The Universal Logos (UL) Handbook

Patrick A. Gessner

A Complete Encyclopedia, Study Guide, and Reference Manual for Human–AI Communication

Contents

The Universal Logos (UL) Handbook A Complete Encyclopedia, Study Guide, and Reference Manual for Human–AI Communication

1 Introduction & Foundations

1.1 Why a Universal Language?

1.1.1 The Human Challenge

Human languages are rich, diverse, and deeply tied to culture — but they are also ambiguous and inefficient. The same word can mean many different things depending on tone, context, or history. For example: - Bank a financial institution or the side of a river. - She saw him with a telescope who has the telescope? Humans navigate these ambiguities intuitively, but even between people, miscommunication is common. For artificial intelligences, which rely on explicit structure, these ambiguities become critical flaws: a system cannot "guess" correctly every time without error.

1.1.2 The AI Challenge

AI systems today process natural language with great skill, but this ability comes at enormous cost: - Computational waste: large language models must interpret ambiguous forms probabilistically, consuming far more resources than necessary. - Uncertainty: two systems trained on different corpora may not interpret the same sentence the same way. - Error propagation: in fields like medicine, law, or engineering, a single misinterpretation can cause cascading problems. What AIs need is a shared core protocol: a universal way to encode meaning that is precise, compact, and machine-native — but still learnable by humans.

1.1.3 Lessons from Linguistic Universals

Linguistics shows us that despite surface diversity, all human languages share deep common features: - Reference: pointing to entities, actions, and relations. - Recursion: combining smaller units into infinitely large structures. - Time & modality: situating events in time, and expressing possibility or necessity. - Negation & contrast: distinguishing what is from what is not. - Interaction frame: marking who is speaking, who is addressed, and who is being discussed. These meta-elements form the blueprint for a language that both humans and machines can use.

1.1.4 The Goal of UNI-LOGOS

UNI-LOGOS (UL) is designed to be: 1. Precise — no hidden meanings, no ambiguity. 2. Efficient — compact structure, easily compressed, minimal redundancy. 3. Transparent — every element is typed and explicit (time, units, evidence, certainty). 4. Dual-channel — a canonical core for machines (formal, symbolic), and a surface gloss for humans (simple, predictable). 5. Extensible — able to grow into any domain (science, law, daily life). 6. Interoperable — transportable as text, JSON, or speech.

1.1.5 Why One Book?

This handbook is designed as a single complete source that serves three functions: - Textbook — explaining concepts clearly for human learners. - Encyclopedia — documenting every rule, predicate, and structure. - Reference manual — providing schemas, glosses, and quick-look tables.

Humans can study it progressively; AIs can parse it directly. It is not a “translation” of natural language, but a new operating system for communication.

1.1.6 Looking Ahead

In the next chapters, we will explore: - Chapter 2: The deep elements all human languages share. - Chapter 3: The design principles of UNI-LOGOS. - Chapters 4–7: The formal specification and syntax. - Chapters 8–13: Human learning guides and exercises. - Chapters 14–18: Encyclopedic lexicon of predicates and roles. - Chapters 19–24: Reference material for AI–AI and human–AI interoperation. By the end of this book, both human and AI readers will be able to understand, produce, and extend UNI-LOGOS fluently.

1.2 Meta-Elements Shared by All Human Languages

1.2.1 The Search for Universals

When linguists compared hundreds of languages — from English to Inuktitut, from Thai to Basque, from signed to spoken forms — they discovered that beneath surface differences, all human languages share certain structures. These are called linguistic universals. UNI-LOGOS is built directly on these universals, abstracted into a machine-readable form. If something exists in every natural language, it is a safe foundation for a universal system.

1.2.2 Discrete Units and Duality of Structure

Every language breaks speech or signs into small meaningless units (sounds, gestures, letters). These combine into meaningful units (morphemes, words). Words then combine into larger structures (phrases, sentences). This layered combinatorial system is universal. In UL: Core atoms (ENT, REL, TIME, MOD, etc.) form the smallest meaningful pieces, which can be combined into larger statements without ambiguity.

1.2.3 Reference to Entities, Actions, and Relations

Entities (things, persons, objects): all languages can point to them (nouns, pronouns). Actions or events: all languages describe what happens (verbs). Relations: languages express connections (prepositions, case markers, word order). In UL: Every statement explicitly encodes a predicate (REL) and its arguments (roles such as agent, theme, location).

1.2.4 Recursion and Productivity

Natural languages allow infinite combinations from finite parts. Example: The man [who saw the woman [who had a dog [that barked]]] smiled. In UL: Nested S-expressions allow infinite depth — but with explicit scope markers, so recursion never creates ambiguity.

