Ludics — Complete Glossary (Locus Solum)

Compact definitions keyed to the Mesh onboarding notes

Ludics — Complete Glossary / Definitions (Locus *Solum*)

> Concise, repo-friendly definitions keyed to *Locus Solum* (Girard). Terms are organized roughly by dependency; each entry is phrased for engineers and researchers.

Addresses, Loci

A **locus** is an explicit *place* of play (address) such as `ε`, `σ`, `σ·1`, `σ·1.2`. Branching encodes independence. *Why it matters:* makes locality/testability explicit.

Action (±), Polarity

Positive `(+ , ξ, I)` opens (enables) a finite set `I` of sub-addresses under `ξ`. **Negative** `(− , ξ.i)` focuses/responds at `ξ.i`. *Why it matters:* alternation drives the interaction.

Ramification / Directory

Ramification is the child-set `I` enabled by a positive action. The **directory** `Dir(B)` of a (negative) behaviour at its base is the set of immediately testable sub-addresses. Controls additivity.

Chronicle / View

A coherent alternating path of actions with proper justification pointers (what enabled what, where). Observational trace through a design.

Design

A (possibly infinite) **strategy** built of justified actions at addresses; locally finite and coherent. Think “interaction-ready proof object”.

Daimon (♦, ⊥)

Special terminal action; ends a run as success. Carries both technical and methodological roles (“Give up” in proof-nets).

Interaction, Normalization

Running a design against a counter-design along shared addresses. The **run normalizes** when it reaches a terminal configuration (often via `♦`).

Orthogonality (□)

`D □ E` iff the run `⟨D | E⟩` **converges** (normalizes). Defines compatibility.

Behaviour (Type) and Bi-orthogonality

A **behaviour** is a set `B` of designs closed under **bi-orthogonality**: `B = B⊥⊥`, where

`B⊥ = { E | ∀D∈B, D □ E }` and `B⊥⊥ = { D | ∀E∈B⊥, D □ E }`.

Separation (by tests)

Designs are determined by the counter-tests they pass: if `D⊥ ⊆ E⊥` then observationally `D ≼ E`. In Ludics the preorder is separating.

Incarnation (|B|, material part)

The **material** core of `B`: smallest designs in `B` that are actually visited by tests in `B⊥`. Used to state product-like facts for disjoint additives.

Additives (local to a base locus)

- **with** `&`: for same base/directory, **intersection** of behaviours: `B & C := B ∩ C`.
- **plus** `⊕`: polar dual `B ⊕ C := (B⊥ & C⊥)⊥`.

Disjoint negative case: `|B & C| ≅ |B| × |C|` (components independent).

Multiplicatives (⊗, ⅋)

Arise by composing support across **independent** sub-bases; tests factor component-wise. Duality via orthogonality.

Exponentials (!/? as protocols at addresses)

Structural rules realized **at loci**: copy by opening **fresh** sub-loci (`σ·0, σ·1, …`); discard via `♦`. Requires **freshness** (no aliasing) and **uniformity** across copies.

Delocation / Shift

Injective renaming of loci to restore disjointness/compatibility (e.g., tag `.L` vs `.R`). Used before forming additives when directories collide.

Cut (Composition)

Cut = **play composition**: run two designs along a shared interface for `A`/`A⊥`; normalization executes elimination.

Proof-nets & Switchings

Graphical correctness: **switchings** test acyclicity/connectedness for sequentialization. “Give up” corresponds to **daimon** in Ludics.

Consensus / Divergence

Game by **consensus**: divergence behaves like a **draw**. Enforce rule-following with **consensus-forcing testers** that make deviations lead to stuck runs elsewhere.

Internal Completeness

For the additive fragment at a fixed base: interaction already **saturates** the behaviour (bi-closure adds no new designs).

External Completeness (classical)

Truth ⇒ provability for the target calculus (stated for Π^1-like fragments in Girard’s overview).

Function Space (Arrow on behaviours)

`A ⊢ B := { D | ∀a∈A, ⟨D | a⟩ ∈ B }` — programs as **adapters** that drive any `A`-test to some `B`-design.

Quantifiers / Uniformity

Quantified behaviours require **parameter-independent** tests. Model via fresh-name discipline (and PER-like invariants) so testers cannot observe private codes.

Freshness

Guarantee that new sub-loci (`σ·i`) are distinct and do not alias existing addresses; essential for copy.

Test / Counter-design

An element of `B⊥`; used to define meaning by what interactions **succeed** against it.

Material Design

A design in `|B|`; minimal (no unvisited bureaucracy).

Saturation Tests

Test suites that probe **every** enabled branch (e.g., each `σ·i` after copy); membership in a behaviour requires convergence against all such tests.

Interface / Base

The locus (and its immediate directory) on which a behaviour is defined; determines what top-level tests are admissible.