Package 'largecardinals'

November 25, 2026

Type Package

Title Symbolic Large Cardinals and Scoped Finite Mathematical Models

Version 0.1.0

Description Provides native symbolic descriptions of large cardinals, typed formulas, explicit theory contexts, curated mathematical implications and bounded exact finite models. Optional graph exports retain relation kinds, hypotheses and evidence. Symbolic descriptions do not assert existence, and catalogue premises are distinguished from checked derivation steps.

License Apache License (== 2.0)

Encoding UTF-8

Depends R (>= 4.1.0)

Imports jsonlite, digest, stats, utils

Suggests testthat (>= 3.0.0), igraph, knitr, rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

URL https://github.com/rds0001/largecardinals-r

NeedsCompilation no

Author Dimitrios Geromichalos [aut, cre] (Dr.), RiskDataScience GmbH [cph]

Maintainer Dimitrios Geromichalos <riskdatascience@web.de>

Repository CRAN

Date/Publication 2026-11-25 13:00:02 UTC

Contents

add_assumption 3
analyze_finite_ideal 4
analyze_relation 5
boolean_truth 6
borel_hierarchy_spec 7

cardinal_iteration_spec . . . 8
cardinal_property . . . 9
catalogue_manifest . . . 10
check_applicability . . . 11
check_derivation . . . 11
check_finite_embedding . . . 12
check_finite_generic . . . 13
closure_requirements . . . 14
compatible_conditions . . . 15
conjunction . . . 16
constant . . . 16
critical_point_spec . . . 17
definition . . . 18
deserialize . . . 19
determinacy_query . . . 19
difference_hierarchy_spec . . . 20
difference_layers . . . 21
disjunction . . . 22
embedding_spec . . . 23
equal . . . 24
equivalence . . . 25
evaluate_formula . . . 26
exists_formula . . . 27
explain_derivation . . . 27
extender_spec . . . 28
filter_spec . . . 29
fingerprint . . . 30
finite_poset . . . 31
finite_skolem_hull . . . 32
finite_structure . . . 33
finite_tree . . . 34
finite_ultraproduct . . . 34
forall . . . 35
forcing_spec . . . 36
free_variables . . . 37
game_spec . . . 38
homogeneous_subset . . . 39
ideal_spec . . . 40
implication . . . 41
inner_model_spec . . . 42
iteration_spec . . . 43
laver_table . . . 44
list_definitions . . . 45
list_statements . . . 45
list_topics . . . 46
lookup_statement . . . 47
mahlo_iteration_spec . . . 47
mahlo_property . . . 48

measure_spec . . . 49
member . . . 50
negate . . . 51
normalize . . . 52
norm_spec . . . 52
ordinal_term . . . 53
parse_formula . . . 54
partition_relation_holds . . . 55
partition_spec . . . 56
pointclass_spec . . . 57
predicate . . . 58
preservation_query . . . 59
prewellorder_ranks . . . 60
projective_ordinal . . . 61
query_relation . . . 62
regularity_query . . . 63
relation_graph . . . 64
relativize . . . 65
scale_spec . . . 66
serialize_record . . . 67
small_subset_partition_spec . . . 67
solve_finite_game . . . 69
statement_result . . . 70
substitute_formula . . . 70
theory_context . . . 71
to_igraph . . . 72
transitive_collapse . . . 73
tree_branches . . . 74
tree_code . . . 74
ultrapower_spec . . . 75
validate_context . . . 76
validate_record . . . 77
validate_specification . . . 78
variable . . . 79
variants . . . 79
verify_finite_strategy . . . 80

Index **82**

`add_assumption` *Return a new context with an additional explicit assumption*

Description

Return a new context with an additional explicit assumption.

Usage

```
add_assumption(context, assumption)
```

Arguments

context	An explicit theory_context record, or NULL/None for the default ZFC context.
assumption	One additional nonempty assumption label.

Details

Expressions are sorted syntactic trees. Normalization changes bound names only, not logical content. Substitution avoids capture. A theory context is a declaration. Integers beyond 2^53-1 must use canonical decimal strings in ordinal terms. Serialization accepts bounded finite JSON data and never evaluates code. Typed deserialization validates against the current catalogue where applicable.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A new theory_context record with the additional assumption, retaining the original model and theory.

Examples

```
add_assumption(theory_context("ZF"), "AD")
```

---

analyze_finite_ideal *Check a complete finite ideal table and describe its quotient Boolean algebra*

Description

Check a complete finite ideal table and describe its quotient Boolean algebra.

A proper ideal contains the empty set, is downward closed and is closed under binary unions. Every finite ideal is principal; the atoms surviving its union describe the quotient. No infinite completeness or saturation is established.

Usage

```
analyze_finite_ideal(domain, members, max_steps = 100000L)
```

Arguments

domain	A finite collection of unique nonempty string labels, or a symbolic domain label in a norm/scale specification.
members	The complete finite family of ideal members, each a collection of domain labels. A generating family alone does not suffice.
max_steps	A positive operation budget. Finite algorithms accept at most 10000000; exhaustion raises RESOURCE_LIMIT.

Details

All conclusions are restricted to the supplied finite data. They do not witness infinite cardinals, genericity over a full infinite model or infinite determinacy. Invalid domains, missing interpretations and exhausted budgets raise classified errors. Tree depth is at most 100. Relation analysis uses at most 100 domain labels. Exhaustive partition queries are intentionally restricted.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A FINITE_EXACT query_result with an ideal-law violation or the principal support, quotient atoms, size and saturation information.

Examples

```
analyze_finite_ideal(list("a", "b", "c", "d"), list(list(), list("a"),
    list("b"), list("a", "b")))
```

---

analyze_relation *Compute finite order properties and a rank function for an acyclic relation*

Description

Compute finite order properties and a rank function for an acyclic relation.

Usage

```
analyze_relation(domain, pairs, max_steps = 1000000L)
```

Arguments

domain	A finite collection of unique nonempty string labels, or a symbolic domain label in a norm/scale specification.
pairs	The complete finite relation as pairs of domain labels.
max_steps	A positive operation budget. Finite algorithms accept at most 10000000; exhaustion raises RESOURCE_LIMIT.

Details

All conclusions are restricted to the supplied finite data. They do not witness infinite cardinals, genericity over a full infinite model or infinite determinacy. Invalid domains, missing interpretations and exhausted budgets raise classified errors. Tree depth is at most 100. Relation analysis uses at most 100 domain labels. Exhaustive partition queries are intentionally restricted.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A FINITE_EXACT query_result with reflexive, irreflexive, transitive, antisymmetric, total and well_founded flags, plus finite ordinal ranks when well founded.

Examples

```
analyze_relation(list("a", "b"), list(list("a", "b"), list("b", "a")))
```

---

boolean_truth *Evaluate a propositional formula in the finite Boolean algebra P(atoms)*

Description

Evaluate a propositional formula in the finite Boolean algebra P(atoms).

Usage

```
boolean_truth(expr, atoms, valuation)
```

Arguments

expr	A typed expression built by the public syntax constructors.
atoms	Finite labels for atoms of the Boolean algebra.
valuation	Named subsets of atoms interpreting zero-arity propositions.

Details

All conclusions are restricted to the supplied finite data. They do not witness infinite cardinals, genericity over a full infinite model or infinite determinacy. Invalid domains, missing interpretations and exhausted budgets raise classified errors. Tree depth is at most 100. Relation analysis uses at most 100 domain labels. Exhaustive partition queries are intentionally restricted.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A FINITE_EXACT query_result whose witness is the subset of Boolean atoms on which the propositional formula holds.

Examples

```
boolean_truth(implication(predicate("P"), predicate("Q")), c("a",
    "b"), list(P = "a", Q = c("a", "b")))
```

---

borel_hierarchy_spec *Describe a positive countable Borel rank, including transfinite ranks*

Description

Describe a positive countable Borel rank, including transfinite ranks.

Sigma at rank one is open; Pi is closed. Higher Sigma ranks use countable unions of Pi sets of smaller positive ranks, Pi is dual, and Delta is the intersection. Effective variants require a named ordinal notation relative to the listed real parameters. Countability, recursive presentation and membership remain unverified obligations.

Usage

```
borel_hierarchy_spec(index = 1, polarity = "Sigma", effective = FALSE,
    ordinal_code = NULL, parameters = list(), context = NULL)
```

Arguments

index	A positive countable ordinal rank, as an exact finite integer, ordinal term or symbolic label. Zero is rejected. Countability, positivity of unevaluated expressions and validity of an effective ordinal notation are mathematical obligations.
polarity	Sigma, Pi or Delta.
effective	Whether to use the effective convention; named real parameters are then the explicit oracle parameters for relativization.
ordinal_code	A symbolic ordinal-notation label, required exactly for effective Borel specifications. It must provide a recursive presentation relative to parameters, including effective limit-stage instructions; this is not checked by the constructor.
parameters	Named symbolic real parameters, or concept-specific keyword parameters for cardinal_property.
context	An explicit theory_context record, or NULL/None for the default ZFC context.

Details

Specifications describe infinite objects and retain explicit obligations. Boolean attributes are caller hypotheses; unspecified is not false. No existence, well-foundedness, elementarity or iterability proof is inferred. Mahlo iteration starts with the (weakly) inaccessible class at zero, uses stationary trace at successors, and ordinary intersection at limits. Indescribability levels require polarity and nonnegative finite order and block count (the field named alternations). Order zero uses first-order prenex complexity, not the bounded-quantifier Levy hierarchy.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A canonical specification record with family, concept, parameters, context, obligations, status SYMBOLIC and existence NOT_ASSERTED.

Examples

```
borel_hierarchy_spec()
```

---

cardinal_iteration_spec

Describe a decreasing transfinite hierarchy of regular cardinals

---

Description

Describe a decreasing transfinite hierarchy of regular cardinals.

Stage zero is initial_class. A successor retains members kappa of the preceding stage for which that stage below kappa is unbounded (limit_points) or stationary (stationary_trace). Limit stages use ordinary intersection. A diagonal intersection is a different operation.

Usage

```
cardinal_iteration_spec(operator, index, initial_class = "inaccessible",
    context = NULL)
```

Arguments

operator	limit_points or stationary_trace; selects unboundedness or stationarity of the preceding stage below a retained cardinal.
index	An exact ordinal term, decimal integer or symbolic ordinal name.
initial_class	regular, weakly_inaccessible or inaccessible; this class is stage zero. Successors retain members of the preceding stage and limit stages take ordinary intersections.
context	An explicit theory_context record, or NULL/None for the default ZFC context.

Details

Specifications describe infinite objects and retain explicit obligations. Boolean attributes are caller hypotheses; unspecified is not false. No existence, well-foundedness, elementarity or iterability proof is inferred. Mahlo iteration starts with the (weakly) inaccessible class at zero, uses stationary trace at successors, and ordinary intersection at limits. Indescribability levels require polarity and nonnegative finite order and block count (the field named alternations). Order zero uses first-order prenex complexity, not the bounded-quantifier Levy hierarchy.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A canonical specification record with family, concept, parameters, context, obligations, status SYMBOLIC and existence NOT_ASSERTED.

Examples

```
cardinal_iteration_spec("limit_points", ordinal_term("omega"), initial_class = "regular")
```

---

cardinal_property *Instantiate a cardinal property with exact symbolic parameters*

Description

Instantiate a cardinal property with exact symbolic parameters.

Requirements in the definition are not inferred as facts. The constructor can describe a hypothesis even in a context in which it is inconsistent.

Usage

```
cardinal_property(concept, kappa = "kappa", context = NULL, ...)
```

Arguments

concept	A definition ID; cardinal constructors require kind cardinal_property.
kappa	An ASCII symbolic name for the cardinal under discussion.
context	An explicit theory_context record, or NULL/None for the default ZFC context.
...	Formula operands, ordinal operands, or named concept parameters, as required by the selected constructor.

Details

Specifications describe infinite objects and retain explicit obligations. Boolean attributes are caller hypotheses; unspecified is not false. No existence, well-foundedness, elementarity or iterability proof is inferred. Mahlo iteration starts with the (weakly) inaccessible class at zero, uses stationary trace at successors, and ordinary intersection at limits. Indescribability levels require polarity and nonnegative finite order and block count (the field named alternations). Order zero uses first-order prenex complexity, not the bounded-quantifier Levy hierarchy.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A canonical specification record with family, concept, parameters, context, obligations, status SYMBOLIC and existence NOT_ASSERTED.

Examples

```
cardinal_property("weakly_hyperwoodin")
```

---

catalogue_manifest *Return versions, counts, content fingerprint and trust policy*

Description

Return versions, counts, content fingerprint and trust policy.