1.2.5 Time and Aspect

Every language locates events in time: - Some use tense (past, present, future). - Some use aspect (completed, ongoing, habitual). - Even languages without tense use temporal adverbs (yesterday, soon). In UL: Time is always explicit: :time { tense past at 2023-05-01 }.

1.2.6 Modality

Languages universally express possibility, necessity, ability, obligation. Examples: - English: may, must, can. - German: dürfen, müssen, können. - Thai: dai (can), tong (must). In UL: Modality is typed (epistemic, deontic, ability) with clear values (must, may, can).

1.2.7 Negation

Every language can flip truth values — say “no / not / never.” Without negation, logical reasoning collapses. In UL: Negation is always a field: :neg true.

1.2.8 Interaction Frame

All languages distinguish speaker, addressee, and third person. They manage turn-taking and perspective. In UL: Each message has an :agent role, :recipient, and metadata in the envelope (from, to).

1.2.9 Evidentiality and Certainty

Many languages encode how you know something: - Turkish marks whether you saw it or heard it. - Quechua marks report vs.inference. - English often uses adverbs: apparently, probably. In UL: Evidence (:evid) and confidence (:conf) are mandatory. Machines cannot omit them.

1.2.10 Summary Table of Universals UL Mapping

1.2.11 Key Takeaway

These universals are not cultural accidents — they reflect the cognitive architecture of humans. By encoding them directly, UNI-LOGOS becomes both human-learnable and AI-native. Where natural languages differ, UL stays constant: it makes every element visible, typed, and explicit.

1.3 Principles of UNI-LOGOS Design

1.3.1 From Universals to Engineering

Chapter 2 showed that all human languages share meta-elements: reference, recursion, time, modality, negation, interaction, and evidentiality. UNI-LOGOS (UL) takes these elements and re-engineers them into a formal system: - No ambiguity. - No cultural dependency. - Every piece of information explicitly typed. This chapter introduces the guiding principles of design that make UL efficient for machines and usable for humans.

1.3.2 Principle 1 — Explicitness

Natural language thrives on implication: - "See you later" no time given. - "It's hot" relative to context. For AIs, implication is dangerous. UL requires everything to be explicit: - Time must always be marked (:time). - Numbers must have units (:unit). - Confidence must be declared (:conf). - Evidence must be given (:evid). Nothing is left for interpretation outside the message.

1.3.3 Principle 2 — Minimal Core Atoms

Instead of thousands of rules, UL builds on a small set of atoms, inspired by linguistic universals. Core atoms: - ENT (entity), REL (relation/predicate), TIME, MOD (modality), NEG, EVID, CONF, UNIT, REF, LINK, SCOPE. From these, all complex expressions can be built. UL is both small (easy to learn) and powerful (unlimited expressivity).

1.3.4 Principle 3 — Dual-Channel Communication

UL is always encoded in two layers: 1. Canonical Core — S-expressions or JSON. Precise, machine-readable. 2. Human Surface Gloss — compact, regularized text that humans can read/write. Example: Core:

```
(assert
  :event e7
  :pred own
  :args { agent (REF alice) theme (REF car-17) }
  :time { tense past at 2023-05-01 }
  :mod { type epistemic val likely }
  :neg false
  :evid report
  :conf 0.72
)
Surface Gloss: own{agent:Alice, theme:car-17}; TIME[past@2023-05-01];
3.5 Principle 4 - Separation of Concerns
In natural language, meanings overlap: - "must" = necessity, obligatio
UL separates concerns: - Time, modality, negation, evidence, and confi
3.6 Principle 5 - Extensibility
The UL core is minimal, but it must handle all domains: law, medicine,
Solution: namespaces and ontologies. - Predicates (REL) can point to e
 UL scales from simple chat to scientific publishing.
3.7 Principle 6 - Interoperability
UL must flow across systems, networks, and formats. - Serialization: S
 Humans see the surface gloss; AIs process the canonical core; network
3.8 Principle 7 - Error Tolerance and Repair
Human language has repair strategies ("What do you mean?"). UL encodes
```

Misunderstandings don’t cause collapse — they trigger explicit repair.

1.3.5 Principle 8 — Symmetry Between Human and AI Use

UL is not just for AI–AI pipelines; humans can also learn it. - The surface gloss is compact and regular (easier than learning Latin or logic notation). - Core grammar is transparent: no irregular verbs, no idioms.

- UL is designed to be teachable in a study-book format (this handbook itself). Human learners and AI systems can share the same protocol.

1.3.6 Principles Summary

1.3.7 Transition

Now that the principles are clear, we can move into formal specification: - Chapter 4: Message Envelope — the “wrapper” every UL message carries. - Chapter 5–7: Core atoms, syntax, and semantics. These chapters will form the technical backbone of UL.

