LUDICS AND LOGICAL COMPLETENESS

Geometry of Interaction, Traced Monoidal Categories and Implicit Complexity Workshop, Kyoto, Japan.
28 August 2009
Completeness (Gödel 1929)

Duality proof — countermodels:

- either there exists a proof $P$ such that $\vdash A$ is provable;
- or there exists a countermodel $\mathcal{M}$ such that $\mathcal{M} \models \neg A$.

One can imagine a debate on a general proposition $A$, where

- Player tries to justify $A$ by giving a proof;
- Opponent tries to refute it by giving a countermodel.
- The completeness theorem states that exactly one of them wins.
Completeness (Gödel 1929)

Duality proof — countermodels:

- either there exists a proof $P$ such that $\vdash A$ is provable;
- or there exists a countermodel $\mathcal{M}$ such that $\mathcal{M} \models \neg A$.

One can imagine a debate on a general proposition $A$, where

- Player tries to justify $A$ by giving a proof;
- Opponent tries to refute it by giving a countermodel.
- The completeness theorem states that exactly one of them wins.
Completeness (Gödel 1929)

Duality proof — countermodels:

- either there exists a proof $P$ such that $\vdash A$ is provable;
- or there exists a countermodel $\mathcal{M}$ such that $\mathcal{M} \models \neg A$.

One can imagine a debate on a general proposition $A$, where

- Player tries to justify $A$ by giving a proof;
- Opponent tries to refute it by giving a countermodel.
- The completeness theorem states that exactly one of them wins.
Completeness (Gödel 1929)

Duality proof — countermodels:

- either there exists a proof $P$ such that $\vdash A$ is provable;
- or there exists a countermodel $\mathcal{M}$ such that $\mathcal{M} \models \neg A$.

One can imagine a debate on a general proposition $A$, where

- Player tries to justify $A$ by giving a proof;
- Opponent tries to refute it by giving a countermodel.
- The completeness theorem states that exactly one of them wins.
Completeness (Gödel 1929)

Duality proof — countermodels:

- *either* there exists a proof $P$ such that $\vdash A$ is provable;
- *or* there exists a countermodel $\mathcal{M}$ such that $\mathcal{M} \models \neg A$.

One can *imagine* a debate on a general proposition $A$, where

- **Player** tries to justify $A$ by giving a proof;
- **Opponent** tries to refute it by giving a countermodel.

- The completeness theorem states that exactly one of them wins.
Completeness (Gödel 1929)

Duality proof — countermodels:

- either there exists a proof $P$ such that $\vdash A$ is provable;
- or there exists a countermodel $\mathcal{M}$ such that $\mathcal{M} \models \neg A$.

One can imagine a debate on a general proposition $A$, where

- Player tries to justify $A$ by giving a proof;
- Opponent tries to refute it by giving a countermodel.
- The completeness theorem states that exactly one of them wins.
Proofs, Models, Completeness

Proofs:

- Finite.
- **Provability** defined by induction on *proofs*.

Models:

- Infinite: arbitrary cardinality.
- Non standard models (Löwenheim — Skolem, Compactness Theorem).
- **Satisfiability** defined by induction on *formulas*.

Completeness proof:

- Nondeterministic principles: König Lemma (Schütte), Zorn’s Lemma (Henkin).

but ... there is no (clear) interaction between proofs and models ....
Proofs, Models, Completeness

Proofs:

- Finite.
- **Provability** defined by induction on **proofs**.

Models:

- Infinite: arbitrary cardinality.
- Non standard models (Löwenheim — Skolem, Compactness Theorem).
- **Satisfiability** defined by induction on **formulas**.

Completeness proof:

- Nondeterministic principles: König Lemma (Schütte), Zorn’s Lemma (Henkin).

but ... there is no (clear) interaction between proofs and models ....
Proofs, Models, Completeness

Proofs:

- Finite.
- **Provability** defined by induction on proofs.

Models:

- Infinite: arbitrary cardinality.
- Non standard models (Löwenheim — Skolem, Compactness Theorem).
- **Satisfiability** defined by induction on formulas.

Completeness proof:

- Nondeterministic principles: König Lemma (Schütte), Zorn’s Lemma (Henkin).

but ... there is no (clear) interaction between proofs and models ...
Proofs, Models, Completeness

Proofs:
- Finite.
- Provability defined by induction on proofs.

Models:
- Infinite: arbitrary cardinality.
- Non standard models (Löwenheim — Skolem, Compactness Theorem).
- Satisfiability defined by induction on formulas.

Completeness proof:
- Nondeterministic principles: König Lemma (Schütte), Zorn’s Lemma (Henkin).

but . . . there is no (clear) interaction between proofs and models . . . .
An interactive account of completeness

We are interested in (models of) proofs rather than provability.

**QUESTION** : What about the duality proofs — countermodels in Girard’s ludics?

**ANSWER** : Proofs and models are objects of the same kind (*designs*) only distinguished by their structural properties.
We are interested in (models of) proofs rather than provability.

**QUESTION**: What about the duality proofs — countermodels in Girard’s ludics?

**ANSWER**: Proofs and models are objects of the same kind (designs) only distinguished by their structural properties.
An interactive account of completeness

- We are interested in (models of) proofs rather than provability.

**QUESTION**: What about the duality **proofs** — **countermodels** in Girard’s ludics?

**ANSWER**: *Proofs and models are objects of the same kind* (**designs**) *only distinguished by their structural properties.*
Completeness revisited (ludics, game semantics)

For any logical behaviour $A$ (semantical type) and for any design $P$ either:

- either $P$ is a proof of $\vdash A$, or
- there exists a model $M \models A \perp$ which rejects $P$.

$M$ rejects $P$ means that $M \not\models P$ and hence, $P \notin A$.

Proofs : Finite, deterministic, $\Box$-free designs
Models : Infinite, nondeterministic, linear designs
Completeness proof : a real interaction between proofs and models.
For any logical behaviour $A$ (semantical type) and for any design $P$ either:

- either $P$ is a proof of $\vdash A$, or
- there exists a model $M \models A \bot$ which rejects $P$.

$M$ rejects $P$ means that $M \not\vDash P$ and hence, $P \notin A$.

Proofs: Finite, deterministic, $\forall$-free designs
Models: Infinite, nondeterministic, linear designs
Completeness proof: a real interaction between proofs and models.
Completeness revisited (ludics, game semantics)

For any logical behaviour $\mathbf{A}$ (semantical type) and for any design $P$ either:

- either $P$ is a proof of $\vdash \mathbf{A}$, or
- there exists a model $M \models \mathbf{A} \perp$ which rejects $P$.

$M$ rejects $P$ means that $M \not\models P$ and hence, $P \notin \mathbf{A}$.

Proofs : Finite, deterministic, $\Box$-free designs

Models : Infinite, nondeterministic, linear designs

Completeness proof : a real interaction between proofs and models.
Completeness revisited (ludics, game semantics)

For any logical behaviour $A$ (semantical type) and for any design $P$ either:

- *either* $P$ is a proof of $\vdash A$, or
- there exists a model $M \models A \bot$ which rejects $P$.

$M$ rejects $P$ means that $M \not\models P$ and hence, $P \not\in A$.

**Proofs**: Finite, deterministic, $\bowtie$-free designs

**Models**: Infinite, nondeterministic, linear designs

**Completeness proof**: a real interaction between proofs and models.
Completeness revisited (ludics, game semantics)

For any logical behaviour \( \mathbf{A} \) (semantical type) and for any design \( P \) either:

1. either \( P \) is a proof of \( \vdash \mathbf{A} \), or
2. there exists a model \( M \models \mathbf{A} \bot \) which rejects \( P \).

\( M \) rejects \( P \) means that \( M \not\models P \) and hence, \( P \notin \mathbf{A} \).

Proofs : Finite, deterministic, \( \boxtimes \)-free designs
Models : Infinite, nondeterministic, linear designs
Completeness proof : a real interaction between proofs and models.
In this talk:

▶ We show a completeness result: ludics is a model for a variant of (propositional) polarized linear logic (with exponentials) = a constructive version of classical propositional logic.

▶ ...but before that: we explain what ludics is!
In this talk:

- We show a completeness result: ludics is a model for a variant of (propositional) polarized linear logic (with exponentials) = a constructive version of classical propositional logic.
- ...but before that: we explain what ludics is!
What is ludics? (I)

A purely interactive approach to logic.

Ludics arose as the study of the interaction between syntax and syntax, typically in cut-elimination. It was necessary to replace syntax with something more geometrical, and this is why ludics lies between syntax and semantics, as a ‘semantics of syntax-as-syntax’, a monist explanation of logic. The thesis of ludics, which was already present in the programmatic paper [Towards a geometry of interaction], is that logic reflects the hidden geometrical properties of something.