Usage

```
catalogue_manifest()
```

Details

Catalogue premises are internally reviewed mathematical statements, not formal-kernel proofs. Only applicable same-cardinal implications are transitively composed. Other relation kinds remain direct. A missing relation returns UNKNOWN. Named theory-extension records are assumption schemas; their existence claims require explicit caller assumptions. Rechecking a derivation validates its trusted edges and composition, not the underlying theorems.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A catalogue_manifest record containing version, definition/statement/relation counts, SHA-256 and trust policy.

Examples

```
catalogue_manifest()
```

check_applicability — *Check a statement's declared prerequisites, without proving those prerequisites*

Description

Check a statement's declared prerequisites, without proving those prerequisites.

Usage

```
check_applicability(identifier, context = NULL)
```

Arguments

identifier	A public catalogue identifier; unknown IDs raise an input error.
context	An explicit theory_context record, or NULL/None for the default ZFC context.

Details

Catalogue premises are internally reviewed mathematical statements, not formal-kernel proofs. Only applicable same-cardinal implications are transitively composed. Other relation kinds remain direct. A missing relation returns UNKNOWN. Named theory-extension records are assumption schemas; their existence claims require explicit caller assumptions. Rechecking a derivation validates its trusted edges and composition, not the underlying theorems.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A SYMBOLIC query_result with missing_assumptions and model_matches in its witness, or an inconsistent-context result.

Examples

```
check_applicability("pressing_down", theory_context())
```

check_derivation — *Recheck edge identities, direction, contexts and allowed composition rules*

Description

Recheck edge identities, direction, contexts and allowed composition rules.

Usage

```
check_derivation(proof)
```

Arguments

proof	A derivation record carrying the exact kernel, catalogue version, context and edge identities.

Details

Catalogue premises are internally reviewed mathematical statements, not formal-kernel proofs. Only applicable same-cardinal implications are transitively composed. Other relation kinds remain direct. A missing relation returns UNKNOWN. Named theory-extension records are assumption schemas; their existence claims require explicit caller assumptions. Rechecking a derivation validates its trusted edges and composition, not the underlying theorems.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A CHECKED_DERIVATION query_result after checking the allowed steps against the installed catalogue; invalid steps raise a classified error.

Examples

```
check_derivation(query_relation("measurable", "mahlo")$witness)
```

---

check_finite_embedding

Check full first-order elementarity between two finite relational structures

---

Description

Check full first-order elementarity between two finite relational structures.

In the full first-order language with equality, an elementary map between finite structures is necessarily an isomorphism onto the target. This test checks that characterization, including constants and relation reflection.

Usage

```
check_finite_embedding(source, target, mapping, max_steps = 100000L)
```

Arguments

source	The source finite_structure record.
target	The target finite_structure record. The supplied map must be onto this domain to be elementary between finite structures.
mapping	A total named map from the finite source domain into the finite target domain.
max_steps	A positive operation budget. Finite algorithms accept at most 10000000; exhaustion raises RESOURCE_LIMIT.

Details

All conclusions are restricted to the supplied finite data. They do not witness infinite cardinals, genericity over a full infinite model or infinite determinacy. Invalid domains, missing interpretations and exhausted budgets raise classified errors. Tree depth is at most 100. Relation analysis uses at most 100 domain labels. Exhaustive partition queries are intentionally restricted.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A FINITE_EXACT result checking an elementary map between the two finite structures. For finite relational structures this requires an isomorphism onto the entire target.

Examples

```
check_finite_embedding(finite_structure(list("a", "b"), list(P = list(arity = 1,
    tuples = list(list("a"))))), finite_structure(list("u", "v"),
    list(P = list(arity = 1, tuples = list(list("v"))))), list(a = "v",
    b = "u"))
```

---

check_finite_generic *Check a filter and meeting a named finite family of genuinely dense sets*

Description

Check a filter and meeting a named finite family of genuinely dense sets.

Usage

```
check_finite_generic(poset, filter_members, dense_sets)
```

Arguments

poset	A finite_poset record, or a symbolic forcing-poset label in forcing_spec.
filter_members	The proposed finite filter as condition labels.
dense_sets	A finite list of genuinely dense subsets of the supplied poset. This list does not stand for all dense sets of an infinite model.

Details

All conclusions are restricted to the supplied finite data. They do not witness infinite cardinals, genericity over a full infinite model or infinite determinacy. Invalid domains, missing interpretations and exhausted budgets raise classified errors. Tree depth is at most 100. Relation analysis uses at most 100 domain labels. Exhaustive partition queries are intentionally restricted.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A FINITE_EXACT result checking the filter laws and meeting each supplied dense set. It does not quantify over an infinite ground model.

Examples

```
check_finite_generic(finite_poset(list("p", "q", "top"), list(list("p",
    "p"), list("q", "q"), list("top", "top"), list("p", "top"), list("q",
    "top"))), list("p", "top"), list(list("p", "q")))
```

---

closure_requirements *Extract the closure declaration; an absent requirement remains unspecified*

Description

Extract the closure declaration; an absent requirement remains unspecified.

Usage

```
closure_requirements(embedding)
```

Arguments

embedding	A validated embedding_spec record.

Details

Specifications describe infinite objects and retain explicit obligations. Boolean attributes are caller hypotheses; unspecified is not false. No existence, well-foundedness, elementarity or iterability proof is inferred. Mahlo iteration starts with the (weakly) inaccessible class at zero, uses stationary trace at successors, and ordinary intersection at limits. Indescribability levels require polarity and nonnegative finite order and block count (the field named alternations). Order zero uses first-order prenex complexity, not the bounded-quantifier Levy hierarchy.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A declaration with the symbolic closure condition, status DECLARED and verified false.

Examples

```
closure_requirements(embedding_spec(closure = "lambda_sequences"))
```

---

compatible_conditions *Find common stronger conditions in an explicit finite partial order*

Description

Find common stronger conditions in an explicit finite partial order.

Usage

```
compatible_conditions(poset, left, right)
```

Arguments

poset	A finite_poset record, or a symbolic forcing-poset label in forcing_spec.
left	The left operand, or a condition label for compatibility queries.
right	The right operand, or a condition label for compatibility queries.

Details

All conclusions are restricted to the supplied finite data. They do not witness infinite cardinals, genericity over a full infinite model or infinite determinacy. Invalid domains, missing interpretations and exhausted budgets raise classified errors. Tree depth is at most 100. Relation analysis uses at most 100 domain labels. Exhaustive partition queries are intentionally restricted.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A FINITE_EXACT query_result with common stronger conditions witnessing compatibility, or REFUTED.

Examples

```
compatible_conditions(finite_poset(list("p", "q", "top"), list(list("p",
    "p"), list("q", "q"), list("top", "top"), list("p", "top"), list("q",
    "top"))), "p", "q")
```

conjunction *Form conjunction; the empty conjunction is true*

Description

Form conjunction; the empty conjunction is true.

Usage

```
conjunction(...)
```

Arguments

...	Formula operands, ordinal operands, or named concept parameters, as required by the selected constructor.

Details

Expressions are sorted syntactic trees. Normalization changes bound names only, not logical content. Substitution avoids capture. A theory context is a declaration. Integers beyond 2^53-1 must use canonical decimal strings in ordinal terms. Serialization accepts bounded finite JSON data and never evaluates code. Typed deserialization validates against the current catalogue where applicable.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A formula expression with the supplied conjuncts; an empty conjunction is true.

Examples

```
conjunction(predicate("P"), predicate("Q"))
```

constant *Declare a named term, without asserting that it has an interpretation*

Description

Declare a named term, without asserting that it has an interpretation.

Usage

```
constant(name, sort = "set")
```

Arguments

name	A mathematical identifier. Formula identifiers use ASCII letters, digits, underscores and dots, starting with a letter or underscore.
sort	One of set, ordinal, cardinal, class, real or natural. The finite evaluator accepts only set-sorted terms.

Details

Expressions are sorted syntactic trees. Normalization changes bound names only, not logical content. Substitution avoids capture. A theory context is a declaration. Integers beyond 2^53-1 must use canonical decimal strings in ordinal terms. Serialization accepts bounded finite JSON data and never evaluates code. Typed deserialization validates against the current catalogue where applicable.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A typed constant expression with the supplied name and sort.

Examples

```
constant("a")
```

---

critical_point_spec *Extract a declared critical point with its source and target obligations*

Description

Extract a declared critical point with its source and target obligations.

Usage

```
critical_point_spec(embedding)
```

Arguments

embedding	A validated embedding_spec record.

Details

Specifications describe infinite objects and retain explicit obligations. Boolean attributes are caller hypotheses; unspecified is not false. No existence, well-foundedness, elementarity or iterability proof is inferred. Mahlo iteration starts with the (weakly) inaccessible class at zero, uses stationary trace at successors, and ordinary intersection at limits. Indescribability levels require polarity and nonnegative finite order and block count (the field named alternations). Order zero uses first-order prenex complexity, not the bounded-quantifier Levy hierarchy.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A declaration with symbol, source, target, status DECLARED and existence NOT_ASSERTED.

Examples

```
critical_point_spec(embedding_spec())
```

`definition` *Return an independent copy of a definition with its parameters and assumptions*

Description

Return an independent copy of a definition with its parameters and assumptions.

Usage

```
definition(identifier)
```

Arguments

`identifier`	A public catalogue identifier; unknown IDs raise an input error.

Details

Catalogue premises are internally reviewed mathematical statements, not formal-kernel proofs. Only applicable same-cardinal implications are transitively composed. Other relation kinds remain direct. A missing relation returns UNKNOWN. Named theory-extension records are assumption schemas; their existence claims require explicit caller assumptions. Rechecking a derivation validates its trusted edges and composition, not the underlying theorems.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

An independent definition record including text, kind, parameters, requirements and notes.

Examples

```
definition("mahlo")
```

`deserialize` *Load a typed record; duplicate keys and non-finite numeric values are rejected*

Description

Load a typed record; duplicate keys and non-finite numeric values are rejected.

Usage

```
deserialize(text)
```

Arguments

`text`	A bounded JSON string with unique object keys and a supported typed root record.

Details

Expressions are sorted syntactic trees. Normalization changes bound names only, not logical content. Substitution avoids capture. A theory context is a declaration. Integers beyond 2^53-1 must use canonical decimal strings in ordinal terms. Serialization accepts bounded finite JSON data and never evaluates code. Typed deserialization validates against the current catalogue where applicable.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A reconstructed validated typed record or expression; malformed input raises a classified error.

Examples

```
deserialize(largecardinals::serialize_record(theory_context()))
```

`determinacy_query` *Query determinacy with explicit payoff class and move spaces*

Description

Query determinacy with explicit payoff class and move spaces.

moves accepts naturals, reals, reals_vs_naturals (player I plays reals) or naturals_vs_reals (player II plays reals). All games have length omega. Ordinary AD and PD do not settle real-move games here. For such games, AD_R or the appropriate AD_mixed hypothesis is required; no general Borel determinacy claim for a different move topology is inferred.

Usage

```
determinacy_query(pointclass, context = NULL, moves = "naturals")
```

Arguments

pointclass	A symbolic pointclass label. Queries accept borel, analytic, coanalytic, projective or arbitrary.
context	An explicit theory_context record, or NULL/None for the default ZFC context.
moves	naturals, reals, reals_vs_naturals or naturals_vs_reals. In a mixed game the first component is player I and the second player II; length is omega.

Details

Catalogue premises are internally reviewed mathematical statements, not formal-kernel proofs. Only applicable same-cardinal implications are transitively composed. Other relation kinds remain direct. A missing relation returns UNKNOWN. Named theory-extension records are assumption schemas; their existence claims require explicit caller assumptions. Rechecking a derivation validates its trusted edges and composition, not the underlying theorems.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A query_result for the requested payoff and move spaces. ESTABLISHED uses a declared determinacy axiom or applicable literature premise; it supplies no infinite strategy.

Examples

```
determinacy_query("borel", context = theory_context("ZF_DC", list("AD")))
```

---

difference_hierarchy_spec

Specify alternating differences of a decreasing sequence of sets

---

Description

Specify alternating differences of a decreasing sequence of sets.

A point is selected when its first exit index is odd; a point that never exits uses the sequence length as its index. Thus lengths one, two and three give A0, A0 minus A1, and (A0 minus A1) union A2. Transfinite parity uses the finite tail after the limit part. The length is required mathematically to be countable. For effective differences the named code must jointly define the sequence and a recursive well-order of its length relative to the named parameters; these obligations are declared, not verified from a symbolic label.

Usage

```
difference_hierarchy_spec(base_pointclass = "coanalytic", index = 1,
    effective = FALSE, sequence_code = NULL, context = NULL)
```

Arguments

base_pointclass	The symbolic class supplying the decreasing sequence of sets.
index	An exact symbolic countable ordinal length; zero denotes the empty difference. Countability and an effective presentation, where requested, remain mathematical obligations.
effective	Whether to use the effective convention; named real parameters are then the explicit oracle parameters for relativization.
sequence_code	A symbolic joint code for the sequence, required in the effective convention. Recursive presentation and joint definability are mathematical obligations, not verified from the label.
context	An explicit theory_context record, or NULL/None for the default ZFC context.