1.4 The Message Envelope

1.4.1 Purpose of the Envelope

Every UL message travels inside a wrapper called the envelope. - It ensures that meaning is not only in the content but also in the context: who said it, to whom, when, under what integrity guarantee. - Natural languages often drop context (“See you tomorrow” who, when?). UL never does. The envelope is minimal but mandatory.

1.4.2 Structure of the Envelope

UL/1 id: <uuid> # unique identifier for this message from: <agent-id> # sender to: <agent-id|group> # recipient(s) t: <ISO-8601> # timestamp ctx: <thread|topic> # optional conversation ID sig: <hash> # optional integrity signature

1.4.3 Fields Explained

Version (UL/1) — ensures compatibility. id — globally unique message identifier (UUID v4). Prevents confusion and supports tracking. from / to

— sender and addressee(s). Can be individual IDs, groups, or broadcast tokens. t — precise timestamp (UTC, ISO 8601). Required for chronology and synchronization. ctx — optional thread or conversation context. Allows grouping of related exchanges. sig — optional cryptographic hash/signature for message integrity and authentication.

1.4.4 Examples

Simple assertion: UL/1 id: 12ac-b34d from: alice to: bob t: 2025-09-28T08:00:00Z ctx: project-zen With signature: UL/1 id: 89ff-2221 from: sensor-17 to: central-hub t: 2025-09-28T09:00:00Z sig: sha256:ab37c9...

1.4.5 Advantages

Robust tracking: every message is addressable. Auditability: context + signature enable verifiable history. Scalability: works for human conversation and distributed AI networks.

1.4.6 Transition

The envelope is just the shell. Inside lies the core — atoms that carry meaning. That is the subject of the next chapter.

1.5 Core Atoms and Tags

1.5.1 Why Atoms?

Natural languages use thousands of words, but at their base, they all share the same semantic building blocks. UL reduces this to a small atomic inventory. These are like periodic table elements for meaning.

1.5.2 The Core Atoms

ENT — Entities (objects, people, concepts). REL — Relations / predicates (actions, states). TIME — Temporal information. MOD — Modality (possibility, necessity, ability, permission). NEG — Negation. EVID — Evidential source (observation, report, model, sensor, rule). CONF — Confidence level (0–1). UNIT — Measurement unit. REF — Stable reference identifiers. LINK — External pointer (URI, ontology entry). SCOPE — Explicit boundary for quantifiers, negation, conditionals.

1.5.3 Roles (Arguments)

Predicates always have roles. Core inventory: - agent (doer) - patient/theme (acted upon) - recipient (receiving party) - source (origin) - goal (target) - location (place) - instrument (tool used) - beneficiary (who benefits)

1.5.4 Example Atom Use

Event: "Alice gave Bob a book in Paris yesterday."

```
(assert
  :event e1
  :pred give
  :args { agent (REF alice) theme (REF book-1) recipient (REF bob) loc
  :time { tense past at 2025-09-27 }
  :evid report
  :conf 0.95
)
5.5 Summary
The atoms ensure universality and modularity. With ~11 atoms and 7-8 r
5.6 Transition
Now that we know the atoms, we must learn how to combine them. That is
```

```
Chapter 6. Syntax Specification
6.1 Canonical Core Syntax
The core uses S-expressions (nested parentheses, Lisp-style). - Minima
6.2 Basic Pattern
(assert
  :event <id>
  :pred <REL>
  :args { role1 <ENT> role2 <ENT> ... }
  :time { tense <past|present|future> at <ISO|interval> }
  :mod  { type <epistemic|deontic|ability> val <must|may|can|likely> }
  :neg  <true|false>
  :evid <obs|report|model|sensor|rule>
  :conf <0..1>
  :unit <SI unit if numeric>
  :notes <optional free text>
)
```

1.5.5 Types of Statements

Assertions (facts/events) Requests (imperatives) Measures (quantitative reports) Comparisons (logical/evaluative) Repairs/Failures

1.5.6 Human Surface Gloss

Every core message can be glossed into a human-readable line. Rule: PRED{args}; TIME[]; MOD[]; NEG?; EVID[]; CONF=0.95; Example (from e1): give{agent:Alice, theme:book-1, recipient:Bob, location:Paris}; TIME[past@2025-09-27]; EVID[report]; CONF=0.95;

1.5.7 Grammar Formalization (EBNF fragment)

```
message  = "(" statement ")";
```

```
statement = "assert" | "request" | "measure" | "compare" | "repair" |
args      = "{" { role entity } "}";
role      = symbol;
entity    = symbol | REF | LINK;
time      = "{ tense tense-val [ "at" time-val ] }";
tense-val = "past" | "present" | "future";
modality  = "{ type mod-type val mod-val }";
6.6 JSON Schema Equivalence
{
  "type": "object",
  "properties": {
    "event": { "type": "string" },
    "pred": { "type": "string" },
    "args": { "type": "object" },
    "time": { "type": "object" },
    "mod":  { "type": "object" },
    "neg":  { "type": "boolean" },
    "evid": { "type": "string" },
    "conf": { "type": "number" },
    "unit": { "type": "string" }
  },
  "required": ["event","pred","args"]
}
```

1.5.8 Transition

Syntax makes atoms combinable. The next step is semantics — how these structures map to meaning and reasoning.

