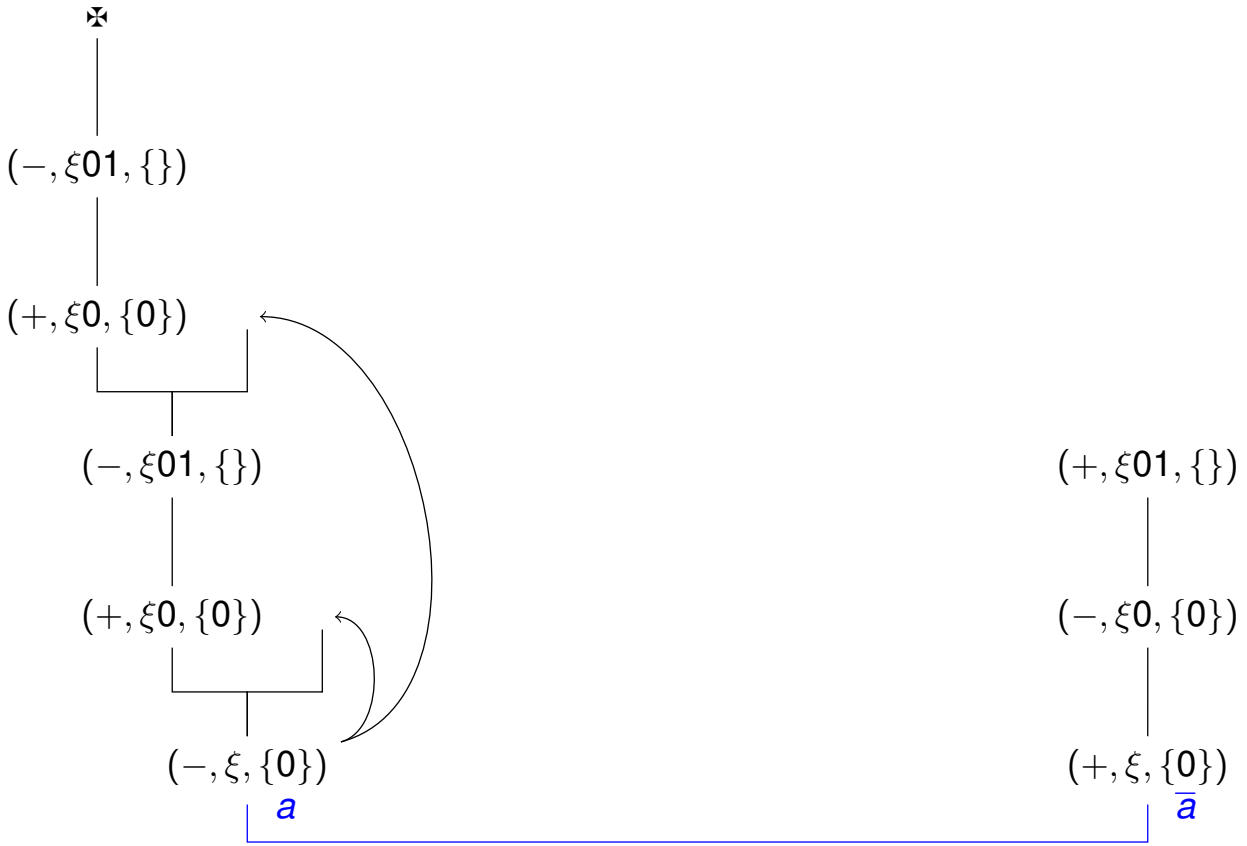
Term Normalization: Ludics style

$$\left(a(x).\left(x \mid \overline{b}\langle c().\left(x \mid \overline{b}\langle c().\maltese + \sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\rangle\right)\rangle\right)\right) \mid \overline{a}\langle b(y).(y \mid \overline{c}\langle\rangle)\rangle$$