Details

Specifications describe infinite objects and retain explicit obligations. Boolean attributes are caller hypotheses; unspecified is not false. No existence, well-foundedness, elementarity or iterability proof is inferred. Mahlo iteration starts with the (weakly) inaccessible class at zero, uses stationary trace at successors, and ordinary intersection at limits. Indescribability levels require polarity and nonnegative finite order and block count (the field named alternations). Order zero uses first-order prenex complexity, not the bounded-quantifier Levy hierarchy.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A canonical specification record with family, concept, parameters, context, obligations, status SYMBOLIC and existence NOT_ASSERTED.

Examples

```
difference_hierarchy_spec(index = ordinal_term("omega"))
```

---

difference_layers *Evaluate alternating differences of a finite decreasing sequence*

Description

Evaluate alternating differences of a finite decreasing sequence.

Uses zero-based first-exit indices, with exhaustion assigned len(layers). A point belongs exactly when that index is odd. Input sets must be nested subsets of the finite domain, with no duplicate labels. Work is O(len(domain) * (len(layers) + 1)); no transfinite hierarchy membership is decided. At most 1000 domain labels are accepted.

Usage

```
difference_layers(domain, layers, max_steps = 100000L)
```

Arguments

`domain`	A finite collection of unique nonempty string labels, or a symbolic domain label in a norm/scale specification.
`layers`	An ordered finite list of decreasing subsets of domain; each subset has unique domain labels. Empty layers are allowed.
`max_steps`	A positive operation budget. Finite algorithms accept at most 10000000; exhaustion raises RESOURCE_LIMIT.

Details

All conclusions are restricted to the supplied finite data. They do not witness infinite cardinals, genericity over a full infinite model or infinite determinacy. Invalid domains, missing interpretations and exhausted budgets raise classified errors. Tree depth is at most 100. Relation analysis uses at most 100 domain labels. Exhaustive partition queries are intentionally restricted.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A FINITE_EXACT query_result with selected members, zero-based first_exit indices and the finite length.

Examples

```
difference_layers(list("a", "b", "c", "outside"), list())
```

---

disjunction *Form disjunction; the empty disjunction is false*

Description

Form disjunction; the empty disjunction is false.

Usage

```
disjunction(...)
```

Arguments

...	Formula operands, ordinal operands, or named concept parameters, as required by the selected constructor.

Details

Expressions are sorted syntactic trees. Normalization changes bound names only, not logical content. Substitution avoids capture. A theory context is a declaration. Integers beyond 2^53-1 must use canonical decimal strings in ordinal terms. Serialization accepts bounded finite JSON data and never evaluates code. Typed deserialization validates against the current catalogue where applicable.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A formula expression with the supplied disjuncts; an empty disjunction is false.

Examples

```
disjunction(predicate("P"), predicate("Q"))
```

---

embedding_spec *Declare source, target, critical point, language fragment and closure*

Description

Declare source, target, critical point, language fragment and closure.

Usage

```
embedding_spec(source = "V", target = "M", critical_point = "kappa",
    elementarity = "full_first_order", closure = NULL, context = NULL)
```

Arguments

source	Source text for the parser, or a symbolic source/model label for a specification.
target	A finite homogeneous-set size or symbolic target label, depending on the function.
critical_point	A symbolic declared critical point; its existence is not established.
elementarity	The declared formula fragment preserved by an embedding.
closure	A symbolic closure hypothesis, or unspecified.
context	An explicit theory_context record, or NULL/None for the default ZFC context.

Details

Specifications describe infinite objects and retain explicit obligations. Boolean attributes are caller hypotheses; unspecified is not false. No existence, well-foundedness, elementarity or iterability proof is inferred. Mahlo iteration starts with the (weakly) inaccessible class at zero, uses stationary trace at successors, and ordinary intersection at limits. Indescribability levels require polarity and nonnegative finite order and block count (the field named alternations). Order zero uses first-order prenex complexity, not the bounded-quantifier Levy hierarchy.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A canonical specification record with family, concept, parameters, context, obligations, status SYMBOLIC and existence NOT_ASSERTED.

Examples

```
embedding_spec()
```

`equal` *Form equality of two terms*

Description

Form equality of two terms.

Usage

```
equal(left, right)
```

Arguments

`left`	The left operand, or a condition label for compatibility queries.
`right`	The right operand, or a condition label for compatibility queries.

Details

Expressions are sorted syntactic trees. Normalization changes bound names only, not logical content. Substitution avoids capture. A theory context is a declaration. Integers beyond 2^53-1 must use canonical decimal strings in ordinal terms. Serialization accepts bounded finite JSON data and never evaluates code. Typed deserialization validates against the current catalogue where applicable.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

An equality formula on two terms, retaining their declared sorts.

Examples

```
equal(variable("x"), variable("y"))
```

`equivalence` *Form logical equivalence, not equiconsistency of theories*

Description

Form logical equivalence, not equiconsistency of theories.

Usage

```
equivalence(left, right)
```

Arguments

`left`	The left operand, or a condition label for compatibility queries.
`right`	The right operand, or a condition label for compatibility queries.

Details

Expressions are sorted syntactic trees. Normalization changes bound names only, not logical content. Substitution avoids capture. A theory context is a declaration. Integers beyond 2^53-1 must use canonical decimal strings in ordinal terms. Serialization accepts bounded finite JSON data and never evaluates code. Typed deserialization validates against the current catalogue where applicable.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A formula expression for logical equivalence.

Examples

```
equivalence(predicate("P"), predicate("Q"))
```

`evaluate_formula` *Evaluate a first-order formula on the specified finite structure only*

Description

Evaluate a first-order formula on the specified finite structure only.

Usage

```
evaluate_formula(expr, structure, assignment = list(), max_steps = 100000L)
```

Arguments

`expr`	A typed expression built by the public syntax constructors.
`structure`	A finite_structure record. No claim that it satisfies ZF is made.
`assignment`	Named assignments of all free variables to members of the finite domain.
`max_steps`	A positive operation budget. Finite algorithms accept at most 10000000; exhaustion raises RESOURCE_LIMIT.

Details

All conclusions are restricted to the supplied finite data. They do not witness infinite cardinals, genericity over a full infinite model or infinite determinacy. Invalid domains, missing interpretations and exhausted budgets raise classified errors. Tree depth is at most 100. Relation analysis uses at most 100 domain labels. Exhaustive partition queries are intentionally restricted.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A FINITE_EXACT query_result with ESTABLISHED or REFUTED and witness fields truth, assignment and domain.

Examples

```
evaluate_formula(forall(variable("x"), predicate("P", variable("x"))),
   finite_structure(list("a", "b"), list(P = list(arity = 1, tuples = list(list("a"))))))
```

exists_formula *Existentially bind a variable; syntax construction supplies no witness*

Description

Existentially bind a variable; syntax construction supplies no witness.

Usage

```
exists_formula(bound, body)
```

Arguments

bound	The variable expression bound by the quantifier.
body	A formula expression.

Details

Expressions are sorted syntactic trees. Normalization changes bound names only, not logical content. Substitution avoids capture. A theory context is a declaration. Integers beyond 2^53-1 must use canonical decimal strings in ordinal terms. Serialization accepts bounded finite JSON data and never evaluates code. Typed deserialization validates against the current catalogue where applicable.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

An existentially quantified formula expression with an explicit bound variable.

Examples

```
largecardinals::exists_formula(variable("x"), predicate("P", variable("x")))
```

explain_derivation *Return typed steps and full premise metadata for a relation query*

Description

Return typed steps and full premise metadata for a relation query.

Usage

```
explain_derivation(result)
```

Arguments

result	A relation query result whose dependency metadata will be inspected.

Details

Catalogue premises are internally reviewed mathematical statements, not formal-kernel proofs. Only applicable same-cardinal implications are transitively composed. Other relation kinds remain direct. A missing relation returns UNKNOWN. Named theory-extension records are assumption schemas; their existence claims require explicit caller assumptions. Rechecking a derivation validates its trusted edges and composition, not the underlying theorems.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A collection of the relation query dependencies with their full catalogue metadata.

Examples

```
explain_derivation(query_relation("measurable", "mahlo"))
```

---

extender_spec *Describe extender support length and independent coherence/well-foundedness obligations*

Description

Describe extender support length and independent coherence/well-foundedness obligations.

Usage

```
extender_spec(critical_point = "kappa", length = "lambda", source_model = "V",
    well_founded = NULL, coherent = NULL, context = NULL)
```

Arguments

critical_point	A symbolic declared critical point; its existence is not established.
length	An exact ordinal length. Games default to omega.
source_model	A symbolic source-model label.
well_founded	A Boolean well-foundedness hypothesis; unspecified does not mean false.
coherent	A Boolean coherence hypothesis, or unspecified.
context	An explicit theory_context record, or NULL/None for the default ZFC context.

Details

Specifications describe infinite objects and retain explicit obligations. Boolean attributes are caller hypotheses; unspecified is not false. No existence, well-foundedness, elementarity or iterability proof is inferred. Mahlo iteration starts with the (weakly) inaccessible class at zero, uses stationary trace at successors, and ordinary intersection at limits. Indescribability levels require polarity and nonnegative finite order and block count (the field named alternations). Order zero uses first-order prenex complexity, not the bounded-quantifier Levy hierarchy.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A canonical specification record with family, concept, parameters, context, obligations, status SYMBOLIC and existence NOT_ASSERTED.

Examples

```
extender_spec()
```

`filter_spec` *Describe a filter, distinguishing internal hypotheses from external completeness*

Description

Describe a filter, distinguishing internal hypotheses from external completeness.

Completeness, normality and ultrafilter status are relative to relative_model, which defaults to the context model. Weak amenability means that membership in the filter can be collected for carrier-length sequences in that model. External countable completeness requires a nonempty intersection for every ambient countable family of filter members, even families outside the model. Each attribute is a declaration, not a check of an infinite filter.

Usage

```
filter_spec(carrier = "kappa", completeness = NULL, normal = NULL,
    fine = NULL, nonprincipal = NULL, ultrafilter = NULL, weakly_amenable = NULL,
    externally_countably_complete = NULL, relative_model = NULL, context = NULL)
```

Arguments

`carrier`	A symbolic carrier label.
`completeness`	An exact ordinal completeness parameter, or unspecified.
`normal`	A Boolean normality hypothesis, or unspecified.
`fine`	A Boolean fineness hypothesis, or unspecified.
`nonprincipal`	A Boolean nonprincipality hypothesis, or unspecified.

`ultrafilter` In filter_spec, a Boolean declaration or NULL/None. In ultrapower_spec, a symbolic ultrafilter label.

`weakly_amenable`
Declared weak amenability for sequences in relative_model; NULL/None leaves it unspecified.

`externally_countably_complete`
Declared nonempty intersection for every ambient countable family of filter members, including families outside relative_model. This is distinct from internal completeness.

`relative_model` Model in which the filter or ideal hypotheses are interpreted; defaults to the context model.

`context` An explicit theory_context record, or NULL/None for the default ZFC context.

Details

Specifications describe infinite objects and retain explicit obligations. Boolean attributes are caller hypotheses; unspecified is not false. No existence, well-foundedness, elementarity or iterability proof is inferred. Mahlo iteration starts with the (weakly) inaccessible class at zero, uses stationary trace at successors, and ordinary intersection at limits. Indescribability levels require polarity and nonnegative finite order and block count (the field named alternations). Order zero uses first-order prenex complexity, not the bounded-quantifier Levy hierarchy.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A canonical specification record with family, concept, parameters, context, obligations, status SYMBOLIC and existence NOT_ASSERTED.

Examples

```
filter_spec()
```

`fingerprint` *SHA-256 of canonical representation; not a semantic equivalence hash*

Description

SHA-256 of canonical representation; not a semantic equivalence hash.

Usage

```
fingerprint(value)
```

Arguments

value	The value or typed record to validate, serialize or fingerprint.

Details

Expressions are sorted syntactic trees. Normalization changes bound names only, not logical content. Substitution avoids capture. A theory context is a declaration. Integers beyond 2^53-1 must use canonical decimal strings in ordinal terms. Serialization accepts bounded finite JSON data and never evaluates code. Typed deserialization validates against the current catalogue where applicable.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A lowercase 64-character hexadecimal SHA-256 digest of canonical serialization.

Examples

```
fingerprint(theory_context())
```

---

finite_poset *Construct a partial order where p <= q means p is the stronger condition*

Description

Construct a partial order where p <= q means p is the stronger condition.

Usage

```
finite_poset(domain, order)
```

Arguments

domain	A finite collection of unique nonempty string labels, or a symbolic domain label in a norm/scale specification.
order	A complete finite relation table. Posets include reflexive pairs and use smaller-is-stronger.