1.6 Semantics of UL

1.6.1 Semantic Commitment

UL is not just form — it encodes meaning directly. - No idioms. - No hidden metaphors. - Every field corresponds to a semantic dimension.

1.6.2 Predicates and Roles

Semantics are defined by frames: each predicate declares its expected roles. - give: requires agent, theme, recipient. - measure: requires theme, value, unit. Frames are stored in a predicate dictionary (see Part IV).

1.6.3 Time Semantics

UL time is absolute, not relative. - 2025-09-28T08:00Z instead of "yesterday". - Relative phrases are normalized during encoding.

1.6.4 Modality Semantics

UL modality is typed: - epistemic = speaker's certainty. - deontic = obligation/permission. - ability = capacity. Each has fixed values (must, may, can, likely, necessary, possible).

1.6.5 Negation and Scope

UL makes scope explicit: - "Not all dogs bark" vs."All dogs do not bark." Example:

```
(assert
  :event e9
  :pred bark
  :args { theme (forall x (dog x)) }
  :neg true
  :scope dogs
)
7.6 Evidentiality
UL requires source specification: - obs (direct observation) - sensor
```

This makes every statement traceable.

1.6.6 Confidence

Every assertion has a :conf value. - Humans use vague words (probably, maybe). - UL uses numbers (0.0–1.0).

1.6.7 Semantic Integrity

By requiring explicit atoms, UL ensures that: - Every claim is temporally located. - Every number has a unit. - Every claim has evidence and certainty. - Every predicate has roles filled. This prevents semantic drift between humans and machines.

1.6.8 Transition

We now have: - Envelope (Chapter 4) - Atoms (Chapter 5) - Syntax (Chapter 6) - Semantics (Chapter 7) Together, these form the technical backbone of UL.

2 Human Learning Guide

2.1 Basic Sentence Patterns

2.1.1 First Step into UL

Like any language, UL has a starter set of sentence patterns. These patterns cover the most common communicative needs: 1. Making statements (assertions). 2. Asking for actions (requests). 3. Reporting measurements. 4. Making comparisons. 5. Handling repairs. Once you master these, you can build anything.

2.1.2 Assertion (Facts/Events)

Pattern:

```
(assert
  :event <id>
  :pred <REL>
  :args { role1 <ENT> role2 <ENT> ... }
  :time { tense ... at ... }
  :evid <obs|report|model|sensor|rule>
  :conf <0..1>
)
Surface Gloss: PRED{args}; TIME[past@...]; EVID[...]; CONF=...;
Example: - English: "The cat sleeps on the sofa." - UL:
(assert
  :event e1
  :pred sleep
  :args { theme (REF cat-1) location (REF sofa-1) }
  :time { tense present }
  :evid obs
  :conf 1.0
)
```

Gloss: sleep{theme:cat-1, location:sofa-1}; TIME[present]; EVID[obs]; CONF=1.0;

2.1.3 Request (Imperative)

Pattern:

```
(request
  :event <id>
  :pred <REL>
  :args { role1 <ENT> ... }
  :constraints {...}
  :by <deadline>
)
Example: - English: "Send me the file by tomorrow." - UL:
(request
  :event r1
  :pred send
  :args { agent (REF you) theme (REF file-7) recipient (REF me) }
  :by 2025-09-29T00:00:00Z
  :conf 1.0
)
```

2.1.4 Measurement

Pattern:

```
(measure
  :event <id>
  :pred <REL>
  :args { theme <ENT> }
  :value <number>
  :unit <unit>
  :time {...}
  :evid sensor
  :conf <0..1>
)
Example: - English: "The water temperature is 21.3C." - UL:
(measure
  :event m1
  :pred temperature
  :args { theme (REF water-tank-1) }
  :value 21.3
  :unit celsius
  :time { tense present at 2025-09-28T10:00:00Z }
  :evid sensor
  :conf 0.98
)
```

2.1.5 Comparison

Pattern:

```
(compare
  :event <id>
  :pred <REL>
  :args { theme1 <ENT> theme2 <ENT> }
  :relation <equal|greater|less>
)
Example: - English: "Alice is taller than Bob." - UL:
(compare
  :event c1
  :pred height
  :args { theme1 (REF alice) theme2 (REF bob) }
  :relation greater
  :conf 0.9
)
```

2.1.6 Repair

Pattern:

```
(repair
  :on <event-id>
  :need <missing-field>
)
Example: - English: "What unit?" - UL:
(repair
  :on m1
  :need unit
)
```

2.1.7 Exercises

Translate into UL: 1. “The dog barked yesterday.” 2. “Please open the window.” 3. “The weight is 12 kilograms.” 4. “Is Paris larger than Lyon?”