What is ludics? (I)

A purely interactive approach to logic.

Ludics arose as the study of the interaction between syntax and syntax, typically in cut-elimination. It was necessary to replace syntax with something more geometrical, and this is why ludics lies between syntax and semantics, as a ‘semantics of syntax-as-syntax’, a monist explanation of logic. The thesis of ludics, which was already present in the programmatic paper [Towards a geometry of interaction], is that logic reflects the hidden geometrical properties of something.

Monism: An uniform framework in which syntax (proofs) and semantics (counterproofs, models) can be uniformly expressed.

Designs: Untyped paraproofs
- “untyped”: proofs from which the logical content has been almost erased.
- “para”: proofs which might contain errors and might be incomplete.

Interaction: Designs interact together via normalization which induces an orthogonality relation $\perp$ between designs in such a way that $P \perp M$ holds if the normalization of $P$ applied to $M$ terminates.
- A proof $P$ and “its model” $P\perp := \{N : P \perp N\}$.
- An automaton $A$ and a datum $D : A$ accepts $D$ iff $A \perp D$. 
What is ludics? (II)

- **Monism**: An uniform framework in which **syntax** (proofs) and **semantics** (counterproofs, models) can be uniformly expressed.

- **Designs**: Untyped paraproofs
  - “untyped” : proofs from which the logical content has been almost erased.
  - “para” : proofs which might contain errors and might be incomplete.

- **Interaction**: Designs interact together via **normalization** which induces an **orthogonality relation** \( \perp \) between designs in such a way that \( P \perp M \) holds if the normalization of \( P \) applied to \( M \) terminates.
  - A proof \( P \) and “its model” \( P \perp := \{N : P \perp N\} \).
  - An automaton \( A \) and a datum \( D : A \) accepts \( D \) iff \( A \perp D \).
What is ludics? (II)

- **Monism**: An uniform framework in which *syntax* (proofs) and *semantics* (counterproofs, models) can be uniformly expressed.

- **Designs**: Untyped paraproofs
  - “untyped”: proofs from which the logical content has been almost erased.
  - “para”: proofs which might contain errors and might be incomplete.

- **Interaction**: Designs interact together via normalization which induces an orthogonality relation \( \perp \) between designs in such a way that \( P \perp M \) holds if the normalization of \( P \) applied to \( M \) terminates.
  - A proof \( P \) and “its model” \( P \perp := \{ N : P \perp N \} \).
  - An automaton \( A \) and a datum \( D : A \) accepts \( D \) iff \( A \perp D \).
What is ludics? (II)

- **Monism**: An uniform framework in which *syntax* (proofs) and *semantics* (counterproofs, models) can be uniformly expressed.

- **Designs**: Untyped paraproofs
  - “untyped” : proofs from which the logical content has been almost erased.
  - “para” : proofs which might contain errors and might be incomplete.

- **Interaction** : Designs interact together via normalization which induces an orthogonality relation $\perp$ between designs in such a way that $P \perp M$ holds if the normalization of $P$ applied to $M$ terminates.
  - A proof $P$ and “its model” $P \perp := \{ N : P \perp N \}$.
  - An automaton $A$ and a datum $D : A$ accepts $D$ iff $A \perp D$. 
What is ludics? (II)

- **Monism**: An uniform framework in which *syntax* (proofs) and *semantics* (counterproofs, models) can be uniformly expressed.

- **Designs**: Untyped paraproofs
  - “untyped”: proofs from which the logical content has been almost erased.
  - “para”: proofs which might contain errors and might be incomplete.

- **Interaction**: Designs interact together via normalization which induces an orthogonality relation \( \perp \) between designs in such a way that \( P \perp M \) holds if the normalization of \( P \) applied to \( M \) terminates.
  - A proof \( P \) and “its model” \( P^\perp := \{ N : P \perp N \} \).
  - An automaton \( A \) and a datum \( D : A \) accepts \( D \) iff \( A \perp D \).
A dialogue between the automata and the datum.

\[
A := x \langle \text{zero}.OK + \text{succ}(x).A \rangle \\
0 := S(x).x \langle \text{zero} \rangle \\
N + 1 := S(x).x \langle \text{succ}(N) \rangle \\
A[0/x] = (S(x).x \langle \text{zero} \rangle) \langle \text{zero}.OK + \text{succ}(x).A \rangle \\
\rightarrow (\text{zero}.OK + \text{succ}(x).A) \langle \text{zero} \rangle \\
\rightarrow \text{OK}.
\]

\[
A[N + 1/x] = (S(x).x \langle \text{succ}(N) \rangle) \langle \text{zero}.OK + \text{succ}(x).A \rangle \\
\rightarrow (\text{zero}.OK + \text{succ}(x).A) \langle \text{succ}(N) \rangle \\
\rightarrow A[N/x].
\]
Example

\[ A = \ \xrightarrow{\text{start}} S \xrightarrow{0} OK \]

\[ n = \underbrace{ssss\ldots s}_n \] 0

A dialogue between the automata and the datum.

\[
\begin{align*}
A & := x \overline{S}(\text{zero}.OK + \text{succ}(x).A) \\
0 & := S(x).x \overline{\text{zero}} \\
N + 1 & := S(x).x \overline{\text{succ}}(N) \\
\end{align*}
\]

\[
\begin{align*}
A[0/x] & = (S(x).x \overline{\text{zero}}) \overline{S}(\text{zero}.OK + \text{succ}(x).A) \\
\rightarrow & (\text{zero}.OK + \text{succ}(x).A) \overline{\text{zero}} \\
\rightarrow & OK.
\end{align*}
\]

\[
\begin{align*}
A[N + 1/x] & = (S(x).x \overline{\text{succ}}(N)) \overline{S}(\text{zero}.OK + \text{succ}(x).A) \\
\rightarrow & (\text{zero}.OK + \text{succ}(x).A) \overline{\text{succ}}(N) \\
\rightarrow & A[N/x].
\end{align*}
\]
Example

A dialogue between the automata and the datum.

\[
A := x | S (\text{zero}.OK + \text{succ}(x).A) \\
0 := S(x).x | \text{zero} \\
N + 1 := S(x).x | \text{succ}(N) \\
\]

\[
A[0/x] = (S(x).x | \text{zero}) | S(\text{zero}.OK + \text{succ}(x).A) \\
\quad \rightarrow (\text{zero}.OK + \text{succ}(x).A) | \text{zero} \\
\quad \rightarrow \text{OK}. \\
\]

\[
A[N + 1/x] = (S(x).x | \text{succ}(N)) | S(\text{zero}.OK + \text{succ}(x).A) \\
\quad \rightarrow (\text{zero}.OK + \text{succ}(x).A) | \text{succ}(N) \\
\quad \rightarrow A[N/x]. \\
\]

A \[=\] \begin{tikzpicture}

\node[state, initial] (S) at (0,0) {$S$};
\node[state, accepting] (OK) at (2,0) {$OK$};

\draw[->] (S) edge [loop above] node {$s$} (S)
(S) edge node [below] {$0$} (OK)
(OK) edge node [above] {$n = \underbrace{s\ldots s}_{n \text{ times}}$} (OK);
\end{tikzpicture}
What is ludics? (III)

The core of ludics: focalization

<table>
<thead>
<tr>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>⊗</td>
<td>⊰</td>
</tr>
<tr>
<td>⊕</td>
<td>&amp;</td>
</tr>
<tr>
<td>0</td>
<td>⊤</td>
</tr>
<tr>
<td>1</td>
<td>⊥</td>
</tr>
<tr>
<td>?</td>
<td>⊽</td>
</tr>
</tbody>
</table>

- Negative = reversible, deterministic: \( \vdash \Sigma, A, B \quad \vdash \Sigma, A \Rightarrow A \)

- Positive = irreversible, nondeterministic: \( \vdash \Sigma, A \quad \vdash \Sigma_1, A \quad \vdash \Sigma_2, B \quad \vdash \Sigma, A \otimes B \)
What is ludics? (III)

The core of ludics: focalization

<table>
<thead>
<tr>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>⊗</td>
<td>⊨</td>
</tr>
<tr>
<td>⊕</td>
<td>&amp;</td>
</tr>
<tr>
<td>0</td>
<td>⊤</td>
</tr>
<tr>
<td>1</td>
<td>⊥</td>
</tr>
<tr>
<td>?</td>
<td>!</td>
</tr>
</tbody>
</table>

- **Negative** = reversible, deterministic: \( \vdash \Sigma, A, B \quad \vdash \Sigma, A \uparrow A \)

- **Positive** = irreversible, nondeterministic: \( \vdash \Sigma_1, A \quad \vdash \Sigma_2, B \quad \vdash \Sigma, A \otimes B \)
What is ludics? (III)