Details

All conclusions are restricted to the supplied finite data. They do not witness infinite cardinals, genericity over a full infinite model or infinite determinacy. Invalid domains, missing interpretations and exhausted budgets raise classified errors. Tree depth is at most 100. Relation analysis uses at most 100 domain labels. Exhaustive partition queries are intentionally restricted.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A finite_poset record with a checked reflexive partial order and convention smaller_is_stronger.

Examples

```
finite_poset(list("p", "q", "top"), list(list("p", "p"), list("q",
    "q"), list("top", "top"), list("p", "top"), list("q", "top")))
```

`finite_skolem_hull` *Close finite generators under explicitly supplied total finitary operations*

Description

Close finite generators under explicitly supplied total finitary operations.

This is closure under the supplied functions. Calling them Skolem functions does not establish that they witness every existential formula of a theory.

Usage

```
finite_skolem_hull(domain, operations, generators, max_steps = 100000L)
```

Arguments

`domain`	A finite collection of unique nonempty string labels, or a symbolic domain label in a norm/scale specification.
`operations`	Named total operation tables. Each has an arity and a table of args/value records, one for every argument tuple.
`generators`	The finite subset whose closure under the supplied operations is constructed.
`max_steps`	A positive operation budget. Finite algorithms accept at most 10000000; exhaustion raises RESOURCE_LIMIT.

Details

All conclusions are restricted to the supplied finite data. They do not witness infinite cardinals, genericity over a full infinite model or infinite determinacy. Invalid domains, missing interpretations and exhausted budgets raise classified errors. Tree depth is at most 100. Relation analysis uses at most 100 domain labels. Exhaustive partition queries are intentionally restricted.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A FINITE_EXACT result with the generated hull and a derivation of its elements from the supplied operation tables. Full elementarity requires a complete Skolem family.

Examples

```
finite_skolem_hull(list("a", "b", "c", "d"), list(f = list(arity = 1,
    table = list(list(args = list("a"), value = "b"), list(args = list("b"),
        value = "c"), list(args = list("c"), value = "c"), list(args = list("d"),
        value = "d")))), list("a"))
```

---

finite_structure *Specify a finite relational structure; relation tables include their arity*

Description

Specify a finite relational structure; relation tables include their arity.

Usage

```
finite_structure(domain, relations = list(), constants = list(), max_size = 1000L)
```

Arguments

domain	A finite collection of unique nonempty string labels, or a symbolic domain label in a norm/scale specification.
relations	Named finite relation tables; each has arity and tuples. All tuple entries must belong to the domain.
constants	A named mapping of constant symbols to domain labels.
max_size	Maximum finite domain size; the constructor also imposes limits on relation tables.

Details

All conclusions are restricted to the supplied finite data. They do not witness infinite cardinals, genericity over a full infinite model or infinite determinacy. Invalid domains, missing interpretations and exhausted budgets raise classified errors. Tree depth is at most 100. Relation analysis uses at most 100 domain labels. Exhaustive partition queries are intentionally restricted.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A canonical finite_structure record with sorted domain, relation tables and constant interpretations.

Examples

```
finite_structure(list("a", "b"), list(P = list(arity = 1, tuples = list(list("a")))))
```

finite_tree *Build the prefix closure of finitely many finite paths, including the root*

Description

Build the prefix closure of finitely many finite paths, including the root.

Usage

```
finite_tree(paths, max_nodes = 10000L)
```

Arguments

paths	Finite paths, each a sequence of nonempty string labels. Prefix closure and the root are added.
max_nodes	Maximum prefix-closed tree size; individual paths have depth at most 100.

Details

All conclusions are restricted to the supplied finite data. They do not witness infinite cardinals, genericity over a full infinite model or infinite determinacy. Invalid domains, missing interpretations and exhausted budgets raise classified errors. Tree depth is at most 100. Relation analysis uses at most 100 domain labels. Exhaustive partition queries are intentionally restricted.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A finite_tree record containing the root and all prefixes of the supplied paths.

Examples

```
finite_tree(list(list("a", "x"), list("a", "y"), list("b", "x"), list("b",
    "y")))
```

finite_ultraproduct *Return the factor representing a finite-index principal ultraproduct*

Description

Return the factor representing a finite-index principal ultraproduct.

Usage

```
finite_ultraproduct(structures, principal_index)
```

Arguments

structures	Nonempty list of finite structures with the same relation and constant signature.
principal_index	Zero-based index of the selected factor, in both Python and R.

Details

All conclusions are restricted to the supplied finite data. They do not witness infinite cardinals, genericity over a full infinite model or infinite determinacy. Invalid domains, missing interpretations and exhausted budgets raise classified errors. Tree depth is at most 100. Relation analysis uses at most 100 domain labels. Exhaustive partition queries are intentionally restricted.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A FINITE_EXACT query_result whose witness is the finite structure of the factor selected by the principal ultrafilter.

Examples

```
finite_ultraproduct(list(finite_structure(list("a", "b"), list(P = list(arity = 1,
    tuples = list(list("a"))))), finite_structure(list("a", "b"),
    list(P = list(arity = 1, tuples = list(list("a")))))), 0)
```

---

forall *Universally bind a variable in a formula*

Description

Universally bind a variable in a formula.

Usage

```
forall(bound, body)
```

Arguments

bound	The variable expression bound by the quantifier.
body	A formula expression.

Details

Expressions are sorted syntactic trees. Normalization changes bound names only, not logical content. Substitution avoids capture. A theory context is a declaration. Integers beyond 2^53-1 must use canonical decimal strings in ordinal terms. Serialization accepts bounded finite JSON data and never evaluates code. Typed deserialization validates against the current catalogue where applicable.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A universally quantified formula expression with an explicit bound variable.

Examples

```
forall(variable("x"), predicate("P", variable("x")))
```

---

forcing_spec *Describe a forcing configuration; no generic filter is constructed*

Description

Describe a forcing configuration; no generic filter is constructed.

Usage

```
forcing_spec(poset = "P", ground_model = "M", generic_filter = "G",
    closure = NULL, chain_condition = NULL, context = NULL)
```

Arguments

poset	A finite_poset record, or a symbolic forcing-poset label in forcing_spec.
ground_model	A symbolic ground-model label.
generic_filter	A symbolic filter label; genericity is not established by the declaration.
closure	A symbolic closure hypothesis, or unspecified.
chain_condition	A symbolic chain-condition hypothesis, or unspecified.
context	An explicit theory_context record, or NULL/None for the default ZFC context.

Details

Specifications describe infinite objects and retain explicit obligations. Boolean attributes are caller hypotheses; unspecified is not false. No existence, well-foundedness, elementarity or iterability proof is inferred. Mahlo iteration starts with the (weakly) inaccessible class at zero, uses stationary trace at successors, and ordinary intersection at limits. Indescribability levels require polarity and nonnegative finite order and block count (the field named alternations). Order zero uses first-order prenex complexity, not the bounded-quantifier Levy hierarchy.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A canonical specification record with family, concept, parameters, context, obligations, status SYMBOLIC and existence NOT_ASSERTED.

Examples

```
forcing_spec()
```

---

free_variables *Return sorted free variable names; shadowing is lexical*

Description

Return sorted free variable names; shadowing is lexical.

Usage

```
free_variables(expr)
```

Arguments

expr	A typed expression built by the public syntax constructors.

Details

Expressions are sorted syntactic trees. Normalization changes bound names only, not logical content. Substitution avoids capture. A theory context is a declaration. Integers beyond 2^53-1 must use canonical decimal strings in ordinal terms. Serialization accepts bounded finite JSON data and never evaluates code. Typed deserialization validates against the current catalogue where applicable.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

The sorted collection of names of free variables.

Examples

```
free_variables(forall(variable("y"), member(variable("x"), variable("y"))))
```

`game_spec` *Describe an infinite game; solving finite games is a separate operation*

Description

Describe an infinite game; solving finite games is a separate operation.

Usage

```
game_spec(moves = "naturals", length = NULL, payoff = "A", pointclass = "arbitrary",
    context = NULL)
```

Arguments

`moves`	The symbolic move-space label.
`length`	An exact ordinal length. Games default to omega.
`payoff`	A symbolic payoff-set label.
`pointclass`	A symbolic pointclass label. Queries accept borel, analytic, coanalytic, projective or arbitrary.
`context`	An explicit theory_context record, or NULL/None for the default ZFC context.

Details

Specifications describe infinite objects and retain explicit obligations. Boolean attributes are caller hypotheses; unspecified is not false. No existence, well-foundedness, elementarity or iterability proof is inferred. Mahlo iteration starts with the (weakly) inaccessible class at zero, uses stationary trace at successors, and ordinary intersection at limits. Indescribability levels require polarity and nonnegative finite order and block count (the field named alternations). Order zero uses first-order prenex complexity, not the bounded-quantifier Levy hierarchy.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A canonical specification record with family, concept, parameters, context, obligations, status SYMBOLIC and existence NOT_ASSERTED.

Examples

```
game_spec()
```

`homogeneous_subset` *Find a homogeneous subset of a complete explicitly supplied finite coloring*

Description

Find a homogeneous subset of a complete explicitly supplied finite coloring.

Usage

```
homogeneous_subset(vertices, arity, coloring, target_size, max_steps = 100000L)
```

Arguments

`vertices`	A finite collection of unique nonempty string labels to be colored.
`arity`	A nonnegative finite arity, or finite_subsets where explicitly supported.
`coloring`	A complete finite table of subset/color records for the chosen arity.
`target_size`	The required size of a homogeneous subset.
`max_steps`	A positive operation budget. Finite algorithms accept at most 10000000; exhaustion raises RESOURCE_LIMIT.

Details

All conclusions are restricted to the supplied finite data. They do not witness infinite cardinals, genericity over a full infinite model or infinite determinacy. Invalid domains, missing interpretations and exhausted budgets raise classified errors. Tree depth is at most 100. Relation analysis uses at most 100 domain labels. Exhaustive partition queries are intentionally restricted.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A FINITE_EXACT query_result with a homogeneous subset when found, or a negative result after exhaustive bounded search.

Examples

```
homogeneous_subset(c("a", "b"), 2, list(list(subset = c("a", "b"),
    color = "red")), 2)
```

`ideal_spec` *Describe an ideal with explicit completeness, saturation and properness*

Description

Describe an ideal with explicit completeness, saturation and properness.

All properties are relative to relative_model, defaulting to the context model. In particular, the carrier alone asserts neither completeness nor inclusion of all bounded subsets. Unspecified flags remain unknown.

Usage

```
ideal_spec(carrier = "kappa", completeness = NULL, saturation = NULL,
    normal = NULL, proper = NULL, contains_bounded = NULL, relative_model = NULL,
    context = NULL)
```

Arguments

`carrier`	A symbolic carrier label.
`completeness`	An exact ordinal completeness parameter, or unspecified.
`saturation`	An exact ordinal saturation parameter, or unspecified.
`normal`	A Boolean normality hypothesis, or unspecified.
`proper`	Declared exclusion of the entire carrier from the ideal; NULL/None is unspecified.
`contains_bounded`	Declared inclusion of every bounded subset of the ordinal carrier in the ideal; NULL/None is unspecified.
`relative_model`	Model in which the filter or ideal hypotheses are interpreted; defaults to the context model.
`context`	An explicit theory_context record, or NULL/None for the default ZFC context.

Details

Specifications describe infinite objects and retain explicit obligations. Boolean attributes are caller hypotheses; unspecified is not false. No existence, well-foundedness, elementarity or iterability proof is inferred. Mahlo iteration starts with the (weakly) inaccessible class at zero, uses stationary trace at successors, and ordinary intersection at limits. Indescribability levels require polarity and nonnegative finite order and block count (the field named alternations). Order zero uses first-order prenex complexity, not the bounded-quantifier Levy hierarchy.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A canonical specification record with family, concept, parameters, context, obligations, status SYMBOLIC and existence NOT_ASSERTED.

Examples

```
ideal_spec()
```

`implication` *Form material implication in one fixed interpretation*

Description

Form material implication in one fixed interpretation.

Usage

```
implication(left, right)
```

Arguments

`left`	The left operand, or a condition label for compatibility queries.
`right`	The right operand, or a condition label for compatibility queries.

Details

Expressions are sorted syntactic trees. Normalization changes bound names only, not logical content. Substitution avoids capture. A theory context is a declaration. Integers beyond 2^53-1 must use canonical decimal strings in ordinal terms. Serialization accepts bounded finite JSON data and never evaluates code. Typed deserialization validates against the current catalogue where applicable.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A formula expression for material implication.

Examples

```
implication(predicate("P"), predicate("Q"))
```

`inner_model_spec` *Distinguish L[A], L(A), L(R) and other inner-model constructions*

Description

Distinguish L[A], L(A), L(R) and other inner-model constructions.