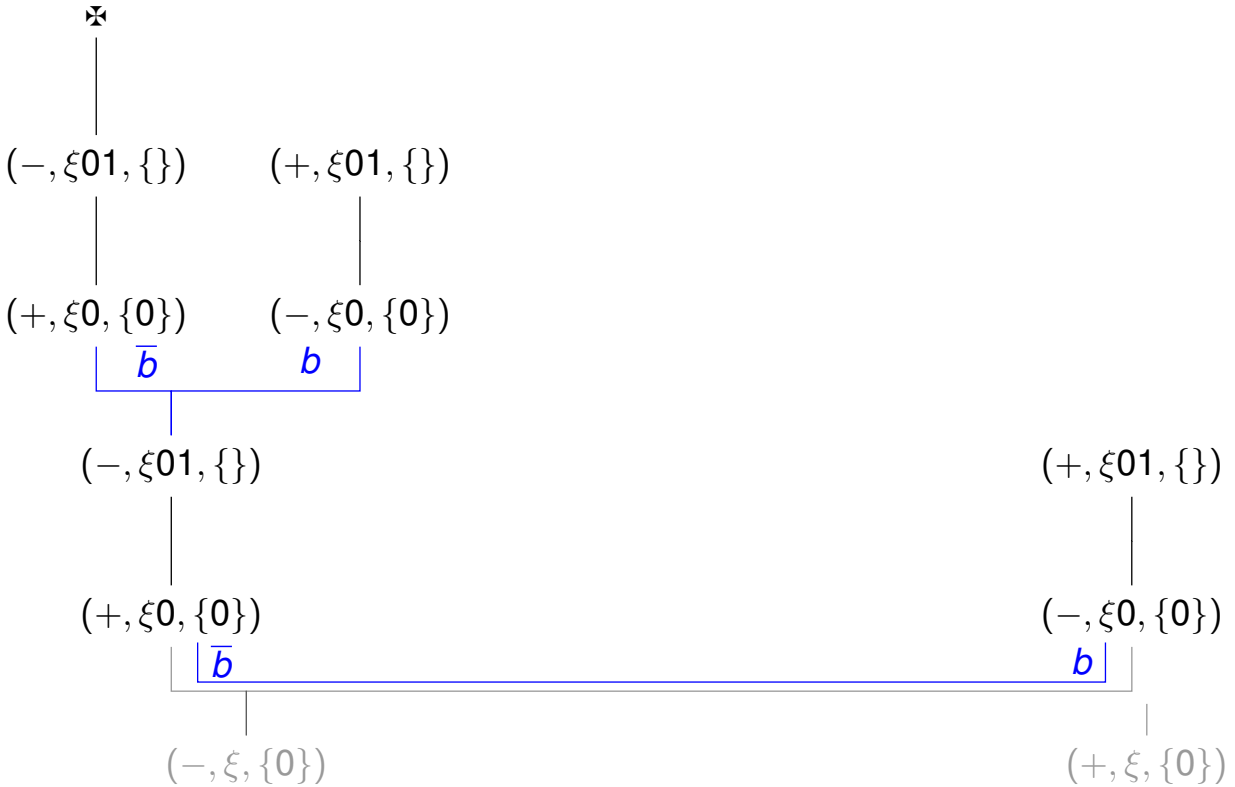
Term Normalization: Ludics style

$$\Big(a(x).\Big(x \mid \overline{b}\langle c().\Big(x \mid \overline{b}\langle c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\rangle\Big)\rangle\Big)\Big) \mid \overline{a}\langle b(y).(y \mid \overline{c}\langle\rangle)\rangle$$