The core of ludics: focalization

<table>
<thead>
<tr>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>⊗</td>
<td>⊨</td>
</tr>
<tr>
<td>⊕</td>
<td>&amp;</td>
</tr>
<tr>
<td>0</td>
<td>⊥</td>
</tr>
<tr>
<td>1</td>
<td>!</td>
</tr>
</tbody>
</table>

- **Negative** = reversible, deterministic: \[ \vdash \Sigma, A, B \quad \vdash \Sigma, A \neg A \]
- **Positive** = irreversible, nondeterministic: \[ \vdash \Sigma, A \otimes B \]
What is ludics? (III)

The core of ludics: focalization

<table>
<thead>
<tr>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>⊗</td>
<td>¬</td>
</tr>
<tr>
<td>⊕</td>
<td>&amp;</td>
</tr>
<tr>
<td>0</td>
<td>⊥</td>
</tr>
<tr>
<td>1</td>
<td>!</td>
</tr>
</tbody>
</table>

- **Negative** = reversible, deterministic: \[ \vdash \sum, A, B \quad \vdash \sum, A \otimes A \]

- **Positive** = irreversible, nondeterministic: \[ \vdash \sum_1, A \quad \vdash \sum_2, B \quad \vdash \sum, A \otimes B \]
What is ludics? (IV)

▷ \( \vdash N_1, \ldots, N_m, P_1, \ldots, P_n \) choose a negative formula (if any) and keep decomposing until one get to atoms or positive subformulas;

▷ \( \vdash P_1, \ldots, P_n \) choose a positive formula and keep decomposing it up to atoms or negative subformulas.

(Andreoli 92) The focalization discipline is a complete proof-search strategy.
What is ludics? (IV)

- \( \vdash N_1, \ldots, N_m, P_1, \ldots, P_n \) choose a negative formula (if any) and keep decomposing until one get to atoms or positive subformulas;
- \( \vdash P_1, \ldots, P_n \) choose a positive formula and keep decomposing it up to atoms or negative subformulas.

(Andreoli 92) The focalization discipline is a complete proof-search strategy.
What is ludics? (V)

**Synthetic connectives**

- Focalization allows **synthetic connectives**: clusters of connectives of the same polarity.
- $N \otimes (M_1 \oplus M_2)$ can be written as $\bar{a}\langle N, M_1, M_2 \rangle$. Think $\bar{a}$ as a “generalized” ternary connective $\_ \otimes (\_ \oplus \_)$.

- Alternation of positive and negative layers.
What is ludics? (V)

Synthetic connectives

- Focalization allows **synthetic connectives**: clusters of connectives of the same polarity.

- $N \otimes (M_1 \oplus M_2)$ can be written as $\bar{a}\langle N, M_1, M_2 \rangle$. Think $\bar{a}$ as a “generalized” ternary connective $\_ \otimes (\_ \oplus \_)$.

$$
\begin{align*}
\Sigma_1, N & \vdash \Sigma_2, M_1 & \vdash \Sigma_2, M_1 \oplus M_2 & \otimes_1 \\
\Sigma_1, N & \vdash \Sigma_2, M_1 \oplus M_2 & \vdash \Sigma, N \otimes (M_1 \oplus M_2) & \otimes \\
\Sigma_1, N & \vdash \Sigma_2, M_1 & \vdash \Sigma, N \otimes (M_1 \oplus M_2) & \otimes_1 \\
\Sigma_1, N & \vdash \Sigma_2, M_2 & \vdash \Sigma_2, M_1 \oplus M_2 & \otimes_2 \\
\Sigma_1, N & \vdash \Sigma_2, M_1 \oplus M_2 & \vdash \Sigma, N \otimes (M_1 \oplus M_2) & \otimes \\
\Sigma_1, N & \vdash \Sigma_2, M_2 & \vdash \Sigma, N \otimes (M_1 \oplus M_2) & \otimes_2
\end{align*}
$$

- Alternation of positive and negative layers.
What is ludics? (V)

Synthetic connectives

- Focalization allows **synthetic connectives**: clusters of connectives of the same polarity.
- \( N \otimes (M_1 \oplus M_2) \) can be written as \( \overline{a} \langle N, M_1, M_2 \rangle \). Think \( \overline{a} \) as a “generalized” ternary connective \( __ \otimes (__ \oplus __) \).

\[
\begin{align*}
\Sigma_1, N & \vdash \Sigma_2, M_1 \\
\Sigma_1, N & \vdash \Sigma_2, M_1 \oplus M_2 \\
\Sigma_1, N & \vdash \Sigma_2, M_1 \\
\Sigma_1, N & \vdash \Sigma_2, M_2
\end{align*}
\]

\[
\begin{align*}
\Sigma_1, N & \vdash \Sigma_2, M_1 \oplus M_2 \\
\Sigma_1, N & \vdash \Sigma, N \otimes (M_1 \oplus M_2) \\
\Sigma_1, N & \vdash \Sigma, N \otimes (M_1 \oplus M_2) \\
\Sigma_1, N & \vdash \Sigma, N \otimes (M_1 \oplus M_2)
\end{align*}
\]

- Alternation of positive and negative layers.
Computational ludics (I)

**Designs** (Terui 08) \(\approx\) **infinitary** lambda terms (Böhm trees) + **named** applications + named and **superimposed** abstractions.

cf.

- the "concrete syntax" (Curien 05) \(\approx\) abstract Böhm trees,
- the correspondence with linear \(\pi\)-calculus (Faggian-Piccolo 07).

**Signature:** \(\mathcal{A} = (A, \text{ar})\)

- \(A\) is a set of **names**,
- \(\text{ar} : A \rightarrow \mathbb{N}\) gives an **arity** to each name.
Computational ludics (II)

The set of designs is coinductively defined by:

\[ P ::= \begin{array}{c|c}
\mathbf{✠} & \text{Daimon} \\
\mid & \\
\mathbf{Ω} & \text{Divergence} \\
\mid & \\
N_0 | \overline{a} \langle N_1, \ldots, N_n \rangle & \text{Application} \\
\end{array} \]

\[ N ::= x \]

\[ \sum a(\vec{x}). P_a \]

\[ \text{where } ar(a) = n, \vec{x} = x_1, \ldots, x_n \]

\[ \sum a(\vec{x}). P_a \text{ is built from } \{ a(\vec{x}). P_a \}_{a \in A}. \]

Compare it with:

\[ P ::= (N_0)N_1 \ldots N_n \]

\[ N ::= x | \lambda x_1 \ldots x_n . P \]
The set of designs is coinductively defined by:

\[ P ::= \begin{array}{ll}
\top & \text{Daimon} \\
\Omega & \text{Divergence} \\
N_0 | \bar{a}\langle N_1, \ldots, N_n \rangle & \text{Application} \\
\end{array} \]

\[ N ::= \begin{array}{ll}
x & \text{Variable} \\
\sum a(\bar{x}).P_a & \text{Abstraction} \\
\end{array} \]

where \(ar(a) = n, \bar{x} = x_1, \ldots, x_n\)

\(\sum a(\bar{x}).P_a\) is built from \(\{a(\bar{x}).P_a\}_{a \in A}\).

Compare it with:

\[ P ::= (N_0)N_1 \ldots N_n \]
\[ N ::= x | \lambda x_1 \cdots x_n.P \]
Reduction

- $\Omega$ allows partial branching:

$$a(\vec{x}).P + b(\vec{y}).Q := a(\vec{x}).P + b(\vec{y}).Q + c(\vec{z}).\Omega + d(\vec{z}).\Omega + \cdots$$

- Reduction rule:

$$(\sum a(x_1, \ldots, x_n).P_a) |a\langle N_1, \ldots, N_n \rangle \rightarrow P_a[N_1/x_1, \ldots, N_n/x_n].$$

- Compare it with

$$(\lambda x_1 \cdots x_n.P)N_1 \cdots N_n \rightarrow P[N_1/x_1, \ldots, N_n/x_n]$$
Reduction

- Ω allows partial branching:

\[ a(\vec{x}).P + b(\vec{y}).Q := a(\vec{x}).P + b(\vec{y}).Q + c(\vec{z}).\Omega + d(\vec{z}).\Omega + \cdots \]

- Reduction rule:

\[ \left( \sum a(x_1, \ldots, x_n).P_a \right) |\overline{a}\langle N_1, \ldots, N_n \rangle \longrightarrow P_a[N_1/x_1, \ldots, N_n/x_n]. \]

- Compare it with

\[ (\lambda x_1 \cdots x_n.P)N_1 \cdots N_n \longrightarrow P[N_1/x_1, \ldots, N_n/x_n] \]
Reduction

- $\Omega$ allows partial branching:
  
  \[ a(\vec{x}).P + b(\vec{y}).Q := a(\vec{x}).P + b(\vec{y}).Q + c(\vec{z}).\Omega + d(\vec{z}).\Omega + \cdots \]

- Reduction rule:
  
  \[ (\sum a(x_1, \ldots, x_n).P_a) |\overline{a}\langle N_1, \ldots, N_n \rangle \rightarrow P_a[N_1/x_1, \ldots, N_n/x_n]. \]

- Compare it with
  
  \[ (\lambda x_1 \cdots x_n.P)N_1 \cdots N_n \rightarrow P[N_1/x_1, \ldots, N_n/x_n] \]
Orthogonality

A positive design $P$ is one of the following forms:

\[
\begin{align*}
x | \overline{a} \langle N_1, \ldots, N_n \rangle \\
(\sum a(\vec{x}).P_a) | \overline{a} \langle N_1, \ldots, N_n \rangle \\
\times \\
\Omega
\end{align*}
\]

Head normal form 
Cut 
Daimon 
Divergence

- **Dichotomy**: For any closed positive design $P$, $P \longrightarrow^* \times$ or diverges.