Usage

```
inner_model_spec(construction = "L", predicate = NULL, ambient_model = "V",
    axioms = list("ZF"), context = NULL)
```

Arguments

`construction`	L, L[A], L(A), L(R), HOD, L[U] or extender_model. These constructions are not identified with each other.
`predicate`	A symbolic predicate label, or unspecified.
`ambient_model`	A symbolic ambient-model label.
`axioms`	The axioms hypothesized for the specified inner model.
`context`	An explicit theory_context record, or NULL/None for the default ZFC context.

Details

Specifications describe infinite objects and retain explicit obligations. Boolean attributes are caller hypotheses; unspecified is not false. No existence, well-foundedness, elementarity or iterability proof is inferred. Mahlo iteration starts with the (weakly) inaccessible class at zero, uses stationary trace at successors, and ordinary intersection at limits. Indescribability levels require polarity and nonnegative finite order and block count (the field named alternations). Order zero uses first-order prenex complexity, not the bounded-quantifier Levy hierarchy.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A canonical specification record with family, concept, parameters, context, obligations, status SYMBOLIC and existence NOT_ASSERTED.

Examples

```
inner_model_spec()
```

`iteration_spec` *Describe iteration rules, ordinal length and a separate iterability hypothesis*

Description

Describe iteration rules, ordinal length and a separate iterability hypothesis.

Usage

```
iteration_spec(model = "M", length = "alpha", successor_rule = "ultrapower",
    limit_rule = "direct_limit", iterability = NULL, context = NULL)
```

Arguments

`model`	The ambient model label (V by default), or the model label of an ultrapower/iteration.
`length`	An exact ordinal length. Games default to omega.
`successor_rule`	The declared successor construction for an iteration.
`limit_rule`	The declared limit construction for an iteration.
`iterability`	A Boolean iterability hypothesis, not a computed iteration proof.
`context`	An explicit theory_context record, or NULL/None for the default ZFC context.

Details

Specifications describe infinite objects and retain explicit obligations. Boolean attributes are caller hypotheses; unspecified is not false. No existence, well-foundedness, elementarity or iterability proof is inferred. Mahlo iteration starts with the (weakly) inaccessible class at zero, uses stationary trace at successors, and ordinary intersection at limits. Indescribability levels require polarity and nonnegative finite order and block count (the field named alternations). Order zero uses first-order prenex complexity, not the bounded-quantifier Levy hierarchy.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A canonical specification record with family, concept, parameters, context, obligations, status SYMBOLIC and existence NOT_ASSERTED.

Examples

```
iteration_spec()
```

`laver_table` *Construct a finite Laver table on 1..2^n and check left distributivity*

Description

Construct a finite Laver table on 1..2^n and check left distributivity.

Use 0 <= n <= 7. The first column is successor modulo 2^n, with 2^n representing zero. Construction takes O(4^n) space and the exhaustive law check takes O(8^n) time. Finite tables require no rank-embedding assumption.

Usage

```
laver_table(n, max_steps = 1000000L)
```

Arguments

`n`	A finite source size, or the logarithmic table index in laver_table (table size 2^n).
`max_steps`	A positive operation budget. Finite algorithms accept at most 10000000; exhaustion raises RESOURCE_LIMIT.

Details

All conclusions are restricted to the supplied finite data. They do not witness infinite cardinals, genericity over a full infinite model or infinite determinacy. Invalid domains, missing interpretations and exhausted budgets raise classified errors. Tree depth is at most 100. Relation analysis uses at most 100 domain labels. Exhaustive partition queries are intentionally restricted.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A FINITE_EXACT query_result with the operation table, size, checked left-distributive law and number of tested triples.

Examples

```
laver_table(0)
```

list_definitions *List available definitions, optionally filtered by topic and object kind*

Description

List available definitions, optionally filtered by topic and object kind.

Usage

```
list_definitions(topic = NULL, kind = NULL)
```

Arguments

topic	An optional topic ID from list_topics.
kind	An explicitly named constructor, definition kind or relation type; inspect the relevant function details.

Details

Catalogue premises are internally reviewed mathematical statements, not formal-kernel proofs. Only applicable same-cardinal implications are transitively composed. Other relation kinds remain direct. A missing relation returns UNKNOWN. Named theory-extension records are assumption schemas; their existence claims require explicit caller assumptions. Rechecking a derivation validates its trusted edges and composition, not the underlying theorems.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

The sorted collection of definition identifiers matching the optional filters.

Examples

```
list_definitions()
```

list_statements *List the IDs of curated mathematical statements*

Description

List the IDs of curated mathematical statements.

Usage

```
list_statements(topic = NULL)
```

Arguments

topic	An optional topic ID from list_topics.

Details

Catalogue premises are internally reviewed mathematical statements, not formal-kernel proofs. Only applicable same-cardinal implications are transitively composed. Other relation kinds remain direct. A missing relation returns UNKNOWN. Named theory-extension records are assumption schemas; their existence claims require explicit caller assumptions. Rechecking a derivation validates its trusted edges and composition, not the underlying theorems.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

The sorted collection of statement identifiers matching the optional topic.

Examples

```
list_statements()
```

---

list_topics *List mathematical topic IDs present in the installed catalogue*

Description

List mathematical topic IDs present in the installed catalogue.

Usage

```
list_topics()
```

Details

Catalogue premises are internally reviewed mathematical statements, not formal-kernel proofs. Only applicable same-cardinal implications are transitively composed. Other relation kinds remain direct. A missing relation returns UNKNOWN. Named theory-extension records are assumption schemas; their existence claims require explicit caller assumptions. Rechecking a derivation validates its trusted edges and composition, not the underlying theorems.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

The sorted collection of installed mathematical topic identifiers.

Examples

```
list_topics()
```

lookup_statement *Inspect a statement without asserting that its assumptions hold*

Description

Inspect a statement without asserting that its assumptions hold.

Usage

```
lookup_statement(identifier)
```

Arguments

identifier	A public catalogue identifier; unknown IDs raise an input error.

Details

Catalogue premises are internally reviewed mathematical statements, not formal-kernel proofs. Only applicable same-cardinal implications are transitively composed. Other relation kinds remain direct. A missing relation returns UNKNOWN. Named theory-extension records are assumption schemas; their existence claims require explicit caller assumptions. Rechecking a derivation validates its trusted edges and composition, not the underlying theorems.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

An independent statement record including its conditional text, requirements, model, kind and evidence.

Examples

```
lookup_statement("pressing_down")
```

mahlo_iteration_spec *Describe M_0, M_(alpha+1), and limit intersections without enumerating them*

Description

Describe M_0, M_(alpha+1), and limit intersections without enumerating them.

M_0 is the (weakly) inaccessible class. M_(a+1) consists of members kappa of M_0 for which M_a intersect kappa is stationary in kappa. M_lambda is the intersection of all M_a for a < lambda. No diagonal limit convention or hyper-Mahlo fixed point is silently substituted.

Usage

```
mahlo_iteration_spec(index, weak = FALSE, context = NULL)
```

Arguments

index	An exact ordinal term, decimal integer or symbolic ordinal name.
weak	Whether to use weak inaccessibility as the initial Mahlo convention.
context	An explicit theory_context record, or NULL/None for the default ZFC context.

Details

Specifications describe infinite objects and retain explicit obligations. Boolean attributes are caller hypotheses; unspecified is not false. No existence, well-foundedness, elementarity or iterability proof is inferred. Mahlo iteration starts with the (weakly) inaccessible class at zero, uses stationary trace at successors, and ordinary intersection at limits. Indescribability levels require polarity and nonnegative finite order and block count (the field named alternations). Order zero uses first-order prenex complexity, not the bounded-quantifier Levy hierarchy.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A canonical specification record with family, concept, parameters, context, obligations, status SYMBOLIC and existence NOT_ASSERTED.

Examples

```
mahlo_iteration_spec(ordinal_term("omega"))
```

---

mahlo_property *Describe the strong or weak Mahlo property, with its convention explicit*

Description

Describe the strong or weak Mahlo property, with its convention explicit.

Usage

```
mahlo_property(kappa = "kappa", weak = FALSE, context = NULL)
```

Arguments

kappa	An ASCII symbolic name for the cardinal under discussion.
weak	Whether to use weak inaccessibility as the initial Mahlo convention.
context	An explicit theory_context record, or NULL/None for the default ZFC context.

Details

Specifications describe infinite objects and retain explicit obligations. Boolean attributes are caller hypotheses; unspecified is not false. No existence, well-foundedness, elementarity or iterability proof is inferred. Mahlo iteration starts with the (weakly) inaccessible class at zero, uses stationary trace at successors, and ordinary intersection at limits. Indescribability levels require polarity and nonnegative finite order and block count (the field named alternations). Order zero uses first-order prenex complexity, not the bounded-quantifier Levy hierarchy.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A canonical specification record with family, concept, parameters, context, obligations, status SYMBOLIC and existence NOT_ASSERTED.

Examples

```
mahlo_property()
```

`measure_spec` *Describe a full-carrier measure without conflating real and two-valued measures*

Description

Describe a full-carrier measure without conflating real and two-valued measures.

Usage

```
measure_spec(carrier = "kappa", additivity = NULL, zero_singletons = NULL,
    two_valued = NULL, context = NULL)
```

Arguments

`carrier`	A symbolic carrier label.
`additivity`	An exact ordinal additivity parameter, or unspecified.
`zero_singletons`	Whether singleton-nullness is hypothesized; unspecified is distinct from false.
`two_valued`	Whether the measure is hypothesized to take only two values.
`context`	An explicit theory_context record, or NULL/None for the default ZFC context.

Details

Specifications describe infinite objects and retain explicit obligations. Boolean attributes are caller hypotheses; unspecified is not false. No existence, well-foundedness, elementarity or iterability proof is inferred. Mahlo iteration starts with the (weakly) inaccessible class at zero, uses stationary trace at successors, and ordinary intersection at limits. Indescribability levels require polarity and nonnegative finite order and block count (the field named alternations). Order zero uses first-order prenex complexity, not the bounded-quantifier Levy hierarchy.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A canonical specification record with family, concept, parameters, context, obligations, status SYMBOLIC and existence NOT_ASSERTED.

Examples

```
measure_spec()
```

---

member *Form set membership; a proper-class term cannot occur on the left*

Description

Form set membership; a proper-class term cannot occur on the left.

Usage

```
member(left, right)
```

Arguments

`left`	The left operand, or a condition label for compatibility queries.
`right`	The right operand, or a condition label for compatibility queries.

Details

Expressions are sorted syntactic trees. Normalization changes bound names only, not logical content. Substitution avoids capture. A theory context is a declaration. Integers beyond 2^53-1 must use canonical decimal strings in ordinal terms. Serialization accepts bounded finite JSON data and never evaluates code. Typed deserialization validates against the current catalogue where applicable.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A membership formula preserving the two supplied terms.

Examples

```
member(variable("x"), variable("y"))
```

`negate` *Form the negation of a formula*

Description

Form the negation of a formula.

Usage

```
negate(body)
```

Arguments

`body`	A formula expression.

Details

Expressions are sorted syntactic trees. Normalization changes bound names only, not logical content. Substitution avoids capture. A theory context is a declaration. Integers beyond 2^53-1 must use canonical decimal strings in ordinal terms. Serialization accepts bounded finite JSON data and never evaluates code. Typed deserialization validates against the current catalogue where applicable.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A formula expression for negation.

Examples

```
negate(predicate("P"))
```

normalize — *Canonicalize bound names only; this does not decide logical equivalence*

Description

Canonicalize bound names only; this does not decide logical equivalence.

Usage

```
normalize(expr)
```

Arguments

expr	A typed expression built by the public syntax constructors.

Details

Expressions are sorted syntactic trees. Normalization changes bound names only, not logical content. Substitution avoids capture. A theory context is a declaration. Integers beyond 2^53-1 must use canonical decimal strings in ordinal terms. Serialization accepts bounded finite JSON data and never evaluates code. Typed deserialization validates against the current catalogue where applicable.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

An alpha-normalized expression. Other logical equivalences are not simplified.

Examples

```
normalize(forall(variable("y"), member(variable("x"), variable("y"))))
```

norm_spec — *Describe an ordinal-valued norm and its pointclass/rank obligations*

Description

Describe an ordinal-valued norm and its pointclass/rank obligations.

Usage

```
norm_spec(domain = "A", pointclass = "Gamma", rank_bound = NULL, context = NULL)
```

Arguments

domain	A finite collection of unique nonempty string labels, or a symbolic domain label in a norm/scale specification.
pointclass	A symbolic pointclass label. Queries accept borel, analytic, coanalytic, projective or arbitrary.
rank_bound	An exact ordinal rank bound, or unspecified.
context	An explicit theory_context record, or NULL/None for the default ZFC context.