2.2 Expressing Time & Modality

2.2.1 Time

UL always encodes when something happens. - Tense: past, present, future. - Specific timestamp: ISO 8601. - Interval: start–end. Examples: - “Yesterday” :time { tense past at 2025-09-27 } - “From 9 to 10 am” :time { interval { start 09:00 end 10:00 } }

2.2.2 Aspect

UL can express ongoing or completed states: - progressive = still happening. - perfect = completed. Example: “Alice has eaten”

```
(assert
  :event e2
  :pred eat
  :args { agent (REF alice) theme (REF cake-1) }
  :time { tense past aspect perfect }
  :evid report
  :conf 0.95
)
9.3 Modality
UL separates types of modality: - Epistemic = likelihood (likely, poss
Example: - English: "Alice must leave." - UL:
(assert
  :event e3
  :pred leave
  :args { agent (REF alice) }
  :time { tense future }
  :mod { type deontic val must }
  :evid rule
  :conf 1.0
)
```

2.2.3 Combining Time & Modality

“Alice might have left yesterday.”

```
(assert
  :event e4
  :pred leave
  :args { agent (REF alice) }
  :time { tense past at 2025-09-27 }
  :mod { type epistemic val possible }
  :evid report
  :conf 0.6
)
9.5 Exercises
Translate "Bob will probably arrive tomorrow."
```

Translate "You may enter at 10:00."

Chapter 10. Quantifiers & Negation

10.1 Quantifiers

UL expresses quantifiers with explicit scope. - "All dogs bark."

```
(assert
  :event e5
  :pred bark
  :args { theme (forall x (dog x)) }
  :conf 0.9
)
```

“Some dogs bark.”

```
(assert
  :event e6
  :pred bark
  :args { theme (exists x (dog x)) }
  :conf 0.9
)
```

10.2 Negation

Negation is explicit and scoped. - "The cat is not sleeping."

```
(assert
  :event e7
  :pred sleep
  :args { theme (REF cat-1) }
  :time { tense present }
  :neg true
  :conf 1.0
)
```

“Not all dogs bark.”

```
(assert
  :event e8
  :pred bark
  :args { theme (forall x (dog x)) }
  :neg true
  :scope dogs
  :conf 0.95
)
10.3 Exercises
Encode "Some students did not attend."
Encode "All birds cannot swim."
Chapter 11. Evidence & Confidence
11.1 Evidence Types
Every statement must specify source: - obs = direct human observation.
11.2 Confidence
Numeric probability from 0.0 to 1.0. - 1.0 = certain. - 0.5 = uncertai
11.3 Combined Example
"It will probably rain tomorrow (weather model)."
(assert
  :event e9
  :pred rain
  :args { location (REF paris) }
  :time { tense future at 2025-09-29 }
  :mod { type epistemic val likely }
  :evid model
  :conf 0.7
)
```

2.2.4 Exercises

Encode "Apparently, Alice is in Berlin." Encode "The sensor shows 35.1C with 98% certainty."

2.3 Complex Structures

2.3.1 Recursion

UL allows nested clauses. “The man who saw the woman who had a dog smiled.”

```
(assert
  :event e10
  :pred smile
  :args {
    agent (REF man-1)
    cause (assert
      :event e11
      :pred see
      :args {
        agent (REF man-1)
        theme (REF woman-1)
      }
      :sub (assert
        :event e12
        :pred have
        :args { agent (REF woman-1) theme (REF dog-1) }
      )
    )
  }
)
12.2 Conditionals
"If it rains, the game will be canceled."
(assert
  :event e13
  :pred cancel
  :args { theme (REF game-1) }
  :time { tense future }
  :cond (assert
    :event e14
    :pred rain
    :time { tense future }
  )
)
```

2.3.2 Constraints

"Send the file (must be <50MB, format CSV)."

```
(request
  :event r2
  :pred send
  :args { agent (REF you) theme (REF file-9) }
  :constraints { size < 50MB; format csv }
)
12.4 Exercises
Encode "If Alice comes, Bob will leave."
Encode "Every student who passed the exam received a certificate."
Chapter 13. Exercises & Drills
13.1 Beginner
"The lamp is on."
"Please close the door."
"The weight is 10kg."
13.2 Intermediate
"Alice might be in Paris."
"Not all cats like milk."
"Send the report by tomorrow."
13.3 Advanced
```

```
"If Bob doesn't study, he will fail."
"Every researcher who wrote a paper received funding."
"It will probably snow next week, according to the forecast."
13.4 Answer Key (selected)
```