- **Orthogonality**: Suppose $fv(P) \subseteq \{x_0\}$ and $fv(M) = \emptyset$.

\[
P \perp M \iff P[M/x_0] \longrightarrow^* \times.
\]

Compare it with:

\[
\pi \perp \pi' \iff \pi \pi' \text{ is nilpotent}.
\]
Orthogonality

A positive design $P$ is one of the following forms:

\[
x | \overline{a} \langle N_1, \ldots, N_n \rangle \\
(\sum a(\vec{x}).P_a) | \overline{a} \langle N_1, \ldots, N_n \rangle
\]

Head normal form
Cut
Daimon
Divergence

- **Dichotomy**: For any **closed** positive design $P$,

  \[ P \xrightarrow{\ast} \bigodot \text{ or diverges.} \]

- **Orthogonality**: Suppose $fv(P) \subseteq \{x_0\}$ and $fv(M) = \emptyset$.

  \[ P \perp M \iff P[M/x_0] \xrightarrow{\ast} \bigodot. \]

Compare it with:

\[ \pi \perp \pi' \iff \pi \pi' \text{ is nilpotent.} \]
Orthogonality

A positive design $P$ is one of the following forms:

\[
x \mid \overline{a} \langle N_1, \ldots, N_n \rangle \quad \text{Head normal form}
\]

\[
(\sum a(x).P_a) \mid \overline{a} \langle N_1, \ldots, N_n \rangle \quad \text{Cut}
\]

\[
\times
\]

\[
\Omega
\quad \text{Daimon}
\]

\[
\Delta
\quad \text{Divergence}
\]

▶ Dichotomy: For any closed positive design $P$,

\[
P \xrightarrow{\ast} \times \text{ or diverges.}
\]

▶ Orthogonality: Suppose $fv(P) \subseteq \{x_0\}$ and $fv(M) = \emptyset$.

\[
P \perp M \iff P[M/x_0] \xrightarrow{\ast} \times.
\]

Compare it with:

\[
\pi \perp \pi' \iff \pi \pi' \text{ is nilpotent.}
\]
Orthogonality

A positive design $P$ is one of the following forms:

- $x | \bar{a} \langle N_1, \ldots, N_n \rangle$ (Head normal form)
- $(\sum a(x).P_a) | \bar{a} \langle N_1, \ldots, N_n \rangle$ (Cut)
- $\varpi$ (Daimon)
- $\Omega$ (Divergence)

- **Dichotomy:** For any closed positive design $P$,
  
  $P \xrightarrow{*} \varpi$ or diverges.

- **Orthogonality:** Suppose $fv(P) \subseteq \{x_0\}$ and $fv(M) = \emptyset$.

  $$P \perp M \iff P[M/x_0] \xrightarrow{*} \varpi.$$ 

  Compare it with:

  $$\pi \perp \pi' \iff \pi \pi' \text{ is nilpotent.}$$
Example: termination

\[
A = \begin{array}{c}
\text{start} \\
\downarrow \\
s \end{array} S \begin{array}{c}
\text{0} \\
\downarrow \\
\times \end{array} \quad n = \underbrace{ssss \ldots s}_n 0
\]

\[
A \quad 0 \quad N + 1
\]

\[
A[0/x] = (S(x).x|\text{zero})|S(\text{zero.} \times + \text{succ}(x).A) \\
\rightarrow (\text{zero.} \times + \text{succ}(x).A)|\text{zero} \\
\rightarrow \times.
\]

\[
A[N + 1/x] = (S(x).x|\text{succ}(N))|S(\text{zero.} \times + \text{succ}(x).A) \\
\rightarrow (\text{zero.} \times + \text{succ}(x).A)|\text{succ}(N) \\
\rightarrow A[N/x].
\]
Example: termination

\[ A = \begin{array}{c}
\text{start} \rightarrow \bullet \rightarrow 0 \\
S \quad s
\end{array} \quad n = \underbrace{\text{ssss} \ldots \text{s0}}_{n \text{ times}} \]

\[
A := x | S \langle \text{zero} \times + \text{succ}(x).A \rangle \\
0 := S(x) . x | \text{zero} \\
N + 1 := S(x) . x | \text{succ} \langle N \rangle
\]

\[
A[0/x] = (S(x) . x | \text{zero}) | S \langle \text{zero} \times + \text{succ}(x).A \rangle \\
\quad \rightarrow (\text{zero} \times + \text{succ}(x).A) | \text{zero} \\
\quad \rightarrow \times.
\]

\[
A[N + 1/x] = (S(x) . x | \text{succ} \langle N \rangle) | S \langle \text{zero} \times + \text{succ}(x).A \rangle \\
\quad \rightarrow (\text{zero} \times + \text{succ}(x).A) | \text{succ} \langle N \rangle \\
\quad \rightarrow A[N/x].
\]
Example: termination

\[ A = \xrightarrow{S} 0 \quad n = \overbrace{s \ldots s}^{n \text{ times}} 0 \]

\[
A := x | S\langle \text{zero.} \ast + \text{succ}(x) \cdot A \rangle \\
0 := S(x) \cdot x | \text{zero} \\
N + 1 := S(x) \cdot x | \text{succ} \langle N \rangle \\
A[0/x] = (S(x) \cdot x | \text{zero}) | S\langle \text{zero.} \ast + \text{succ}(x) \cdot A \rangle \\
\quad \rightarrow (\text{zero.} \ast + \text{succ}(x) \cdot A) | \text{zero} \\
\quad \rightarrow \ast.
\]

\[
A[N + 1/x] = (S(x) \cdot x | \text{succ} \langle N \rangle) | S\langle \text{zero.} \ast + \text{succ}(x) \cdot A \rangle \\
\quad \rightarrow (\text{zero.} \ast + \text{succ}(x) \cdot A) | \text{succ} \langle N \rangle \\
\quad \rightarrow A[N/x].
\]
Example: nontermination

\[ P := x | \overline{a} \langle N \rangle \]
\[ N := a(x).P \]
\[ M := b(y).P \]

\[ P[N/x] = (a(x).P) | \overline{a} \langle N \rangle \]
\[ \rightarrow P[N/x]. \]

\[ P[M/x] = (b(x).P) | \overline{a} \langle N \rangle \]
\[ \rightarrow \Omega. \]
Example: nontermination

\[
P := x \mid \overline{a} \langle N \rangle \\
N := a(x).P \\
M := b(y).P
\]

\[
P[N/x] = (a(x).P) \mid \overline{a} \langle N \rangle \\
\quad \rightarrow P[N/x].
\]

\[
P[M/x] = (b(x).P) \mid \overline{a} \langle N \rangle \\
\quad \rightarrow \Omega.
\]
Ludics and Game Semantics

**Ludics**

Untyped strategies (*designs*)

\[ \perp \perp \]

Types (*Behaviours*)

**Game Semantics**

Typed *strategies*

Types (*Arenas, Games*)

- **Game Semantics**: All strategies are typed. Types GUARANTEE that strategies compose well.
- **Ludics**: Strategies are untyped (all given on a universal arena) Strategies can ALWAYS interact with each other, and interaction may terminate well (\( \perp \)) or not (deadlock, \( \Omega \))
Ludics and Game Semantics

**Ludics**

Untyped strategies (*designs*)

\[\perp \perp\]

Types (*Behaviours*)

**Game Semantics**

Typed *strategies*

Types (*Arenas, Games*)

➤ **Game Semantics**: All strategies are typed. Types GUARANTEE that strategies compose well.