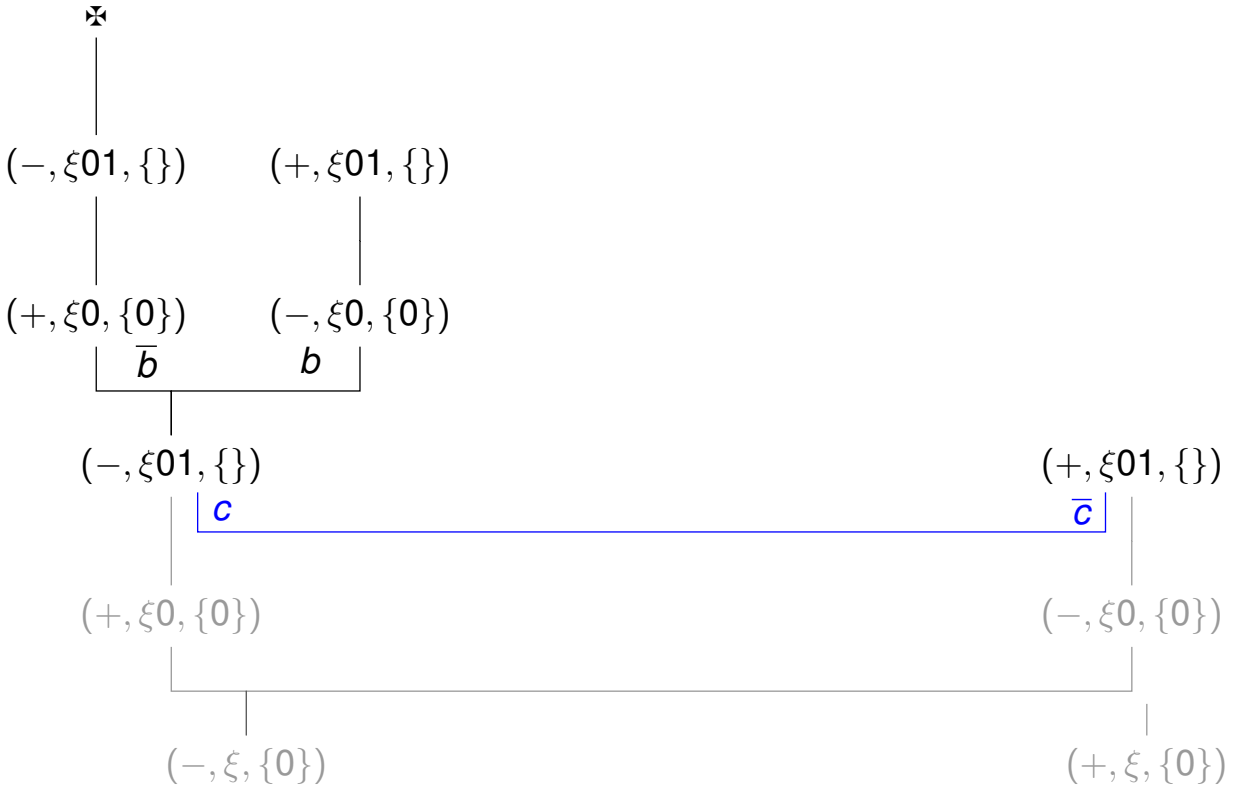
Term Normalization: Ludics style

$$\rightarrow \quad (b(y).(y \mid \overline{c}\langle\rangle)) \mid \overline{b}\langle c().\left((b(y).(y \mid \overline{c}\langle\rangle)) \mid \overline{b}\langle c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\rangle\right)$$