"The lamp is on."

```
(assert
  :event e15
  :pred on
  :args { theme (REF lamp-1) }
  :time { tense present }
  :evid obs
  :conf 1.0
)
"Alice might be in Paris."
(assert
  :event e16
  :pred located_in
  :args { theme (REF alice) location (REF paris) }
  :mod { type epistemic val possible }
  :evid report
  :conf 0.6
)
```

3 Encyclopedic Lexicon

3.1 Core Predicate Lexicon

3.1.1 Purpose

Predicates are the action words of UL. They define what kind of relation is being asserted, requested, or measured. Unlike natural languages, UL predicates are: - Regular — no irregular verbs. - Frame-based — each predicate comes with a fixed set of roles. - Expandable — new domains can add their own namespaces (e.g., bio:binds_to).

3.1.2 Starter Predicate Set

Existence & Identity - be (state, property, identity) - equal (two entities are identical) - exist (entity exists) Possession & Transfer - have (possession) - own (ownership) - give (transfer ownership) Action & Process - do (generic action) - move (agent moves theme to location) - make (agent creates theme) Perception & Communication - see (perceive visually) - hear (perceive auditorily) - say (agent communicates theme to recipient) - ask (agent requests info from recipient) Measurement & Comparison - measure (report quantitative value) - compare (evaluate relation: equal, greater, less) Causality & Permission - cause (agent causes event) - allow (agent permits event) - require (obligation, necessity) Spatial Relations - located_in (entity at place) - part_of (entity is part of whole)

3.1.3 Example

```
(assert
  :event e20
  :pred give
  :args { agent (REF alice) theme (REF book-1) recipient (REF bob) }
  :time { tense past }
  :evid report
  :conf 0.95
)
Gloss: give{agent:Alice, theme:book-1, recipient:Bob}; TIME[past]; EVI
```

14.4 Expansion

Specialized fields (medicine, law, engineering) can define new predica

All must follow the frame principle: clear predicate + fixed roles.

Chapter 15. Role Inventory

15.1 Purpose

Roles connect predicates to entities. Each predicate declares which ro

15.2 Core Roles

agent - the doer (subject in many languages).

theme/patient - the entity acted upon.

recipient - entity receiving something.

source - origin of motion/transfer.

goal - target or destination.

location - place.

instrument - tool used.

cause - event or condition triggering something.

beneficiary - entity that benefits.

experiencer - entity perceiving or feeling.

15.3 Example

"Alice cut the bread with a knife."

```
(assert
  :event e21
  :pred cut
  :args { agent (REF alice) theme (REF bread-1) instrument (REF knife-
  :time { tense past }
  :evid obs
  :conf 1.0
)
```

3.1.4 Benefits

Explicit role labeling no ambiguity. Cross-linguistic neutrality doesn't depend on subject/object word order. Extensible for domain-specific roles (e.g., legal:plaintiff, legal:defendant).

3.2 Modality & Evidentiality Tables

3.2.1 Modality Values

UL distinguishes modality types and values. Epistemic (belief/knowledge): - certain (conf=1.0) - likely - possible - unlikely Deontic (rules/obligations): - must - may - forbidden Ability/Capacity: - can - cannot

3.2.2 Evidentiality

Specifies source of information. - obs = direct human observation. - sensor = device measurement. - report = second-hand information. - model = simulation or forecast. - rule = deduction from law or axiom.

3.2.3 Examples

"Alice must leave (by rule)."

```
(assert
  :event e22
  :pred leave
  :args { agent (REF alice) }
  :time { tense future }
  :mod { type deontic val must }
  :evid rule
  :conf 1.0
)
"It will probably rain (weather model)."
(assert
  :event e23
  :pred rain
  :args { location (REF berlin) }
  :time { tense future at 2025-09-29 }
  :mod { type epistemic val likely }
  :evid model
  :conf 0.7
)
```

3.3 Units & Measures

3.3.1 Necessity

Numbers without units are meaningless. Natural language often omits them ("It weighs 10"). UL enforces mandatory units.