➤ **Ludics**: Strategies are untyped (all given on a universal arena) Strategies can ALWAYS interact with each other, and interaction may terminate well (\(\perp\)) or not (deadlock, \(\Omega\))
Ludics and Game Semantics

Ludics

Untyped strategies (designs)

$\bot \bot$

Types (Behaviours)

Game Semantics

Typed strategies

Types (Arenas, Games)

- **Game Semantics**: All strategies are typed. Types GUARANTEE that strategies compose well.

- **Ludics**: Strategies are untyped (all given on a universal arena) Strategies can ALWAYS interact with each other, and interaction may terminate well ($\bot$) or not (deadlock, $\Omega$)
Nondeterminism: why

- An interactive account and of contraction — duplication rule:

\[
\frac{P(x, y) \vdash x : P, \ y : P}{P(z, z) \vdash z : P}
\]

where:
- \( P \) is a positive logical type;
- \( P(x, y) \) is a positive design with free variables in \( \{x, y\} \);
- \( P(z, z) \) is a positive design with free variable \( z \).

- Two different readings of the rule:
  - Top Down  *Contraction*: an identification of free variables.
  - Bottom Up  *Duplication*: an arbitrary bi-partition of occurrences of \( z \).
Nondeterminism: why

- An interactive account and of contraction — duplication rule:

\[
P(x, y) \vdash x : P, \ y : P
\]
\[
P(z, z) \vdash z : P
\]

where:
- \(P\) is a positive logical type;
- \(P(x, y)\) is a positive design with free variables in \(\{x, y\}\);
- \(P(z, z)\) is a positive design with free variable \(z\).

- Two different readings of the rule:
  - Top Down *Contraction*: an *identification* of free variables.
  - Bottom Up *Duplication*: an arbitrary *bi-partition* of occurrences of \(z\).
Nondeterminism: why

- An interactive account and of contraction — duplication rule:

\[
P(x, y) \vdash x : P, \ y : P
\]

\[
P(z, z) \vdash z : P
\]

where:

- \( P \) is a positive logical type;
- \( P(x, y) \) is a positive design with free variables in \( \{x, y\} \);
- \( P(z, z) \) is a positive design with free variable \( z \).

- Two different readings of the rule:
  
  **Top Down** Contraction: an identification of free variables.
  
  **Bottom Up** Duplication: an arbitrary bi-partition of occurrences of \( z \).
Failure of completeness

Write $P \models \Gamma$ for the interpretation of the sequent $P \vdash \Gamma$. Semantically, we have to show that:

$$\star \quad P(x, y) \models x : P, \ y : P \iff P(z, z) \models z : P$$

In general, $\star$ does not hold in a uniform setting.... We need to enlarge the universe of designs. We introduce (universal) nondeterminism.
Failure of completeness

Write $P \models \Gamma$ for the interpretation of the sequent $P \vdash \Gamma$. Semantically, we have to show that:

$$\star \quad P(x, y) \models x : P, \; y : P \iff P(z, z) \models z : P$$

In general, $\star$ does not hold in a uniform setting.... We need to *enlarge* the universe of designs. We introduce (universal) nondeterminism.
Failure of completeness

Write $P \models \Gamma$ for the interpretation of the sequent $P \vdash \Gamma$. Semantically, we have to show that:

$$\star \quad P(x, y) \models x : P, \ y : P \iff P(z, z) \models z : P$$

In general, $\star$ does not hold in a uniform setting.... We need to enlarge the universe of designs. We introduce (universal) nondeterminism.
Failure of completeness

Write $P \models \Gamma$ for the interpretation of the sequent $P \vdash \Gamma$. Semantically, we have to show that:

★ $P(x, y) \models x : P, \ y : P \iff P(z, z) \models z : P$

In general, ★ does not hold in a uniform setting.... We need to enlarge the universe of designs.

We introduce (universal) nondeterminism.
Write $P \models \Gamma$ for the interpretation of the sequent $P \vdash \Gamma$. Semantically, we have to show that:

\[ \star \quad P(x,y) \models x : P, \ y : P \iff P(z,z) \models z : P \]

In general, $\star$ does not hold in a *uniform* setting.... We need to *enlarge* the universe of designs. We introduce (universal) nondeterminism.
Designs

Coinductively defined terms given by the following grammar:

\[
P ::= \Omega \mid \bigwedge_i Q_i \quad \text{positive designs}
\]

\[
Q_i ::= N_0 \mid \overline{a} \langle N_1, \ldots, N_n \rangle \quad \text{predesigns}
\]

\[
N ::= x \mid \sum a(\vec{x}).P_a \quad \text{negative designs}
\]

- ▶ is now defined as the empty conjunction \( \bigwedge_\emptyset \). \( \bigwedge_{\{i\}} Q_i \) is simply written as \( Q_i \).

- ▶ A designs is deterministic if in any occurrence of subdesign \( \bigwedge_i Q_i \), \( i \) is either empty (and hence \( \bigwedge_i Q_i = \top \)) or a singleton.
Designs

Coinductively defined terms given by the following grammar:

\[ P ::= \emptyset \mid \bigwedge_i Q_i \quad \text{positive designs} \]
\[ Q_i ::= N_0 \mid \overline{a}\langle N_1, \ldots, N_n \rangle \quad \text{predesigns} \]
\[ N ::= x \mid \sum a(\vec{x}).P_a \quad \text{negative designs} \]

- $\emptyset$ is now defined as the empty conjunction $\bigwedge \emptyset$. $\bigwedge \{i\} Q_i$ is simply written as $Q_i$.
- A designs is deterministic if in any occurrence of subdesign $\bigwedge_i Q_i$, $I$ is either empty (and hence $\bigwedge_i Q_i = \emptyset$) or a singleton.
Designs

Coinductively defined terms given by the following grammar:

\[ P ::= \Omega \mid \bigwedge_i Q_i \quad \text{positive designs} \]
\[ Q_i ::= N_0 | \overline{a}\langle N_1, \ldots, N_n \rangle \quad \text{predesigns} \]
\[ N ::= x \mid \sum a(x).P_a \quad \text{negative designs} \]

- ♦ is now defined as the empty conjunction \( \bigwedge_\varnothing \). \( \bigwedge\{i\} Q_i \) is simply written as \( Q_i \).

- A designs is deterministic if in any occurrence of subdesign \( \bigwedge_i Q_i \), \( I \) is either empty (and hence \( \bigwedge_i Q_i = ♦ \)) or a singleton.
The **reduction relation** $\rightarrow$ is defined over the set of positive designs as follows:

$$\Omega \quad \rightarrow \quad \Omega;$$

$$Q \land \land \left( \sum a(\vec{x}).P_a \mid \overline{a}\langle\vec{N}\rangle \right) \quad \rightarrow \quad Q \land \land (P_a[\vec{N}/\vec{x}]).$$

Given two positive designs $Q, R$, we define:

**Convergence**: $Q \Downarrow R$, if $Q \rightarrow^* R$ and $R$ is a conjunction of head normal forms (no cuts);

**Divergence**: $Q \Uparrow$, otherwise. $Q \rightarrow^* \Omega, Q \rightarrow \ldots \rightarrow \ldots$
Normalization: Reduction

The **reduction relation** $\rightarrow$ is defined over the set of positive designs as follows:

$$
\Omega \rightarrow \Omega;
Q \land \land \left( \sum a(\vec{x}).P_a \mid \overline{a(\vec{N})} \right) \rightarrow Q \land \land \left( P_a[\vec{N}/\vec{x}] \right).
$$

Given two positive designs $Q, R$, we define:

**Convergence**: $Q \Downarrow R$, if $Q \rightarrow^* R$ and $R$ is a conjunction of head normal forms (no cuts);

**Divergence**: $Q \Uparrow$, otherwise. $Q \rightarrow^* \Omega$, $Q \rightarrow \ldots \rightarrow \ldots$
The normal form function $\llbracket \cdot \rrbracket : \mathcal{D} \rightarrow \mathcal{D}$ is defined by corecursion as follows:

\[
\begin{align*}
\llbracket x \rrbracket &= x; \\
\llbracket P \rrbracket &= \Omega, & \text{if } P \uparrow; \\
&= \bigwedge_i x_i|\overline{a}_i\langle \vec{N}_i \rangle & \text{if } P \downarrow \bigwedge_i x_i|\overline{a}_i\langle \vec{N}_i \rangle; \quad \sum a(\vec{x}).P_a \rrbracket = \sum a(\vec{x}).[P_a].
\end{align*}
\]

- $(a(\vec{x}).\boxtimes)|\overline{a}\langle \vec{N} \rangle = (a(\vec{x}).\bigwedge \emptyset)|\overline{a}\langle \vec{N} \rangle = \bigwedge \emptyset = \boxtimes$
- The dichotomy between $\boxtimes$ and $\Omega$ in the closed case is maintained: $\llbracket \bigwedge_i Q_i \rrbracket = \boxtimes$ iff any reduction sequence from any $Q_i$ is finite.
- $\bigwedge$ is universal: $\llbracket Q_1 \bigwedge Q_2 \rrbracket = \boxtimes$ iff $\llbracket Q_1 \rrbracket = \boxtimes$ and $\llbracket Q_2 \rrbracket = \boxtimes$. 