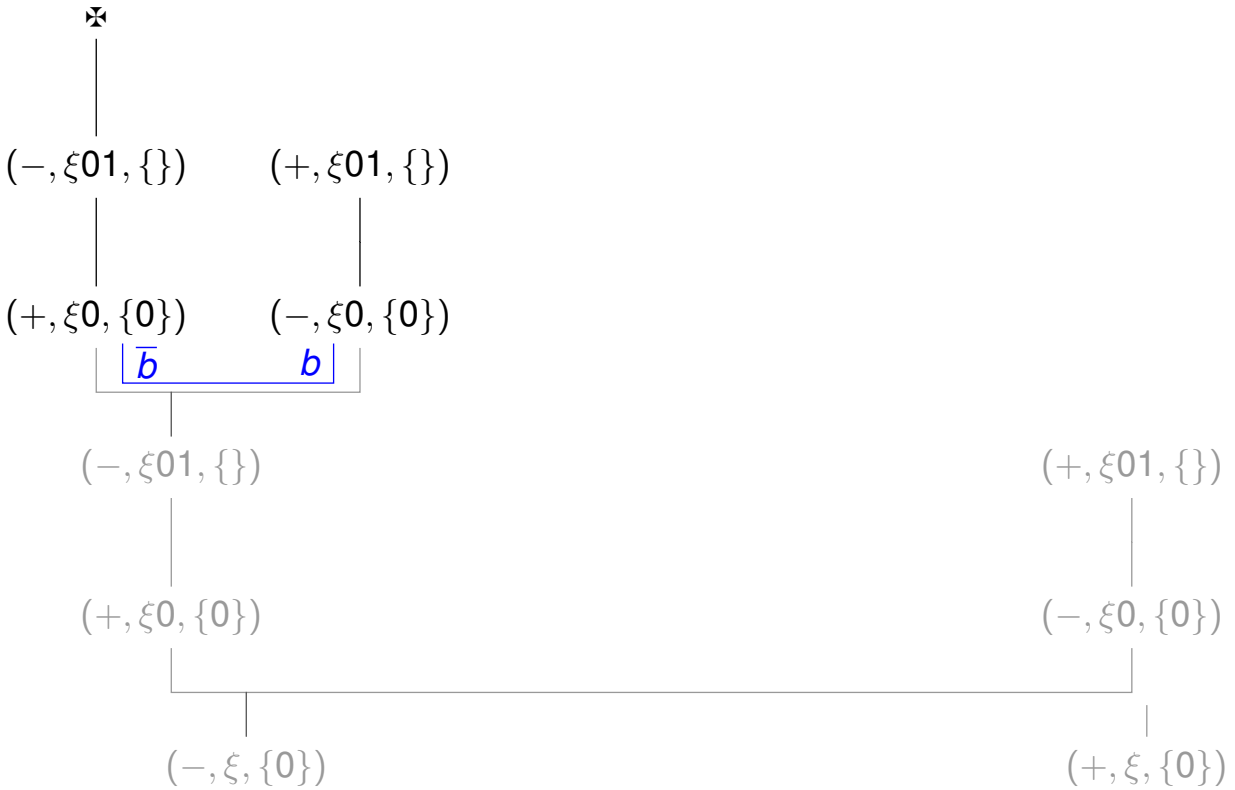
Term Normalization: Ludics style

$$\rightarrow \quad c().\left((b(y).(y \mid \overline{c}\langle\rangle)) \mid \overline{b}\langle c().\maltese + \sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\rangle\right) \mid \overline{c}\langle\rangle$$