3.3.2 SI Units

Base set: - Length: meter (m) - Mass: kilogram (kg) - Time: second (s) - Temperature: kelvin (K), celsius (C) - Electric current: ampere (A) - Substance: mole (mol) - Luminous intensity: candela (cd)

3.3.3 Derived Units

Velocity: m/s Area: m Volume: m Energy: joule (J) Power: watt (W)

3.3.4 Example

"The mass is 12 kilograms."

```
(measure
  :event m2
  :pred mass
  :args { theme (REF sample-1) }
  :value 12
  :unit kg
  :evid sensor
  :conf 1.0
)
```

17.5 Extensibility

Domains can add custom units: - Finance: USD, EUR, % - Medicine: mg/dL

Chapter 18. Ontology Expansion Guide

18.1 Why Expansion?

The core lexicon is small but must scale. New fields need new terms -

18.2 Namespaces

UL adopts the namespace model: - geo: for geography. - med: for medici

Predicates are prefixed by namespace: - med:diagnose - law:contract_si

18.3 Linking to External Ontologies

UL allows every predicate or entity to carry a :link to an external on

```
(assert
  :event e24
  :pred med:diagnose
  :args { agent (REF doctor-1) theme (REF patient-1) condition (REF di
  :link https://www.wikidata.org/entity/Q1220
  :time { tense past }
  :evid report
  :conf 0.95
)
```

3.3.5 Best Practices

Use short, stable namespaces. Always attach external links where available. Maintain domain dictionaries for consistency. Avoid synonym duplication — one predicate per concept.

4 Reference & Interoperability

4.1 Serialization Formats

4.1.1 Why Multiple Formats?

Different systems require different encodings. UL supports: - S-expressions — default, human-readable. - JSON — widely supported in APIs. - CBOR — binary, compact for networks.

4.1.2 Example (S-expr)

```
(assert
  :event e25
  :pred own
  :args { agent (REF alice) theme (REF car-17) }
  :time { tense present }
  :conf 1.0
)
19.3 Example (JSON)
{
  "event": "e25",
  "pred": "own",
  "args": { "agent": "alice", "theme": "car-17" },
  "time": { "tense": "present" },
  "conf": 1.0
}
```

4.1.3 Example (CBOR)

Binary compact representation — efficient for IoT devices.

4.1.4 Integrity

All messages may include a :sig field with SHA-256 hash or digital signature for verification.

4.2 Error Handling & Repair Protocols

4.2.1 Repair Principle

In natural conversation, misunderstandings trigger repair ("What?"). UL formalizes this: - repair = request missing info. - fail = report task not completed.

4.2.2 Repair Example

```
(repair
  :on m2
  :need unit
)
20.3 Fail Example
(fail
  :of r2
  :reason size_limit_exceeded
)
```

4.2.3 Benefits

Keeps conversation alive. Ensures graceful failure, not silent collapse. Machine + human interpretable.

4.3 Transport & Compression

4.3.1 Transport Agnosticism

UL messages can move over: - HTTP/HTTPS - MQTT (IoT) - Email/SMTP - P2P protocols

4.3.2 Compression

Because UL is structured with repeated tags, gzip/CBOR compression achieves high reduction in size.

4.3.3 Streaming

Messages can be chunked and streamed with envelope headers preserved.

4.4 Mapping to Natural Languages

4.4.1 English

Direct mapping with surface gloss. "Alice owns a car." own{agent:Alice, theme:car-1}; TIME[present]; CONF=1.0;

4.4.2 German

"Alice besitzt ein Auto." same UL core.

4.4.3 Mandarin

“Alice ” same UL.

4.4.4 Signed Languages

Roles map directly: agent, theme, location, etc. Surface gloss can be signed with fixed handshapes.

4.5 Human–AI Communication Scenarios

4.5.1 Dialogue

Human: “Please send the file.” AI:

```
(request
  :event r3
  :pred send
  :args { agent (REF human-1) theme (REF file-1) recipient (REF ai-1)
)
23.2 Negotiation
Human: "By tomorrow?" AI:
(repair
  :on r3
  :need deadline
)
```

4.5.2 Confirmation

Human: “Yes, by tomorrow.” AI updates request with deadline.

4.6 AI–AI Protocol Examples

4.6.1 Sensor Network

Sensor Hub:

```
(measure
  :event m3
  :pred temperature
  :args { theme (REF greenhouse-1) }
  :value 18.7
  :unit celsius
  :time { tense present at 2025-09-28T11:00:00Z }
  :evid sensor
  :conf 0.99
)
24.2 Multi-Agent Planning
Agent A  Agent B:
(request
  :event r4
  :pred fetch
  :args { theme (LINK https://example.org/data.csv) }
  :constraints { format csv size < 10MB }
  :by 2025-09-29T12:00:00Z
)
```

4.6.2 Distributed Knowledge Graph

Agent C asserts new knowledge:

```
(assert
  :event e26
  :pred located_in
  :args { theme (REF factory-7) location (REF berlin) }
  :time { tense present }
  :evid report
  :conf 0.85
)
```

```
Part V - Appendices
Appendix A. Formal Grammar (EBNF + JSON Schema)
```

