Overview of The LC Logical Typing System

The LC Logical Typing System attempts to account for all logical/semantic aspects of meaningful expressions. This includes

- A language/model for representing anything that can be meant (by an expression) including so-called empirical facts, information, knowledge, patterns, rules, abstract (e.g., logical and mathematical) relationships and estimates of believability/confidence in a unified and consistent fashion
- Rules for creating, compiling and managing truth within a system of expressions, including, on the compilation side, criteria for identifying and processing various kinds of meaningless/ill-formed expressions (independent of their surface grammatical or syntactic form).

It does this by providing a novel account of

- **Types** as the basic building blocks for both schemas and expressions. In contrast with traditional typing systems that focus on definitional reasoning (e.g., deductive theorem proving in logic and units checking in computer science[1]), LC Types provide a unified treatment for empirical data, empirical patterns and abstract/definitional rules[2].
  - The popular notions of number system, dimension, hierarchy, measure, attribute, variable, data type, network, directed graph, (subjects and predicates), and (functions and arguments), may be thought of as specializations of the more general notion of Type.

- **Schemas** (built from types, and) which serve as the mechanism for capturing empirical data, empirical patterns and abstract relationships. LC Schemas are responsible for sending, receiving, calculating and storing all kinds of expressions (e.g., queries, assertions and commands) as well as the sources and believabilities of those expressions. They include rules for assessing the meaningfulness of any

---

[1] see for example http://plato.stanford.edu/entries/type-theory/ , http://plato.stanford.edu/entries/type-theory-church/ , http://en.wikipedia.org/wiki/System_F

[2] Towards that end, they provide

- Type definitions that contain both a logical specification and one or more physical representations,
  - And, the ability for any type to serve as an independent or a dependent variable.
- An explicit language for controlling the type's topology which in turn alters the atomic operators that a type might support.
  - Thus providing a unified treatment for hierarchies, multi-unit systems and compound structures.

expression and a unified approach to logical inferencing that has been shown[3] to cover (and extend) both the propositional and the predicate calculus[4].

- o The popular notions of model, world (model), multidimensional hypercube, multi-cube, Relation, Class diagram, frame, script, system of equations, shape file, process, application and program may be thought of as specializations of the more general notion of Schema.

Within the LC Typing System, all behaviors are carried out in terms of expressions. For example, expressions

- Query for and test other expressions
- Specify and execute calculations
- Modify and create new schemas
- Add to or query data within schemas
- Query any aspect of a type
- Modify or Create new Types

[3] See
A Functional Basis for Tractarian Number Theory
presented at the Wittgenstein Symposium on the Foundations of Mathematics
Austria 1992
published in Wittgenstein's Philosophy of Mathematics
Verlag Holder-Pichler-Tempsky Wien 1993

A Tractarian Approach to Information Modeling
presented at the Wittgenstein Symposium on the Foundations of Language
Austria 1989
published in Wittgenstein-Towards a Re-evaluation
Verlag Holder-Pichler-Tempsky Wien 1990

[4] Unlike traditional information modeling theories (such as any flavor of Relational or OO modeling)

- Where intra- type/domain/class operators and inter- type/domain/class operators are distinct,
    - The building of structures/schemas out of LC Types uses the same algebra as the building of structures within LC Types.
        - o This makes it easier to link different models.
- Where inferencing is based on the predicate calculus which is known to suffer from both inconsistencies and semantic gaps (e.g., dealing with hierarchies and time)
    - LC schemas provide a mechanism for determining the well formedness for exchanged and for compiling expressions.
        - o This provides for better inferencing in the presence of spotty/irregular data and for efficient communication strategies that take context into account.
- Where the definitions of schemas, and of the types out of which said schemas are composed, are static relative to the data that enters the system of schemas as schema instances
    - ▪ Both LC Type and Schema definitions may be altered as a result of processing new data (schema instances)

Higher level concepts like believability and logical state (e.g., notions of missing and meaningless) are defined in terms of second order expressions (i.e., expressions that take other expressions as arguments).

Higher level competencies (e.g., cognitive processes) are defined in terms of systems of schemas wherein any schema(s) may exist in any kind of cardinality relation with other schemas (including M-to-N), and wherein expressions in some schemas (input or derived) may query, edit, activate or deactivate other schemas.