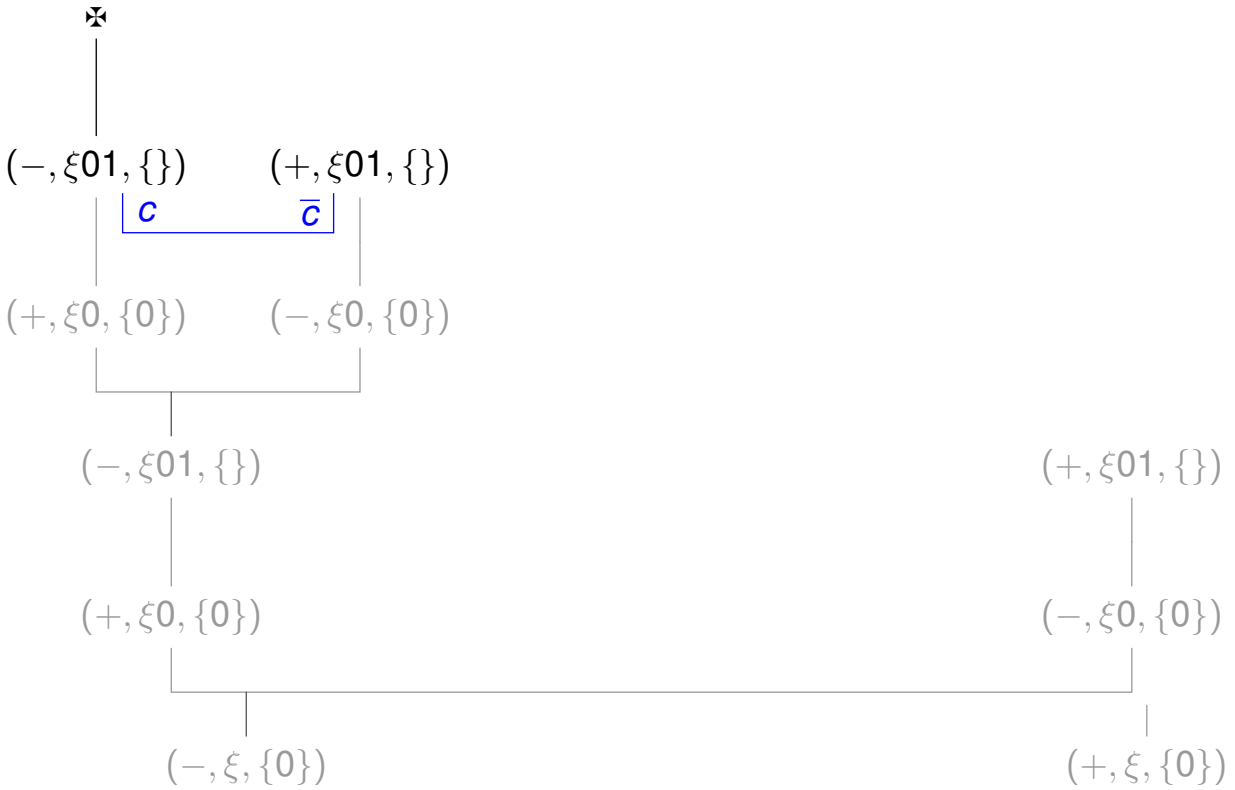
Term Normalization: Ludics style

$$\rightarrow \quad b(y).(y \mid \overline{c}\langle\rangle) \;\Big|\; \overline{b}\langle c().\maltese + \textstyle\sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\rangle$$