Normalization: Normal Form
The normal form function $\llbracket \ldots \rrbracket : \mathcal{D} \rightarrow \mathcal{D}$ is defined by corecursion as follows:

$$
\begin{align*}
\llbracket x \rrbracket &= x; \\
\llbracket P \rrbracket &= \Omega, & \text{if } P \uparrow; \\
&= \bigwedge_i x_i \bar{a}_i \langle \bar{N}_i \rangle, & \text{if } P \downarrow \bigwedge_i x_i \bar{a}_i \langle \bar{N}_i \rangle; \\
\llbracket \sum a(\bar{x}).P_a \rrbracket &= \sum a(\bar{x}).\llbracket P_a \rrbracket.
\end{align*}
$$

$\triangleright \quad (a(\bar{x}).\ast | \bar{a}\langle \bar{N} \rangle = (a(\bar{x}). \bigwedge \emptyset | \bar{a}\langle \bar{N} \rangle = \bigwedge \emptyset = \text{un}}$

$\triangleright \quad$ The dichotomy between $\ast$ and $\Omega$ in the closed case is maintained: $\llbracket \bigwedge_i Q_i \rrbracket = \ast$ iff any reduction sequence from any $Q_i$ is finite.

$\triangleright \quad$ $\bigwedge$ is universal: $\llbracket Q_1 \land Q_2 \rrbracket = \ast$ iff $\llbracket Q_1 \rrbracket = \ast$ and $\llbracket Q_2 \rrbracket = \ast$. 
The normal form function $\llbracket \cdot \rrbracket : \mathcal{D} \rightarrow \mathcal{D}$ is defined by corecursion as follows:

$$
\begin{align*}
\llbracket x \rrbracket &= x; \\
\llbracket P \rrbracket &= \Omega, & \text{if } P \uparrow; \\
&= \bigwedge_i x_i | a_i^{\llbracket \vec{N}_i \rrbracket} & \text{if } P \downarrow \bigwedge_i x_i | a_i^{\llbracket \vec{N}_i \rrbracket}; \\
\llbracket \sum a(\vec{x}).P_a \rrbracket &= \sum a(\vec{x}).\llbracket P_a \rrbracket.
\end{align*}
$$

- $(a(\vec{x}).\blacklozenge)|a^{\llbracket \vec{N} \rrbracket} = (a(\vec{x}).\bigwedge \emptyset)|a^{\llbracket \vec{N} \rrbracket} = \bigwedge \emptyset = \blacklozenge$
- The dichotomy between $\blacklozenge$ and $\Omega$ in the closed case is maintained: $\llbracket \bigwedge_i Q_i \rrbracket = \blacklozenge$ iff any reduction sequence from any $Q_i$ is finite.
- $\bigwedge$ is universal: $\llbracket Q_1 \bigwedge Q_2 \rrbracket = \blacklozenge$ iff $\llbracket Q_1 \rrbracket = \blacklozenge$ and $\llbracket Q_2 \rrbracket = \blacklozenge$. 

\[\blacklozenge\]
The normal form function $\llbracket \cdot \rrbracket : \mathcal{D} \rightarrow \mathcal{D}$ is defined by corecursion as follows:

$$
\llbracket x \rrbracket = x; \\
\llbracket P \rrbracket = \Omega, \quad \text{if } P \uparrow; \\
= \bigwedge_{i} x_{i} | a_{i} \langle \hat{N}_{i} \rangle \quad \text{if } P \downarrow \bigwedge_{i} x_{i} | a_{i} \langle \hat{N}_{i} \rangle; \\
\llbracket \sum a(\vec{x}).P_{a} \rrbracket = \sum a(\vec{x}).\llbracket P_{a} \rrbracket.
$$

- $(a(\vec{x}).\bigstar)|a\langle \hat{N} \rangle = (a(\vec{x}).\bigwedge \emptyset)|a\langle \hat{N} \rangle = \bigwedge \emptyset = \bigstar$

- The dichotomy between $\bigstar$ and $\Omega$ in the closed case is maintained: $\llbracket \bigwedge_{i} Q_{i} \rrbracket = \bigstar$ iff any reduction sequence from any $Q_{i}$ is finite.

- $\bigwedge$ is universal: $\llbracket Q_{1} \land Q_{2} \rrbracket = \bigstar$ iff $\llbracket Q_{1} \rrbracket = \bigstar$ and $\llbracket Q_{2} \rrbracket = \bigstar$. 
Example

\[
x | \overline{a} \langle y \rangle \land a(x).x | \overline{b} \langle y \rangle | \overline{a} \langle z \rangle \land b(x).(c(y).\overline{t} \mid \overline{c} \langle t \rangle) \mid \overline{b} \langle u \rangle \longrightarrow
\]

\[
x | \overline{a} \langle y \rangle \land z | \overline{b} \langle y \rangle \land c(y).\overline{t} \mid \overline{c} \langle t \rangle \longrightarrow x | \overline{a} \langle y \rangle \land z | \overline{b} \langle y \rangle.
\]
Some definitions

- $P$ is **total** if $P \neq \Omega$.
- $T$ is **linear** if for any subterm $N_0|a\langle N_1, \ldots, N_n\rangle$, $fv(N_0), \ldots, fv(N_n)$ are pairwise disjoint.
- $x$ is an **identity** if it occurs as $N_0|a\langle N_1, \ldots, x, \ldots, N_n\rangle$. 
Orthogonality

We consider only total, cut-free and identity free designs.

- $P$ is **closed** if $\text{fv}(P) = \emptyset$, **atomic** if $\text{fv}(P) \subseteq \{x_0\}$ for a certain fixed variable $x_0$.
- $N$ is **atomic** if $\text{fv}(N) = \emptyset$.
- $P, N$ are **orthogonal** $P \perp N$ when $P[N/x_0] = \bot$.
- For $X$ a set of atomic designs (same polarity):
  \[
  X^\perp := \{ E : \forall D \in X, D \perp E \}.
  \]
- A behaviour (interactive type) $G$ is a set of designs of the same polarity such that
  \[
  G^{\perp \perp} = G.
  \]
Orthogonality

We consider only total, cut-free and identity free designs.

- $P$ is **closed** if $\text{fv}(P) = \emptyset$, **atomic** if $\text{fv}(P) \subseteq \{x_0\}$ for a certain fixed variable $x_0$.
- $N$ is **atomic** if $\text{fv}(N) = \emptyset$.
- $P, N$ are **orthogonal** $P \perp N$ when $P[N/x_0] = \perp$.
- For $X$ a set of atomic designs (same polarity):
  \[ X^\perp := \{ E : \forall D \in X, \ D \perp E \} \]
- A **behaviour** (interactive type) $G$ is a set of designs of the same polarity such that
  \[ G^{\perp \perp} = G \]
Orthogonality

We consider only total, cut-free and identity free designs.

- **P** is **closed** if $fv(P) = \emptyset$, **atomic** if $fv(P) \subseteq \{x_0\}$ for a certain fixed variable $x_0$.
- **N** is **atomic** if $fv(N) = \emptyset$.
- **P, N** are **orthogonal** $P \perp N$ when $P[N/x_0] = \bot$.
- For **X** a set of atomic designs (same polarity):
  \[ X^\perp := \{ E : \forall D \in X, D \perp E \} \]
- **A behaviour** (interactive type) $G$ is a set of designs of the same polarity such that $G^{\perp \perp} = G$. 
We consider only total, cut-free and identity free designs.

- $P$ is **closed** if $\text{fv}(P) = \emptyset$, **atomic** if $\text{fv}(P) \subseteq \{x_0\}$ for a certain fixed variable $x_0$.
- $N$ is **atomic** if $\text{fv}(N) = \emptyset$.
- $P, N$ are **orthogonal** $P \perp N$ when $P[N/x_0] = \bot$.
- For $X$ a set of atomic designs (same polarity):
  \[ X \perp := \{ E : \forall D \in X, \ D \perp E \} \]

- A **behaviour** (interactive type) $G$ is a set of designs of the same polarity such that
  \[ G \perp \perp = G. \]
Logical Connectives

Fix a linear order on variables: $x_0, x_1, x_2, \ldots$.

- An $n$-ary logical connective $\alpha$ is a finite set of negative actions $\alpha = \{a_1(\vec{x}_1), \ldots, a_n(\vec{x}_n)\}$, where $\vec{x}_1, \ldots, \vec{x}_n$ are taken over $\{x_1, \ldots, x_n\}$.