Details

Specifications describe infinite objects and retain explicit obligations. Boolean attributes are caller hypotheses; unspecified is not false. No existence, well-foundedness, elementarity or iterability proof is inferred. Mahlo iteration starts with the (weakly) inaccessible class at zero, uses stationary trace at successors, and ordinary intersection at limits. Indescribability levels require polarity and nonnegative finite order and block count (the field named alternations). Order zero uses first-order prenex complexity, not the bounded-quantifier Levy hierarchy.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A canonical specification record with family, concept, parameters, context, obligations, status SYMBOLIC and existence NOT_ASSERTED.

Examples

```
norm_spec()
```

---

ordinal_term *Construct an exact symbolic ordinal term, not an evaluated infinite set*

Description

Construct an exact symbolic ordinal term, not an evaluated infinite set.

Usage

```
ordinal_term(kind, ...)
```

Arguments

kind	An explicitly named constructor, definition kind or relation type; inspect the relevant function details.
...	Formula operands, ordinal operands, or named concept parameters, as required by the selected constructor.

Details

Expressions are sorted syntactic trees. Normalization changes bound names only, not logical content. Substitution avoids capture. A theory context is a declaration. Integers beyond 2^53-1 must use canonical decimal strings in ordinal terms. Serialization accepts bounded finite JSON data and never evaluates code. Typed deserialization validates against the current catalogue where applicable.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

An ordinal_term syntax tree with kind and args. Infinite arithmetic is not evaluated.

Examples

```
ordinal_term("finite", "900719925474099312345")
```

`parse_formula` *Parse bounded S-expressions: '(forall x (implies (P x) (P x)))'*

Description

Parse bounded S-expressions: '(forall x (implies (P x) (P x)))'.

Usage

```
parse_formula(source, max_tokens = 10000L)
```

Arguments

`source`	Source text for the parser, or a symbolic source/model label for a specification.
`max_tokens`	Maximum number of parser tokens; excessively deep inputs are also rejected.

Details

Expressions are sorted syntactic trees. Normalization changes bound names only, not logical content. Substitution avoids capture. A theory context is a declaration. Integers beyond 2^53-1 must use canonical decimal strings in ordinal terms. Serialization accepts bounded finite JSON data and never evaluates code. Typed deserialization validates against the current catalogue where applicable.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A typed expression parsed from the documented bounded prefix syntax.

Examples

```
parse_formula("(forall x (implies (P x) (P x)))")
```

`partition_relation_holds` *Exhaustively decide a small finite Ramsey relation, subject to a strict budget*

Description

Exhaustively decide a small finite Ramsey relation, subject to a strict budget.

Usage

```
partition_relation_holds(n, target, arity = 2L, colors = 2L, max_cases = 100000L)
```

Arguments

`n`	A finite source size, or the logarithmic table index in laver_table (table size 2^n).
`target`	A finite homogeneous-set size or symbolic target label, depending on the function.
`arity`	A nonnegative finite arity, or finite_subsets where explicitly supported.
`colors`	A finite color count for exhaustive algorithms, or an exact symbolic ordinal parameter.
`max_cases`	Maximum exhaustive work budget, accounting for colorings, candidates and subsets.

Details

All conclusions are restricted to the supplied finite data. They do not witness infinite cardinals, genericity over a full infinite model or infinite determinacy. Invalid domains, missing interpretations and exhausted budgets raise classified errors. Tree depth is at most 100. Relation analysis uses at most 100 domain labels. Exhaustive partition queries are intentionally restricted.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A FINITE_EXACT result for the finite partition relation; a failing coloring is returned when the relation is refuted.

Examples

```
partition_relation_holds(3, 2)
```

partition_spec *Describe a partition assertion with separate coloring and witness models*

Description

Describe a partition assertion with separate coloring and witness models.

Both models default to the ambient context. Restricting colorings to L while allowing homogeneous witnesses in V is distinct from asserting the relation internally in L. Square brackets without color_bound require omission of at least one color; with color_bound they require strictly fewer colors than that bound. Ordinary arrows require one color and reject color_bound. Arity may be a finite integer, an ordinal term or symbolic ordinal label, or finite_subsets for all finite sizes. Ordinary arrows require separate homogeneity at each size; square brackets omit a color or bound the image across all sizes together. An ordinal arity means subsets of that order type, not merely that cardinality. Finite enumeration has its own API.

Usage

```
partition_spec(source = "kappa", target = "kappa", arity = 2, colors = 2,
    variant = "ordinary", color_bound = NULL, coloring_model = NULL,
    witness_model = NULL, context = NULL)
```

Arguments

source	Source text for the parser, or a symbolic source/model label for a specification.
target	A finite homogeneous-set size or symbolic target label, depending on the function.
arity	A nonnegative finite integer, ordinal term, symbolic ordinal label, or finite_subsets. Ordinal arity refers to subsets of that order type, not just cardinality. For ordinary arrows, finite_subsets requires separate homogeneity at each finite size; for square brackets, omission or image-size bounds apply across all sizes together. Infinite arities are symbolic only.
colors	A finite color count for exhaustive algorithms, or an exact symbolic ordinal parameter.
variant	ordinary or square_bracket partition convention.
color_bound	For square brackets, an exact ordinal specifying a strict bound on the number of colors used. NULL/None means at least one color is omitted, which need not reduce an infinite cardinality. Ordinary arrows require NULL/None.
coloring_model	The model supplying colorings; defaults to the ambient context model.
witness_model	The model supplying homogeneous witnesses; defaults to the ambient context model. It may differ from the coloring model.
context	An explicit theory_context record, or NULL/None for the default ZFC context.

Details

Specifications describe infinite objects and retain explicit obligations. Boolean attributes are caller hypotheses; unspecified is not false. No existence, well-foundedness, elementarity or iterability proof is inferred. Mahlo iteration starts with the (weakly) inaccessible class at zero, uses stationary trace at successors, and ordinary intersection at limits. Indescribability levels require polarity and nonnegative finite order and block count (the field named alternations). Order zero uses first-order prenex complexity, not the bounded-quantifier Levy hierarchy.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A canonical specification record with family, concept, parameters, context, obligations, status SYMBOLIC and existence NOT_ASSERTED.

Examples

```
partition_spec()
```

---

pointclass_spec *Describe a hierarchy level and its quantifier domain*

Description

Describe a hierarchy level and its quantifier domain.

Families are borel, projective, arithmetic, analytic_hierarchy and third_order_arithmetic. The last denotes Sigma^2_n/Pi^2_n/Delta^2_n, with quantification over sets of reals; it is not projective level two. The implementation supports finite levels, not transfinite Borel ranks. No effective third-order convention is implemented.

Usage

```
pointclass_spec(family = "projective", level = 1, polarity = "Sigma",
    effective = FALSE, parameters = list(), context = NULL)
```

Arguments

`family`	The hierarchy family: borel, projective, arithmetic, analytic_hierarchy or third_order_arithmetic. The third-order family quantifies over sets of reals, not only reals.
`level`	A finite exact hierarchy level. Borel, projective and third-order families start at one; arithmetic and analytical families permit zero. Transfinite Borel ranks are not accepted.
`polarity`	Sigma, Pi or Delta.
`effective`	Whether to use the effective convention; named real parameters are then the explicit oracle parameters for relativization.

parameters	Named symbolic real parameters, or concept-specific keyword parameters for cardinal_property.
context	An explicit theory_context record, or NULL/None for the default ZFC context.

Details

Specifications describe infinite objects and retain explicit obligations. Boolean attributes are caller hypotheses; unspecified is not false. No existence, well-foundedness, elementarity or iterability proof is inferred. Mahlo iteration starts with the (weakly) inaccessible class at zero, uses stationary trace at successors, and ordinary intersection at limits. Indescribability levels require polarity and nonnegative finite order and block count (the field named alternations). Order zero uses first-order prenex complexity, not the bounded-quantifier Levy hierarchy.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A canonical specification record with family, concept, parameters, context, obligations, status SYMBOLIC and existence NOT_ASSERTED.

Examples

```
pointclass_spec()
```

---

predicate *Form an atomic formula; zero-arity predicates represent propositions*

Description

Form an atomic formula; zero-arity predicates represent propositions.

Usage

```
predicate(name, ...)
```

Arguments

name	A mathematical identifier. Formula identifiers use ASCII letters, digits, underscores and dots, starting with a letter or underscore.
...	Formula operands, ordinal operands, or named concept parameters, as required by the selected constructor.

Details

Expressions are sorted syntactic trees. Normalization changes bound names only, not logical content. Substitution avoids capture. A theory context is a declaration. Integers beyond 2^53-1 must use canonical decimal strings in ordinal terms. Serialization accepts bounded finite JSON data and never evaluates code. Typed deserialization validates against the current catalogue where applicable.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A formula expression applying a named relation to the supplied terms.

Examples

```
predicate("P", variable("x"))
```

---

preservation_query *Query small-forcing preservation using an explicit symbolic size hypothesis*

Description

Query small-forcing preservation using an explicit symbolic size hypothesis.

Usage

```
preservation_query(property, forcing_size, context = NULL)
```

Arguments

`property`	The requested regularity or preservation property.
`forcing_size`	The supported symbolic hypothesis below_kappa; this is not a numerical comparison of infinities.
`context`	An explicit theory_context record, or NULL/None for the default ZFC context.

Details

Catalogue premises are internally reviewed mathematical statements, not formal-kernel proofs. Only applicable same-cardinal implications are transitively composed. Other relation kinds remain direct. A missing relation returns UNKNOWN. Named theory-extension records are assumption schemas; their existence claims require explicit caller assumptions. Rechecking a derivation validates its trusted edges and composition, not the underlying theorems.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A CATALOGUE result for the selected conditional small-forcing preservation theorem, or UNKNOWN. The forcing and its size are not verified.

Examples

```
preservation_query("inaccessible", "below_kappa")
```

`prewellorder_ranks` *Compute quotient ranks of a finite total preorder*

Description

Compute quotient ranks of a finite total preorder.

Usage

```
prewellorder_ranks(domain, order)
```

Arguments

`domain`	A finite collection of unique nonempty string labels, or a symbolic domain label in a norm/scale specification.
`order`	A complete finite relation table. Posets include reflexive pairs and use smaller-is-stronger.

Details

All conclusions are restricted to the supplied finite data. They do not witness infinite cardinals, genericity over a full infinite model or infinite determinacy. Invalid domains, missing interpretations and exhausted budgets raise classified errors. Tree depth is at most 100. Relation analysis uses at most 100 domain labels. Exhaustive partition queries are intentionally restricted.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A FINITE_EXACT query_result with ordered equivalence classes, ranks and quotient length. A relation that is not a total preorder raises an input error.

Examples

```
prewellorder_ranks(list("a", "b", "c"), list(list("a", "a"), list("b",
    "b"), list("c", "c"), list("b", "c"), list("c", "b"), list("b",
    "a"), list("c", "a")))
```

`projective_ordinal` *Return the symbolic value of delta^1_level inside a ZF+AD+DC model*

Description

Return the symbolic value of delta^1_level inside a ZF+AD+DC model.

Levels 1 through 32 are supported. The result is an exact ordinal-term tree, not a numerical approximation. All alephs are computed in the context model; no bound on alephs in a larger universe is inferred. Missing AD or DC returns NOT_APPLICABLE; contradictory contexts are rejected.

Usage

```
projective_ordinal(level, context = NULL)
```

Arguments

`level`	An exact integer from 1 through 32. Zero is invalid and larger levels exceed the supported resource bound.
`context`	An explicit theory_context record, or NULL/None for the default ZFC context.

Details

Catalogue premises are internally reviewed mathematical statements, not formal-kernel proofs. Only applicable same-cardinal implications are transitively composed. Other relation kinds remain direct. A missing relation returns UNKNOWN. Named theory-extension records are assumption schemas; their existence claims require explicit caller assumptions. Rechecking a derivation validates its trusted edges and composition, not the underlying theorems.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A CATALOGUE query_result whose witness contains the exact aleph_index, ordinal and model. Missing ZF/AD/DC yields NOT_APPLICABLE; inconsistent contexts are rejected.

Examples

```
projective_ordinal(1, context = theory_context("ZF_DC", list("AD"),
    model = "L(R)"))
```

`query_relation` *Search only context-compatible relations; absence of a path is UNKNOWN*

Description

Search only context-compatible relations; absence of a path is UNKNOWN.

Usage

```
query_relation(premise, conclusion, kind = "IMPLIES_SAME_OBJECT",
    context = NULL, max_steps = 10000L)
```

Arguments

`premise`	A known node ID or a cardinal_property specification.
`conclusion`	A known node ID or a cardinal_property specification.
`kind`	An explicitly named constructor, definition kind or relation type; inspect the relevant function details.
`context`	An explicit theory_context record, or NULL/None for the default ZFC context.
`max_steps`	A positive operation budget. Finite algorithms accept at most 10000000; exhaustion raises RESOURCE_LIMIT.