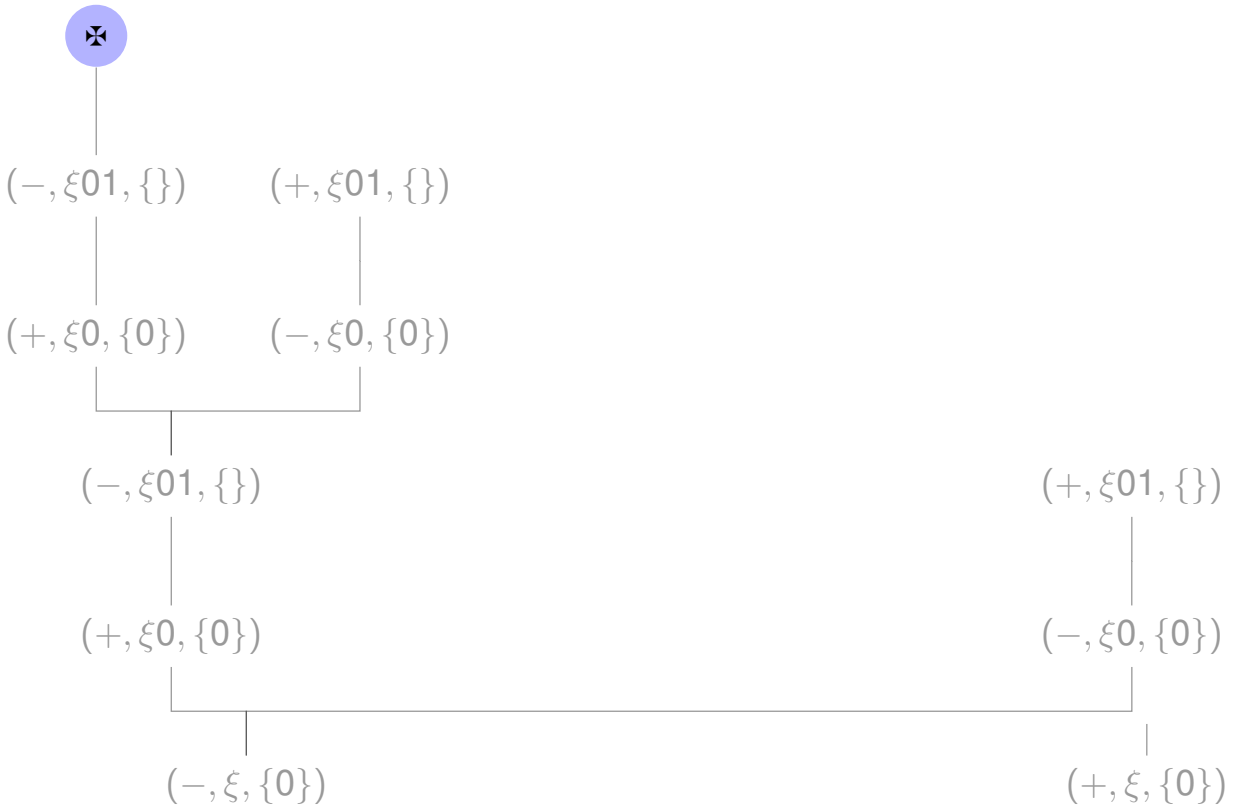
Term Normalization: Ludics style

$$\rightarrow \quad \left(c().\maltese + \sum_{u \neq c} u(\overrightarrow{x_i}).\Omega\right) \mid \overline{c}\langle\rangle$$



Term Normalization: Ludics style

$\rightarrow$ ✠



Generators

A **design generator** is given as a set of recursive definitions of positive or negative c-designs:

Definition

A design generator G is a triple $(\mathcal{P}, \mathcal{N}, g)$ such that:

- $\mathcal{P}$ and $\mathcal{N}$ are disjoint sets
- $\forall P \in \mathcal{P}$, $g(P)$ is
  - either ✠
  - or Ω
  - or $N_0 \mid \overline{u}\langle N_1, \ldots, N_n \rangle$ with $N_i \in \mathcal{N}$
- $\forall N \in \mathcal{N}$, $g(N)$ is
  - either a variable x
  - or $\sum u(\overrightarrow{x_u}).P_u$ with $P_u \in \mathcal{P}$

Generators: Reduction

Reduction rule and **normalization** may be defined on generators in the same way as they are on c-designs.

The reduction rule $\sum_u u(x_1 \dots x_{ar(u)}).P_u | \overline{u_0}\langle N_1 \dots N_{ar(u_0)} \rangle$ is extended by corecursion to c-designs defined by generators.

Generators: Reduction

The procedure is not effective as replacement may be infinite (as a design may contain an infinite number of the same variable).

Instead of the previous reduction, one may define reduction by means of **stack of environments**: substitutions are not explicitly done but kept in the environment.

Generators: an Example

Example

The fax may be generated by $(\{\}, \{N\}, \eta)$ where

$$\eta(N) = \sum u(\overrightarrow{x_u}).\left(N \mid \overline{u}\langle\overrightarrow{\eta(x_u)}\rangle\right)$$

Let P and N without cuts and identities,

- $[\![P[\overrightarrow{\eta(x_i)}/\overrightarrow{x_i}]]\!] = P$
- $[\![\eta(N[\overrightarrow{\eta(x_i)}/\overrightarrow{x_i}])]\!] = N$

Standard C-designs and Finite Automata (1)

A **standard** c-design (ie a *Girard's design*) is a c-design that is

- linear, cut-free, identity-free,
- $\neq \Omega$, with a finite set of free variables.

A **deterministic finite automaton** M is a tuple $(\Sigma, Q, \delta, q_0, Q_F)$:

- Σ is an alphabet
- Q is a finite set
- $\delta : Q \times \Sigma \to Q$ is the transition function,
  $q \xrightarrow{a} q'$ iff $\delta(q, a) = q'$
- $q_0 \in Q$ is the initial state and $Q_F \subset Q$ the set of final states

M **accepts** a word $a_1 \dots a_n$
if $\exists q_i \in Q$ s.t. $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \dots \xrightarrow{a_n} q_f \in Q_F$

Standard C-designs and Finite Automata (2)

Theorem (Terui)

- *Let M be a DFA, there exists a finitely generated positive* **standard** *c-design P such that M accepts w iff $[\![P[w^*/x]]\!] = \maltese$*
- *and vice-versa.*

The proof is based on the following:

- A name `nil` for the empty string
- For each state q, two c-designs q^+ and q^-
- $g(q_i^+) = x \mid \overline{\uparrow}\langle q_i^- \rangle$
- $g(q_i^-) = \sum_{a \in A} a(x).q_{j_a}^+ + \texttt{nil}().\maltese, \quad$ if $\forall a \in A, q_i \xrightarrow{a} q_{j_a}$ and $q_i \in Q_F$
- $g(q_i^-) = \sum_{a \in A} a(x).q_{j_a}^+ + \texttt{nil}().\Omega, \quad$ if $\forall a \in A, q_i \xrightarrow{a} q_{j_a}$ and $q_i \notin Q_F$

C-designs and Turing Machines

Theorem (Terui)

- *Let P be a finitely generated positive c-design, there exists a Turing Machine M such that M accepts w iff $[\![P[w^*/x]]\!] = \maltese$*
- *Let M be a Turing Machine, there exists a finitely generated positive* **identity-free linear** *c-design M such that M accepts w iff $[\![P[w^*/x]]\!] = \maltese$*

C-designs and Turing Machines

In other words:

- **Girard's finite designs recognize exactly regular languages**
- **Cuts are necessary to recover the full expressive power of Turing machines**

C-designs and Grammars: an Example

Example (Language $a^n b^n$, $n > 0$)

A string w is encoded as a negative c-design w^*:

- $\epsilon^* = \uparrow(x).x|\overline{\mathtt{nil}}\langle\rangle$
- $(a.w)^* = \uparrow(x).x|\overline{a}\langle w^*\rangle$

Use of a stack to postpone what remains to be done:

- $G[t] = M[\mathtt{nil}().\maltese, t]$
- $M[z,t] = t \mid \overline{\uparrow}\langle a(x).M'[z,x]\rangle$
- $M'[z,t] = t \mid \overline{\uparrow} \quad \langle$
  $b(x).(x|\overline{\uparrow}\langle z\rangle)$
  $+$
  $a(x).M[b(x).(x|\overline{\uparrow}\langle z\rangle), \uparrow(y).y|\overline{a}\langle x\rangle] \quad \rangle$

C-designs and Grammar

Formalisms used in Natural Language processing may be represented with C-designs.

What do we get? Remember the main objectives of Terui:

- Monism: One kind of object versus grammar/sentence, . . .
- Interaction:
  - Acceptance viewed as orthogonality
  - Symmetry of the process
- Existentialism: not necessarily closed objects

Formalization given in Ludics may be rewritten with C-designs:

Example (Beating)

With the question (a) *Have you stopped beating your father?*, one forces the following exchange:
– (e_0) *Did you beat your father ?*
– (e_1) *Yes*
– (e_2) *Have you stopped beating him ?*

Ludics

$$\dfrac{\dfrac{\dfrac{\xi.0.1.0 \vdash}{\vdash \xi.0.1} \quad \vdash \xi.0.2}{\xi.0 \vdash}}{\vdash \xi}$$

C-design with presupposition

$x \mid \overline{e_0} < e_1.t \mid \overline{e_2} < y > \ >$

Formalization given in Ludics may be rewritten with C-designs:

Example (Beating)

With the question (a) *Have you stopped beating your father?*, one forces the following exchange:
– (e_0) *Did you beat your father ?*
– (e_1) *Yes*
– (e_2) *Have you stopped beating him ?*

Ludics

$$\dfrac{\dfrac{\dfrac{\xi.0.1.0 \vdash}{\vdash \xi.0.1} \quad \vdash \xi.0.2}{\xi.0 \vdash}}{\vdash \xi}$$

C-design without presupposition

$$x \mid \overline{e_0} < e_1.t \mid \overline{e_2} < y > + z >$$

Example (Schopenhauer)

The fourth stratagem (again) If you want to draw a conclusion, you must not let it be foreseen, but you must get the premises admitted one by one, unobserved, mingling them here and there in your talk. [. . .] In this way you conceal your game until you have obtained all the admissions that are necessary, and so reach your goal by making a circuit.