- Given an $n$-ary logical connective $\alpha$ and behaviours $N_1, \ldots, N_n, P_1, \ldots, P_n$ we define:

$$\overline{a}\langle N_1, \ldots, N_m \rangle := \{x_0 | \overline{a}\langle N_1, \ldots, N_m \rangle : N_i \in N_i, 1 \leq i \leq m\}$$

$$\text{PC: } \overline{\alpha}\langle N_1, \ldots, N_n \rangle := (\bigcup_{a \in \alpha} \overline{a}\langle N_{i_1}, \ldots, N_{i_m} \rangle)^{\perp \perp}$$

where $i_1, \ldots, i_m \in \{1, \ldots, n\}$

$$\text{NC: } \alpha(\overline{P}_1, \ldots, \overline{P}_n) := \overline{\alpha}\langle \overline{P}_1^{\perp}, \ldots, \overline{P}_n^{\perp} \rangle^{\perp}$$

$$\overline{\alpha}\langle N_1, \ldots, N_n \rangle^{\perp} = \alpha\langle \overline{N}_1^{\perp}, \ldots, \overline{N}_n^{\perp} \rangle.$$
Logical Connectives

Fix a linear order on variables: \( x_0, x_1, x_2, \ldots \).

- An *n-ary logical connective* \( \alpha \) is a finite set of negative actions \( \alpha = \{ a_1(\vec{x}_1), \ldots, a_n(\vec{x}_n) \} \), where \( \vec{x}_1, \ldots, \vec{x}_n \) are taken over \( \{x_1, \ldots, x_n\} \).

- Given an \( n \)-ary logical connective \( \alpha \) and behaviours \( N_1, \ldots, N_n, P_1, \ldots, P_n \) we define:

\[
\overline{\alpha} \langle N_1, \ldots, N_m \rangle := \{ x_0 | \overline{\alpha} \langle N_1, \ldots, N_m \rangle : N_i \in N_i, 1 \leq i \leq m \}
\]

**PC:** \( \overline{\alpha} \langle N_1, \ldots, N_n \rangle := (\bigcup_{a \in \alpha} \overline{\alpha} \langle N_{i_1}, \ldots, N_{i_m} \rangle) \perp \perp \)
where \( i_1, \ldots, i_m \in \{1, \ldots, n\} \)

**NC:** \( \alpha (P_1, \ldots, P_n) := \overline{\alpha} \langle P_1 \perp, \ldots, P_n \perp \rangle \perp \)

\( (\overline{\alpha} \langle N_1, \ldots, N_n \rangle) \perp = \alpha \langle N_1 \perp, \ldots, N_n \perp \rangle \).
Logical Connectives

Fix a linear order on variables: $x_0, x_1, x_2$...

- **An $n$-ary logical connective** $\alpha$ is a finite set of negative actions $\alpha = \{a_1(\vec{x}_1), \ldots, a_n(\vec{x}_n)\}$, where $\vec{x}_1, \ldots, \vec{x}_n$ are taken over $\{x_1, \ldots, x_n\}$.

- Given an $n$-ary logical connective $\alpha$ and behaviours $N_1, \ldots, N_n, P_1, \ldots, P_n$ we define:

  $$\overline{\alpha}(N_1, \ldots, N_m) := \{x_0 | \overline{\alpha}(N_1, \ldots, N_m) : N_i \in N_i, 1 \leq i \leq m\}$$

  **PC:** $\overline{\alpha}(N_1, \ldots, N_n) := (\bigcup_{a \in \alpha} \overline{\alpha}(N_{i_1}, \ldots, N_{i_m})) \perp\perp$

  where $i_1, \ldots, i_m \in \{1, \ldots, n\}$

  **NC:** $\alpha(P_1, \ldots, P_n) := \overline{\alpha}(P_1 \perp, \ldots, P_n \perp) \perp$

- $(\overline{\alpha}(N_1, \ldots, N_n)) \perp = \alpha(N_1 \perp, \ldots, N_n \perp)$. 
Examples

Usual linear logic connectives can be defined by logical connectives $\emptyset$, $\&$, $\uparrow$, $\top$ below:

- $\emptyset := \{\emptyset\}$, $\bullet := \overline{\emptyset}$, $\otimes := \overline{\emptyset}$;
- $\& := \{\pi_1, \pi_2\}$, $\nu_i := \overline{\pi_i}$, $\oplus := \&$;
- $\uparrow := \{\uparrow\}$, $\downarrow := \overline{\uparrow}$.
- $\top := \emptyset$, $\mathbf{0} = \overline{\top}$.

$\emptyset, \bullet$ binary names, $\pi_i, \nu_i, \uparrow, \downarrow$ unary names.
Examples

Usual linear logic connectives can be defined by logical connectives \( \emptyset, \& \), \( \uparrow \), \( \top \) below:

- \( \emptyset := \{ \emptyset \}, \bullet := \overline{\emptyset}, \otimes := \overline{\emptyset}; \)
- \( \& := \{ \pi_1, \pi_2 \}, \iota_i := \overline{\pi_i}, \oplus := \&; \)
- \( \uparrow := \{ \uparrow \}, \downarrow := \overline{\uparrow}. \)
- \( \top := \emptyset, 0 = \overline{\top}. \)

\( \emptyset, \bullet \) binary names, \( \pi_i, \iota_i, \uparrow, \downarrow \) unary names.

\[
\begin{align*}
N \otimes M &= \bullet\langle N, M \rangle \perp \perp \\
N \oplus M &= (\iota_1 \langle N \rangle \cup \iota_2 \langle M \rangle) \perp \perp \\
\downarrow N &= \downarrow \langle N \rangle \perp \perp \\
1 &= \downarrow \langle T \rangle \perp \perp \\
P \& Q &= \bullet\langle P \perp, Q \perp \rangle \perp \\
P \& Q &= \iota_1 \langle P \perp \rangle \perp \cap \iota_2 \langle Q \perp \rangle \perp \\
\uparrow P &= \downarrow \langle P \perp \rangle \perp \\
\perp &= \downarrow \langle T \rangle \perp 
\end{align*}
\]
Logical behaviours and semantical sequents

**Logical behaviours:** *inductively* defined by

\[
P ::= \overline{\alpha}(N_1, \ldots, N_n) \quad N ::= \alpha(P_1, \ldots, P_n)
\]

- \( P \models x_1 : P_1, x_2 : P_2 \) if \( \text{fv}(P) \subseteq \{x_1, x_2\} \) and \( P[N_1/x_1, N_2/x_2] = \Box \) for any \( N_1 \in P_1, \ N_2 \in P_2 \).

- \( N \models x : P, N \) if \( \text{fv}(N) \subseteq \{x\} \) and \( P[N[M/x]/x_0] = \Box \) for any \( M \in P_\bot, \ P \in N_\bot \).

- \( P \models x_0 : P \) iff \( P \in P \).
Any positive logical behaviour satisfies:

Duplicability: \[ P[x_0/x_1, x_0/x_2] \models x_0 : P \iff P \models x_1 : P, x_2 : P \]

Any negative logical behaviour satisfies:

Closure under \( \land \): \[ N, M \in N \iff N \land M \in N \]

\[ N = \sum a(\vec{x}).P \quad M = \sum a(\vec{x}).Q \quad N \land M = \sum a(\vec{x}).P \land Q. \]
Any positive logical behaviour satisfies:

**Duplicability:** \( P[x_0/x_1, x_0/x_2] \models x_0 : P \iff P \models x_1 : P, x_2 : P \)

Any negative logical behaviour satisfies:

**Closure under \( \land \):** \( N, M \in N \iff N \land M \in N \)

\[ N = \sum a(\bar{x}).P \quad M = \sum a(\bar{x}).Q \quad N \land M = \sum a(\bar{x}).P \land Q. \]
About internal completeness (I)

- A purely monistic, local notion of completeness.
- A direct description of the elements in behaviours (built by logical connectives) without using the orthogonality and without referring to any proof system.

**Internal completeness** holds for negative logical connectives:

\[
\alpha(P_1, \ldots, P_n) = \{ \sum_\alpha a(\vec{x}).P_a : P_a \models x_{i_1} : P_{i_1}, \ldots x_{i_m} : P_{i_m} \}
\]

- \(P_b\) can be arbitrary when \(b(\vec{x}) \notin \alpha\).
- We have a lot of garbage...

\[
P_1 \& P_2 = \{ \pi_1(x_1).P_1 + \pi_2(x_2).P_2 + \cdots : P_i \models x_i : P_i \}
\]

irrelevant components of the sum are suppressed by \(\cdots\).