Details

Catalogue premises are internally reviewed mathematical statements, not formal-kernel proofs. Only applicable same-cardinal implications are transitively composed. Other relation kinds remain direct. A missing relation returns UNKNOWN. Named theory-extension records are assumption schemas; their existence claims require explicit caller assumptions. Rechecking a derivation validates its trusted edges and composition, not the underlying theorems.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A scoped query_result. Supported implications carry a hash-bound derivation in witness; absent evidence yields UNKNOWN and unsupported specializations yield UNSUPPORTED.

Examples

```
query_relation("shelah", "woodin")
```

`regularity_query` *Retrieve selected regularity implications with explicit theory assumptions*

Description

Retrieve selected regularity implications with explicit theory assumptions.

Usage

```
regularity_query(pointclass, property, context = NULL)
```

Arguments

`pointclass`	A symbolic pointclass label. Queries accept borel, analytic, coanalytic, projective or arbitrary.
`property`	The requested regularity or preservation property.
`context`	An explicit theory_context record, or NULL/None for the default ZFC context.

Details

Catalogue premises are internally reviewed mathematical statements, not formal-kernel proofs. Only applicable same-cardinal implications are transitively composed. Other relation kinds remain direct. A missing relation returns UNKNOWN. Named theory-extension records are assumption schemas; their existence claims require explicit caller assumptions. Rechecking a derivation validates its trusted edges and composition, not the underlying theorems.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A CATALOGUE query_result for one of the supported regularity implications, with its assumptions and dependencies. No converse is inferred.

Examples

```
regularity_query("borel", "perfect_set_property")
```

`relation_graph` *Return a directed multigraph of applicable, direct catalogue relations*

Description

Return a directed multigraph of applicable, direct catalogue relations.

“nodes“ selects an induced subgraph. By default only same-object implications are selected. Pass explicit kinds to include other relations. Paths in a mixed graph are not mathematical derivations; use query_relation.

Usage

```
relation_graph(nodes = NULL, kinds = NULL, context = NULL)
```

Arguments

`nodes`	Optional known node IDs selecting an induced subgraph. An empty collection produces an empty graph.
`kinds`	Explicit relation kinds to include. The default is only IMPLIES_SAME_OBJECT; no transitive edges are added.
`context`	An explicit theory_context record, or NULL/None for the default ZFC context.

Details

The graph contains direct reviewed relations compatible with the context. Every edge retains its kind, prerequisites, source/target models and evidence. Parallel relations are preserved. A path through mixed edge kinds is not a proof. Layout coordinates do not measure cardinal size or consistency strength. NetworkX and igraph are optional; the symbolic core does not import them. GraphML writers may require encoding list-valued metadata before export.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A relation_graph record with context, nodes, typed direct edges, catalogue fingerprint and limitations.

Examples

```
relation_graph()
```

`relativize` *Restrict set-sorted quantifiers to membership in a domain term*

Description

Restrict set-sorted quantifiers to membership in a domain term.

Universal quantifiers acquire a membership antecedent and existential quantifiers acquire a membership conjunct. The domain is a set or class variable/constant. Bound variables are renamed to avoid capturing a free domain variable. Atomic formulas are unchanged. Only set-sorted binders are supported; higher-order satisfaction is not constructed.

Usage

```
relativize(expr, domain)
```

Arguments

`expr`	A typed expression built by the public syntax constructors.
`domain`	A set- or class-sorted variable or constant representing the domain. It is a term, not a finite label collection. A colliding binder is renamed before inserting membership guards.

Details

Expressions are sorted syntactic trees. Normalization changes bound names only, not logical content. Substitution avoids capture. A theory context is a declaration. Integers beyond 2^53-1 must use canonical decimal strings in ordinal terms. Serialization accepts bounded finite JSON data and never evaluates code. Typed deserialization validates against the current catalogue where applicable.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A formula expression with its set-sorted quantifiers restricted to membership in the supplied domain term. Free domain variables remain free.

Examples

```
relativize(parse_formula("(forall x (exists y (eq x y)))"), variable("M"))
```

`scale_spec` *Describe a scale with named norms and an explicit limit-property obligation*

Description

Describe a scale with named norms and an explicit limit-property obligation.

Usage

```
scale_spec(domain = "A", pointclass = "Gamma", norms = list(), limit_property = NULL,
    context = NULL)
```

Arguments

`domain`	A finite collection of unique nonempty string labels, or a symbolic domain label in a norm/scale specification.
`pointclass`	A symbolic pointclass label. Queries accept borel, analytic, coanalytic, projective or arbitrary.
`norms`	Labels of the declared norms; a finite list is a description, not construction of a complete infinite scale.
`limit_property`	A Boolean declaration of the scale limit property, or unspecified.
`context`	An explicit theory_context record, or NULL/None for the default ZFC context.

Details

Specifications describe infinite objects and retain explicit obligations. Boolean attributes are caller hypotheses; unspecified is not false. No existence, well-foundedness, elementarity or iterability proof is inferred. Mahlo iteration starts with the (weakly) inaccessible class at zero, uses stationary trace at successors, and ordinary intersection at limits. Indescribability levels require polarity and nonnegative finite order and block count (the field named alternations). Order zero uses first-order prenex complexity, not the bounded-quantifier Levy hierarchy.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A canonical specification record with family, concept, parameters, context, obligations, status SYMBOLIC and existence NOT_ASSERTED.

Examples

```
scale_spec()
```

serialize_record *Canonical UTF-8 JSON with sorted keys, no floats and no executable objects*

Description

Canonical UTF-8 JSON with sorted keys, no floats and no executable objects.

Usage

```
serialize_record(value)
```

Arguments

value	The value or typed record to validate, serialize or fingerprint.

Details

Expressions are sorted syntactic trees. Normalization changes bound names only, not logical content. Substitution avoids capture. A theory context is a declaration. Integers beyond 2^53-1 must use canonical decimal strings in ordinal terms. Serialization accepts bounded finite JSON data and never evaluates code. Typed deserialization validates against the current catalogue where applicable.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

Canonical JSON text with deterministic object-key ordering and exact supported values.

Examples

```
largecardinals::serialize_record(theory_context())
```

small_subset_partition_spec *Describe homogeneity for chains in P_kappa(gamma)*

Description

Describe homogeneity for chains in P_kappa(gamma).

proper_inclusion uses strict subset chains. size_separated also requires |x| < |y intersect kappa| at each step. Homogeneous families are unbounded under inclusion, stationary, or members of the explicitly named filter. The background hypotheses kappa regular uncountable and gamma>=kappa remain obligations. No infinite coloring is enumerated or decided.

Usage

```
small_subset_partition_spec(kappa = "kappa", gamma = "gamma", arity = 2,
   colors = 2, chain_order = "proper_inclusion", homogeneous_requirement = "stationary",
    filter_name = NULL, coloring_model = NULL, witness_model = NULL,
    context = NULL)
```

Arguments

kappa	An ASCII symbolic name for the cardinal under discussion.
gamma	Symbolic ambient ordinal for P_kappa(gamma); gamma>=kappa is a mathematical obligation.
arity	A nonnegative finite arity, or finite_subsets where explicitly supported.
colors	A finite color count for exhaustive algorithms, or an exact symbolic ordinal parameter.
chain_order	proper_inclusion for strict subset chains, or size_separated with the additional inequality |x|<|y intersect kappa|.
homogeneous_requirement	unbounded, stationary or filter_member, describing the required size property of the homogeneous subfamily.
filter_name	A symbolic filter name, required exactly when homogeneous_requirement is filter_member.
coloring_model	The model supplying colorings; defaults to the ambient context model.
witness_model	The model supplying homogeneous witnesses; defaults to the ambient context model. It may differ from the coloring model.
context	An explicit theory_context record, or NULL/None for the default ZFC context.

Details

Specifications describe infinite objects and retain explicit obligations. Boolean attributes are caller hypotheses; unspecified is not false. No existence, well-foundedness, elementarity or iterability proof is inferred. Mahlo iteration starts with the (weakly) inaccessible class at zero, uses stationary trace at successors, and ordinary intersection at limits. Indescribability levels require polarity and nonnegative finite order and block count (the field named alternations). Order zero uses first-order prenex complexity, not the bounded-quantifier Levy hierarchy.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A canonical specification record with family, concept, parameters, context, obligations, status SYMBOLIC and existence NOT_ASSERTED.

Examples

```
small_subset_partition_spec()
```

`solve_finite_game` *Solve a finite perfect-information alternating game by backward induction*

Description

Solve a finite perfect-information alternating game by backward induction.

Usage

```
solve_finite_game(tree, payoffs)
```

Arguments

`tree`	A canonical finite_tree record.
`payoffs`	One path/winner record for every terminal node; players are labelled 0 and 1.

Details

All conclusions are restricted to the supplied finite data. They do not witness infinite cardinals, genericity over a full infinite model or infinite determinacy. Invalid domains, missing interpretations and exhausted budgets raise classified errors. Tree depth is at most 100. Relation analysis uses at most 100 domain labels. Exhaustive partition queries are intentionally restricted.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A FINITE_EXACT query_result identifying the winner and a winning strategy for the supplied finite game.

Examples

```
solve_finite_game(finite_tree(list(list("a", "x"), list("a", "y"),
    list("b", "x"), list("b", "y"))), list(list(path = list("a", "x"),
    winner = 0), list(path = list("a", "y"), winner = 0), list(path = list("b",
    "x"), winner = 1), list(path = list("b", "y"), winner = 1)))
```

statement_result — *Return a trusted catalogue result only when its explicit context is applicable*

Description

Return a trusted catalogue result only when its explicit context is applicable.

Usage

```
statement_result(identifier, context = NULL)
```

Arguments

identifier	A public catalogue identifier; unknown IDs raise an input error.
context	An explicit theory_context record, or NULL/None for the default ZFC context.

Details

Catalogue premises are internally reviewed mathematical statements, not formal-kernel proofs. Only applicable same-cardinal implications are transitively composed. Other relation kinds remain direct. A missing relation returns UNKNOWN. Named theory-extension records are assumption schemas; their existence claims require explicit caller assumptions. Rechecking a derivation validates its trusted edges and composition, not the underlying theorems.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A CATALOGUE query_result for the named conditional statement. Research questions remain UNKNOWN; theory-extension assertions require explicit assumptions.

Examples

```
statement_result("pressing_down")
```

substitute_formula — *Replace free occurrences, renaming binders to avoid capture*

Description

Replace free occurrences, renaming binders to avoid capture.

Usage

```
substitute_formula(expr, name, replacement)
```

Arguments

expr	A typed expression built by the public syntax constructors.
name	A mathematical identifier. Formula identifiers use ASCII letters, digits, underscores and dots, starting with a letter or underscore.
replacement	A term expression of the same sort as the variable being substituted.

Details

Expressions are sorted syntactic trees. Normalization changes bound names only, not logical content. Substitution avoids capture. A theory context is a declaration. Integers beyond 2^53-1 must use canonical decimal strings in ordinal terms. Serialization accepts bounded finite JSON data and never evaluates code. Typed deserialization validates against the current catalogue where applicable.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A new expression with free occurrences substituted and bound variables renamed when necessary to avoid capture.

Examples

```
largecardinals::substitute_formula(forall(variable("y"), member(variable("x"),
    variable("y"))), "x", variable("y"))
```

---

theory_context *Declare a context; valid syntax never establishes consistency*

Description

Declare a context; valid syntax never establishes consistency.

Usage

```
theory_context(base = "ZFC", assumptions = character(), model = "V",
    class_framework = "SCHEMA")
```

Arguments

base	ZF, ZF_DC, ZFC or FOL. ZFC contributes AC and DC; declaring a base does not prove its consistency.
assumptions	A collection of explicit assumption labels; declarations are not proofs.
model	The ambient model label (V by default), or the model label of an ultrapower/iteration.
class_framework	NONE, SCHEMA, NBG or GB; specifies the convention for discussing class embeddings.

Details

Expressions are sorted syntactic trees. Normalization changes bound names only, not logical content. Substitution avoids capture. A theory context is a declaration. Integers beyond 2^53-1 must use canonical decimal strings in ordinal terms. Serialization accepts bounded finite JSON data and never evaluates code. Typed deserialization validates against the current catalogue where applicable.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A theory_context record carrying base, assumptions, model and class_framework; consistency_status is NOT_ESTABLISHED.

Examples

```
theory_context()
```

---

to_igraph *Create an igraph directed multigraph preserving mathematical metadata*

Description

Create an igraph directed multigraph preserving mathematical metadata.