Remember that a formalization of the speaker view point may be done with:

$$\dfrac{\dfrac{\downarrow [[\mathcal{D}_\alpha, \mathfrak{F}ax_{\alpha,\xi.1.0}]] \quad \vdots}{\vdash \xi.1.0} \quad \dfrac{\downarrow [[\mathcal{D}_\beta, \mathfrak{F}ax_{\beta,\xi.2.0}]] \quad \vdots}{\vdash \xi.2.0}}{\dfrac{\xi.1 \vdash \qquad \xi.2 \vdash \qquad \xi.3 \vdash}{\vdash \xi}}$$

$$\dfrac{\dfrac{\downarrow [[\mathcal{D}_\alpha, \mathfrak{F}ax_{\alpha,\xi.1.0}]]}{\vdots}}{\dfrac{\vdash \xi.1.0}{\xi.1 \vdash}} \quad \dfrac{\dfrac{\downarrow [[\mathcal{D}_\beta, \mathfrak{F}ax_{\beta,\xi.2.0}]]}{\vdots}}{\dfrac{\vdash \xi.2.0}{\xi.2 \vdash}} \quad \xi.3 \vdash}{\vdash \xi}$$

It may be rewritten as the following C-design:

$$x \mid \overline{e} < d(t).(Fax_t \mid \overline{a} < x_a >), q(t).(Fax_t \mid \overline{b} < x_b >), z >$$

$$\dfrac{\dfrac{\downarrow [[\mathcal{D}_\alpha, \mathfrak{F}ax_{\alpha,\xi.1.0}]]}{\vdots}}{\dfrac{\vdash \xi.1.0}{\xi.1 \vdash}} \quad \dfrac{\dfrac{\downarrow [[\mathcal{D}_\beta, \mathfrak{F}ax_{\beta,\xi.2.0}]]}{\vdots}}{\dfrac{\vdash \xi.2.0}{\xi.2 \vdash}} \quad \xi.3 \vdash}{\vdash \xi}$$

It may be rewritten as the following C-design:

$$x \mid \overline{e} < d(t).(Fax_t \mid \overline{a} < x_a >), q(t).(Fax_t \mid \overline{b} < x_b >), z >$$

Propositions A and B previously claimed are used now as arguments.

$$\dfrac{\dfrac{\downarrow [[\mathcal{D}_\alpha, \mathfrak{F}ax_{\alpha,\xi.1.0}]]}{\vdots}}{\dfrac{\vdash \xi.1.0}{\xi.1 \vdash}} \quad \dfrac{\dfrac{\downarrow [[\mathcal{D}_\beta, \mathfrak{F}ax_{\beta,\xi.2.0}]]}{\vdots}}{\dfrac{\vdash \xi.2.0}{\xi.2 \vdash}} \quad \xi.3 \vdash}{\vdash \xi}$$

It may be rewritten as the following C-design:

$$x \mid \overline{e} < d(t).(Fax_t \mid \overline{a} < x_a >), q(t).(Fax_t \mid \overline{b} < x_b >), z >$$

Adressee has to locate a free variable. If he still accepts propositions A and B, the only choice is z.

Petitio principii is a right place for using C-designs:
Proposition that has to be proved is already given (implicitly or not) in the premises !

Example

– I would like you to tell where that comes my daughter is dumb.
– There is nothing easier. This comes from what she has lost her speech.

We consider:

- e_1 : Your daughter is dumb
- e_2 : Your daughter has lost her speech

The design $\mathcal{I}$ corresponding to "*Your daughter is dumb because she lost her speech.*":

$$\mathcal{I} = e_0(x).x \mid \overline{e_1} < \mathcal{N} >$$

- $\overline{e_1}$ is the positive action corresponding to the claim of utterance e_1: *Your daughter is dumb*
- C-design $\mathcal{N}$ is a (dialog) design for justifying e_1

- Argumentations for e_1 and e_2 are *quite* identical.
- One can take the same c-design for $\mathcal{N}$ and $\mathcal{I}$: $e'_0(y).y \mid \overline{e_2} < \mathcal{I} >$

Hence the design modeling the second utterance may be an infinite C-design:

$$e_0(x).x \mid \overline{e_1} < e'_0(y).y \mid \overline{e_2} < e_0(z).z \mid \overline{e_1} < e'_0(t).t \mid \overline{e_2} < \cdots >>>>>$$

Hence there is no place (no usable free variable) to continue the exchange!