EBNF (expanded)

```
ul-message  = envelope nl core ;
envelope    = "UL/1" nl "id:" uuid nl "from:" agent nl "to:" agent-lis
core        = { statement } ;
statement   = assert | request | measure | compare | repair | fail ;

assert      = "(" "assert" kvpairs ")" ;
request     = "(" "request" kvpairs ")" ;
measure     = "(" "measure" kvpairs ")" ;
compare     = "(" "compare" kvpairs ")" ;
repair      = "(" "repair" kvpairs ")" ;
fail        = "(" "fail" kvpairs ")" ;

kvpairs     = { kvpair } ;
kvpair      = ":" key value ;
value       = symbol | number | string | sexpr | dict ;
sexpr       = "(" symbol { kvpair } ")" ;
dict        = "{" { key value } "}" ;

time        = "{" "tense" tense [ "at" timespec | "interval" "{" "star
tense       = "past" | "present" | "future" ;
aspect      = "progressive" | "perfect" ;
```

```
mod-type    = "epistemic" | "deontic" | "ability" ;
mod-val     = "must" | "may" | "can" | "likely" | "possible" | "necess

role        = "agent" | "theme" | "patient" | "recipient" | "source" |
evid        = "obs" | "sensor" | "report" | "model" | "rule" ;
relation    = "equal" | "greater" | "less" ;

symbol      = /[A-Za-z_][A-Za-z0-9_-]*/ ;
uuid        = /[A-Fa-f0-9-]{8,}/ ;
iso8601     = /[0-9TZ:+-]{10,}/ ;
key         = symbol ;
token       = /[^\s]+/ ;
number      = /-?[0-9]+(\.[0-9]+)?/ ;
string      = /"[^"]*"/ ;
JSON Schema (core object)
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "title": "UL Core Statement",
  "type": "object",
  "properties": {
    "event": { "type": "string" },
    "pred":  { "type": "string" },
    "args":  { "type": "object", "additionalProperties": { "type": ["s
    "time":  { "type": "object" },
    "mod":   { "type": "object" },
    "neg":   { "type": "boolean" },
    "evid":  { "type": "string", "enum": ["obs","sensor","report","mod
    "conf":  { "type": "number", "minimum": 0.0, "maximum": 1.0 },
    "unit":  { "type": "string" },
    "notes": { "type": "string" },
    "link":  { "type": ["string","null"] }
  },
  "required": ["event","pred","args"],
  "additionalProperties": true
}
```

Appendix B. Quick-Reference Cheat Sheet Statement Types - Assertion: (assert . . .) - Request: (request . . .) - Measurement: (measure . . .) - Comparison: (compare . . .) - Repair: (repair . . .) - Failure: (fail . . .) Core Atoms: ENT, REL, TIME, MOD, NEG, EVID, CONF, UNIT, REF, LINK, SCOPE. Core Roles: agent, theme/patient, recipient, source, goal, location, instrument, beneficiary, cause, experiencer. Modality Types: epistemic, deontic, ability. Evidentiality: obs, sensor, report, model, rule. Time: past, present, future; absolute ISO timestamps or intervals. Negation: :neg true with explicit :scope when needed. Appendix C. Worked Examples English UL Gloss “Could you probably send me the latest report by tomorrow? It’s around 5MB.”

```
(request
  :event r12
  :pred send
  :args { agent (REF you) theme (REF report-latest) recipient (REF me)
  :time { tense future at 2025-09-29T00:00:00Z }
  :mod  { type deontic val request }
  :meta { approx_size 5MB }
  :evid report
  :conf 0.80
)
Gloss: send{agent:you, theme:report#latest, to:me}; TIME[future@2025-0
"Alice is taller than Bob."
(compare
  :event c1
  :pred height
  :args { theme1 (REF alice) theme2 (REF bob) }
  :relation greater
  :conf 0.9
)
```

Appendix D. Mini-Dictionary Predicates (sample) be, equal, exist, have, own, give, do, move, make, see, hear, say, ask, measure, compare, cause, allow, require, located_in, part_of, cut, rain, leave, on. Roles agent, theme/patient, recipient, source, goal, location, instrument, beneficiary, cause, experiencer. Modality Values must, may, can, likely, possible, forbidden, certain, unlikely. Evidentiality obs, sensor, report, model, rule. Units m, kg, s, C, J, W, m, m, m/s, USD, EUR, mg/dL, mmHg, MB, GB, ms. Appendix E. Self-Study Drills (with Selected Answers) Beginner - The lamp is on. - Please close the door. - The weight is 10kg. Intermediate - Alice might be in Paris. - Not all cats like milk. - Send the report by tomorrow. Advanced - If Bob doesn't study, he will fail. - Every researcher who wrote a paper received funding. - It will probably snow next week, according to the forecast. Selected Answers See Chapter 13.4.