Up to *incarnation* (i.e. removal of irrelevant part), \(P_1 \& P_2\), which has been defined by *intersection*, is isomorphic to the cartesian product of \(P_1\) and \(P_2\): a phenomenon called *mystery of incarnation*. 
About internal completeness (I)

- A purely monistic, local notion of completeness.
- A direct description of the elements in behaviours (built by logical connectives) without using the orthogonality and without referring to any proof system.

**Internal completeness** holds for negative logical connectives:

\[
\alpha(P_1, \ldots, P_n) = \left\{ \sum_\alpha a(\vec{x}).P_a : P_a \models x_{i_1} : P_{i_1}, \ldots x_{i_m} : P_{i_m} \right\}
\]

- \(P_b\) can be arbitrary when \(b(\vec{x}) \notin \alpha\).
- We have a lot of garbage...

\[
P_1 \& P_2 = \left\{ \pi_1(x_1).P_1 + \pi_2(x_2).P_2 + \cdots : P_i \models x_i : P_i \right\}
\]

irrelevant components of the sum are suppressed by \(\cdots\)

Up to *incarnation* (i.e. removal of irrelevant part), \(P_1 \& P_2\), which has been defined by *intersection*, is isomorphic to the cartesian product of \(P_1\) and \(P_2\): a phenomenon called *mystery of incarnation*. 
About internal completeness (II)

For positive logical behaviours, it only holds (in that simple form) for *linear and deterministic designs*.

- Because any logical positive behaviour is *built* on linear and deterministic designs...
- But we want to take repetitions into account!
About internal completeness (II)

For positive logical behaviours, it only holds (in that simple form) for *linear* and *deterministic designs*.

- Because any logical positive behaviour is *built* on linear and deterministic designs...

- But we want to take repetitions into account!
For positive logical behaviours, it only holds (in that simple form) for *linear* and *deterministic designs*.

- Because any logical positive behaviour is *built* on linear and deterministic designs...
- But we want to take repetitions into account!
Proofs and Models

- A **proof** is a design in which all the conjunctions are unary. In other words, a proof is a deterministic and $\boxtimes$-free design.

- A **model** is an atomic linear design (in which conjunctions of arbitrary cardinality may occur).
Proof-system

\[
\begin{align*}
M_{i_1} & \vdash \Gamma, N_{i_1} \quad \ldots \quad M_{i_m} & \vdash \Gamma, N_{i_m} \quad (z : \overline{\alpha} \langle N_1, \ldots, N_n \rangle \in \Gamma) \\
& \quad \vdash \Gamma \quad (\overline{\alpha}, \overline{a}) \\
& z \mid \overline{a} \langle M_{i_1}, \ldots, M_{i_m} \rangle & \vdash \Gamma \\
\end{align*}
\]

\[
\begin{align*}
\left\{ P_a & \vdash \Gamma, \tilde{x}_a : \overline{P}_a \right\}_{a \in \alpha} \quad (\alpha) \\
\sum a(\tilde{x}).P_a & \vdash \Gamma, \alpha(P_1, \ldots, P_n) \\
\end{align*}
\]

\[
\begin{align*}
P & \vdash \Gamma, z : \overline{P} \quad N & \vdash \Gamma, \overline{P} \downarrow \\
P[N/z] & \vdash \Gamma \quad (cut)
\end{align*}
\]

where:

- In the rule \((\overline{\alpha}, \overline{a})\), \(a \in \alpha\), \(ar(a) = m\), and 
  \(i_1, \ldots, i_m \in \{1, \ldots, n\}\).
- In \((\alpha)\), \(\tilde{x}_a : \overline{P}_a\) stands for \(x_{i_1} : P_{i_1}, \ldots, x_{i_m} : P_{i_m}\).

Notice that:

- Structural rules (weakening and contraction/duplication) are implicit.
Proof-system

\[
\begin{align*}
M_{i_1} \vdash \Gamma, N_{i_1} & \quad \cdots \quad M_{i_m} \vdash \Gamma, N_{i_m} \quad (z : \bar{\alpha}\langle N_1, \ldots, N_n \rangle \in \Gamma) \\
\text{that } z | \bar{a}\langle M_{i_1}, \ldots, M_{i_m} \rangle \vdash \Gamma \\
\end{align*}
\]

\[
\begin{align*}
\{ P_a \vdash \Gamma, \bar{x}_a : \bar{P}_a \}_{a \in \alpha} \quad (\alpha) \\
\sum a(\bar{x}).P_a \vdash \Gamma, \alpha(P_1, \ldots, P_n) \\
P \vdash \Gamma, z : P \quad N \vdash \Gamma, P^\perp \quad (\text{cut}) \\
P[N/z] \vdash \Gamma
\end{align*}
\]

where:

- In the rule \((\bar{\alpha}, \bar{a})\), \(a \in \alpha\), \(ar(a) = m\), and \(i_1, \ldots, i_m \in \{1, \ldots, n\}\).
- In \((\alpha)\), \(\bar{x}_a : \bar{P}_a\) stands for \(x_{i_1} : P_{i_1}, \ldots, x_{i_m} : P_{i_m}\).

Notice that:

- Structural rules (weakening and contraction/duplication) are implicit.
Example

\[
\frac{M_1 \vdash \Gamma, N_1 \quad M_2 \vdash \Gamma, N_2 \quad (z : N_1 \otimes N_2 \in \Gamma)}{z \mid \bullet \langle M_1, M_2 \rangle \vdash \Gamma} \quad (\otimes, \bullet)
\]

\[
\frac{M \vdash \Gamma, N_i \quad (z : N_1 \oplus N_2 \in \Gamma)}{z \mid \iota_i \langle M \rangle \vdash \Gamma} \quad (\oplus, \iota_i)
\]

\[
\frac{P \vdash \Gamma, x_1 : P_1, x_2 : P_2 \quad \varnothing(x_1, x_2).P + \cdots \vdash \Gamma, P_1 \& P_2}{(\&)}
\]

\[
\frac{P_1 \vdash \Gamma, x_1 : P_1 \quad P_2 \vdash \Gamma, x_2 : P_2}{\pi_1(x_1).P_1 + \pi_2(x_2).P_2 + \cdots \vdash \Gamma, P_1 \& P_2} \quad (\&)
\]
Theorem (Soundness)

\( P \vdash P \iff P \models x : P. \)

The proof is given by induction on the depth of the type derivation \( P \vdash P \).

Theorem (Completeness (for proofs))

If \( P \) is a proof:

\( P \vdash x : P \iff P \vdash P. \)

Likewise for negative logical behaviours.
Theorem (Soundness)

\[ P \vdash P \iff P \models x : P. \]

The proof is given by induction on the depth of the type derivation \( P \vdash P \).

Theorem (Completeness (for proofs))

*If* \( P \) *is a proof:*

\[ P \models x : P \iff P \vdash P. \]

*Likewise for negative logical behaviours.*
Sketch of the proof

- Analogous to Schütte’s proof of Gödel’s completeness. We consider the statement:

\[ P \vdash P \iff P \not\models x : P. \]

1. Given an unprovable sequent \( \vdash P \), find an open branch in the cut-free proof search tree.
2. From the open branch, build a countermodel \( M \) in which \( P \) is false.

- The countermodel is here an atomic linear design in which conjunctions of arbitrary cardinality may occur. We can explicitly construct the countermodel.
- König Lemma is here essential.
- Closure under \( \land \) of \( P^\perp \) is essential to prove that the countermodel belongs to \( P^\perp \).
Sketch of the proof

Analogous to Schütte’s proof of Gödel’s completeness. We consider the statement:

\[ \vdash P \quad \implies \quad P \not\models x : P. \]

1. Given an unprovable sequent \( \vdash P \), find an open branch in the cut-free proof search tree.
2. From the open branch, build a countermodel \( M \) in which \( P \) is false.

The countermodel is here an atomic linear design in which conjunctions of arbitrary cardinality may occur. We can explicitly construct the countermodel.

König Lemma is here essential.

Closure under \( \land \) of \( P \perp \) is essential to prove that the countermodel belongs to \( P \perp \).
Corollaries

**Downward Löwenheim-Skolem** Let $P$ be a proof and $\mathbf{P}$ a logical behaviour. If $P \not\in \mathbf{P}$, then there is a *countable* model $M \in \mathbf{P}^\perp$ such that $P \not\not\in M$ ($M$ is countable in the sense that it consists of countably many actions $\neq \Omega$).

**Finite model property** If $P$ is linear, there is a finite (and deterministic) model $M \in \mathbf{P}^\perp$ such that $P \not\not\in M$. 
Conclusions

- Gödel’s completeness revisited in terms of ludics.
- We have enlighten the duality between proofs and models.
- We can give an explicit construction of a countermodel to any wrong proof attempt.
Related works

- Gödel’s incompleteness theorem.
- Recursive types (Melliès-Vouillon 05).
Thank you!

Questions?
Thank you!

Questions?