Usage

```
to_igraph(graph = NULL, nodes = NULL, kinds = NULL, context = NULL)
```

Arguments

graph	An unmodified relation_graph record, or omitted when selecting by filters.
nodes	Optional known node IDs selecting an induced subgraph. An empty collection produces an empty graph.
kinds	Explicit relation kinds to include. The default is only IMPLIES_SAME_OBJECT; no transitive edges are added.
context	An explicit theory_context record, or NULL/None for the default ZFC context.

Details

The graph contains direct reviewed relations compatible with the context. Every edge retains its kind, prerequisites, source/target models and evidence. Parallel relations are preserved. A path through mixed edge kinds is not a proof. Layout coordinates do not measure cardinal size or consistency strength. NetworkX and igraph are optional; the symbolic core does not import them. GraphML writers may require encoding list-valued metadata before export.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

An igraph directed multigraph with node and edge metadata preserved; the optional igraph dependency is required.

Examples

```
if (requireNamespace("igraph", quietly = TRUE)) {
    g <- to_igraph(nodes = c("measurable", "ramsey"))
    igraph::ecount(g)
}
```

---

transitive_collapse *Collapse a finite well-founded extensional relation into nested finite sets*

Description

Collapse a finite well-founded extensional relation into nested finite sets.

Usage

```
transitive_collapse(domain, pairs, max_steps = 1000000L)
```

Arguments

domain	A finite collection of unique nonempty string labels, or a symbolic domain label in a norm/scale specification.
pairs	The complete finite relation as pairs of domain labels.
max_steps	A positive operation budget. Finite algorithms accept at most 10000000; exhaustion raises RESOURCE_LIMIT.

Details

All conclusions are restricted to the supplied finite data. They do not witness infinite cardinals, genericity over a full infinite model or infinite determinacy. Invalid domains, missing interpretations and exhausted budgets raise classified errors. Tree depth is at most 100. Relation analysis uses at most 100 domain labels. Exhaustive partition queries are intentionally restricted.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A FINITE_EXACT query_result with the finite transitive collapse when the relation is well founded and extensional.

Examples

```
transitive_collapse(list("zero", "one", "two"), list(list("zero",
    "one"), list("zero", "two"), list("one", "two")))
```

tree_branches *List maximal finite branches; no infinite branch is asserted*

Description

List maximal finite branches; no infinite branch is asserted.

Usage

```
tree_branches(tree)
```

Arguments

tree	A canonical finite_tree record.

Details

All conclusions are restricted to the supplied finite data. They do not witness infinite cardinals, genericity over a full infinite model or infinite determinacy. Invalid domains, missing interpretations and exhausted budgets raise classified errors. Tree depth is at most 100. Relation analysis uses at most 100 domain labels. Exhaustive partition queries are intentionally restricted.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A FINITE_EXACT query_result listing the maximal finite branches of the supplied tree.

Examples

```
tree_branches(finite_tree(list(list("a", "x"), list("a", "y"), list("b",
    "x"), list("b", "y"))))
```

tree_code *Describe a tree representation without asserting membership of its projection*

Description

Describe a tree representation without asserting membership of its projection.

Usage

```
tree_code(alphabet = "naturals", height = NULL, projection = NULL,
    well_founded = NULL, context = NULL)
```

Arguments

alphabet	A symbolic label for the tree alphabet.
height	An exact ordinal height; tree codes default to omega.
projection	A symbolic projection label, or unspecified.
well_founded	A Boolean well-foundedness hypothesis; unspecified does not mean false.
context	An explicit theory_context record, or NULL/None for the default ZFC context.

Details

Specifications describe infinite objects and retain explicit obligations. Boolean attributes are caller hypotheses; unspecified is not false. No existence, well-foundedness, elementarity or iterability proof is inferred. Mahlo iteration starts with the (weakly) inaccessible class at zero, uses stationary trace at successors, and ordinary intersection at limits. Indescribability levels require polarity and nonnegative finite order and block count (the field named alternations). Order zero uses first-order prenex complexity, not the bounded-quantifier Levy hierarchy.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A canonical specification record with family, concept, parameters, context, obligations, status SYMBOLIC and existence NOT_ASSERTED.

Examples

```
tree_code()
```

---

ultrapower_spec *Describe an ultrapower without assuming its well-foundedness*

Description

Describe an ultrapower without assuming its well-foundedness.

Usage

```
ultrapower_spec(model = "V", ultrafilter = "U", well_founded = NULL,
    context = NULL)
```

Arguments

model	The ambient model label (V by default), or the model label of an ultrapower/iteration.
ultrafilter	In filter_spec, a Boolean declaration or NULL/None. In ultrapower_spec, a symbolic ultrafilter label.
well_founded	A Boolean well-foundedness hypothesis; unspecified does not mean false.
context	An explicit theory_context record, or NULL/None for the default ZFC context.

Details

Specifications describe infinite objects and retain explicit obligations. Boolean attributes are caller hypotheses; unspecified is not false. No existence, well-foundedness, elementarity or iterability proof is inferred. Mahlo iteration starts with the (weakly) inaccessible class at zero, uses stationary trace at successors, and ordinary intersection at limits. Indescribability levels require polarity and nonnegative finite order and block count (the field named alternations). Order zero uses first-order prenex complexity, not the bounded-quantifier Levy hierarchy.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A canonical specification record with family, concept, parameters, context, obligations, status SYMBOLIC and existence NOT_ASSERTED.

Examples

```
ultrapower_spec()
```

---

validate_context *Check negation and AC/AD conflicts after a fixed closure of named axioms*

Description

Check negation and AC/AD conflicts after a fixed closure of named axioms.

In ZF, AD_R implies AD, AD implies PD and AC_omega_reals, and DC implies AC_omega_reals. Full AC supplies DC. No converse is assumed.

Usage

```
validate_context(context)
```

Arguments

context	An explicit theory_context record, or NULL/None for the default ZFC context.

Details

Expressions are sorted syntactic trees. Normalization changes bound names only, not logical content. Substitution avoids capture. A theory context is a declaration. Integers beyond 2^53-1 must use canonical decimal strings in ordinal terms. Serialization accepts bounded finite JSON data and never evaluates code. Typed deserialization validates against the current catalogue where applicable.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A SYMBOLIC query_result: INCONSISTENT_CONTEXT for a detected conflict, otherwise UNKNOWN; witness lists detected conflicts.

Examples

```
validate_context(theory_context("ZFC", list()))
```

`validate_record` *Validate interchange structure and known semantic record types*

Description

Validate interchange structure and known semantic record types.

Usage

```
validate_record(value)
```

Arguments

`value`	The value or typed record to validate, serialize or fingerprint.

Details

Expressions are sorted syntactic trees. Normalization changes bound names only, not logical content. Substitution avoids capture. A theory context is a declaration. Integers beyond 2^53-1 must use canonical decimal strings in ordinal terms. Serialization accepts bounded finite JSON data and never evaluates code. Typed deserialization validates against the current catalogue where applicable.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

The validated typed record or expression; this is data validation, not mathematical proof.

Examples

```
validate_record(theory_context())
```

`validate_specification` *Reconstruct a specification and reject unknown or altered semantic fields*

Description

Reconstruct a specification and reject unknown or altered semantic fields.

Usage

```
validate_specification(value)
```

Arguments

`value`	The value or typed record to validate, serialize or fingerprint.

Details

Specifications describe infinite objects and retain explicit obligations. Boolean attributes are caller hypotheses; unspecified is not false. No existence, well-foundedness, elementarity or iterability proof is inferred. Mahlo iteration starts with the (weakly) inaccessible class at zero, uses stationary trace at successors, and ordinary intersection at limits. Indescribability levels require polarity and nonnegative finite order and block count (the field named alternations). Order zero uses first-order prenex complexity, not the bounded-quantifier Levy hierarchy.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

The canonical reconstructed specification; altered semantic fields or inconsistent encodings raise classified input errors.

Examples

```
validate_specification(embedding_spec())
```

variable — *Declare a variable of the given mathematical sort*

Description

Declare a variable of the given mathematical sort.

Usage

```
variable(name, sort = "set")
```

Arguments

name	A mathematical identifier. Formula identifiers use ASCII letters, digits, underscores and dots, starting with a letter or underscore.
sort	One of set, ordinal, cardinal, class, real or natural. The finite evaluator accepts only set-sorted terms.

Details

Expressions are sorted syntactic trees. Normalization changes bound names only, not logical content. Substitution avoids capture. A theory context is a declaration. Integers beyond 2^53-1 must use canonical decimal strings in ordinal terms. Serialization accepts bounded finite JSON data and never evaluates code. Typed deserialization validates against the current catalogue where applicable.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A typed variable expression with the supplied name and sort.

Examples

```
variable("x")
```

variants — *List explicitly supported variants of selected cardinal-property families*

Description

List explicitly supported variants of selected cardinal-property families.

Usage

```
variants(concept)
```

Arguments

concept	A definition ID; cardinal constructors require kind cardinal_property.

Details

Specifications describe infinite objects and retain explicit obligations. Boolean attributes are caller hypotheses; unspecified is not false. No existence, well-foundedness, elementarity or iterability proof is inferred. Mahlo iteration starts with the (weakly) inaccessible class at zero, uses stationary trace at successors, and ordinary intersection at limits. Indescribability levels require polarity and nonnegative finite order and block count (the field named alternations). Order zero uses first-order prenex complexity, not the bounded-quantifier Levy hierarchy.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

The collection of supported variant definition identifiers for the supplied concept.

Examples

```
variants("mahlo")
```

---

verify_finite_strategy

Check all opponent continuations of a supplied finite winning strategy

---

Description

Check all opponent continuations of a supplied finite winning strategy.

Usage

```
verify_finite_strategy(tree, payoffs, winner, strategy)
```

Arguments

tree	A canonical finite_tree record.
payoffs	One path/winner record for every terminal node; players are labelled 0 and 1.
winner	Player 0 or 1 whose proposed strategy is checked.
strategy	Records with path and move fields; all reachable decisions for the player must be supplied.

Details

All conclusions are restricted to the supplied finite data. They do not witness infinite cardinals, genericity over a full infinite model or infinite determinacy. Invalid domains, missing interpretations and exhausted budgets raise classified errors. Tree depth is at most 100. Relation analysis uses at most 100 domain labels. Exhaustive partition queries are intentionally restricted.

Input errors inherit from largecardinals_error and expose a code. No function publishes data or accesses the network.

Value

A FINITE_EXACT result checking the proposed strategy at every reachable decision of that player.

Examples

```
verify_finite_strategy(finite_tree(list(list("a", "x"), list("a",
    "y"), list("b", "x"), list("b", "y"))), list(list(path = list("a",
    "x"), winner = 0), list(path = list("a", "y"), winner = 0), list(path = list("b",
    "x"), winner = 1), list(path = list("b", "y"), winner = 1)), 0,
    list(list(path = list(), move = "a")))
```

Index

add_assumption, 3
analyze_finite_ideal, 4
analyze_relation, 5

boolean_truth, 6
borel_hierarchy_spec, 7

cardinal_iteration_spec, 8
cardinal_property, 9
catalogue_manifest, 10
check_applicability, 11
check_derivation, 11
check_finite_embedding, 12
check_finite_generic, 13
closure_requirements, 14
compatible_conditions, 15
conjunction, 16
constant, 16
critical_point_spec, 17

definition, 18
deserialize, 19
determinacy_query, 19
difference_hierarchy_spec, 20
difference_layers, 21
disjunction, 22

embedding_spec, 23
equal, 24
equivalence, 25
evaluate_formula, 26
exists_formula, 27
explain_derivation, 27
extender_spec, 28

filter_spec, 29
fingerprint, 30
finite_poset, 31
finite_skolem_hull, 32
finite_structure, 33
finite_tree, 34
finite_ultraproduct, 34
forall, 35
forcing_spec, 36
free_variables, 37

game_spec, 38

homogeneous_subset, 39

ideal_spec, 40
implication, 41
inner_model_spec, 42
iteration_spec, 43

laver_table, 44
list_definitions, 45
list_statements, 45
list_topics, 46
lookup_statement, 47

mahlo_iteration_spec, 47
mahlo_property, 48
measure_spec, 49
member, 50

negate, 51
norm_spec, 52
normalize, 52

ordinal_term, 53

parse_formula, 54
partition_relation_holds, 55
partition_spec, 56
pointclass_spec, 57
predicate, 58
preservation_query, 59
prewellorder_ranks, 60
projective_ordinal, 61

query_relation, 62

regularity_query, 63
relation_graph, 64
relativize, 65

scale_spec, 66
serialize_record, 67
small_subset_partition_spec, 67
solve_finite_game, 69
statement_result, 70
substitute_formula, 70

theory_context, 71
to_igraph, 72
transitive_collapse, 73
tree_branches, 74
tree_code, 74

ultrapower_spec, 75

validate_context, 76
validate_record, 77
validate_specification, 78
variable, 79
variants, 79
verify_finite_strategy, 80