TYPES IN TRANSPARENT INTENSIONAL LOGIC AND EASEL — A COMPARISON

Aleš Horák
Faculty of Informatics, Masaryk University Brno
Botanická 68a, 602 00 Brno, Czech Republic
E-mail: hales@fi.muni.cz

ABSTRACT
This article describes the extended type hierarchy of the Transparent Intensional Logic (TIL) as a higher order logic theory. We also present the basic ideas of TIL constructions as being a suitable meaning representation for natural language.

A comparison of the purely logically oriented type system of TIL with the property-based types of the Easel language is discussed in the text with the orientation to the possibility of applications combining the two approaches.

KEY WORDS
TIL, intensional logic, Easel, types

1 Introduction

Knowledge representation and reasoning systems usually aim at applications where the speed of responses to questions force using an underlying formalism with limited expressive power. With this approach it is feasible to describe a selected field of interest with a possibly huge amount of facts and reuse this *knowledge* in an expert system tool.

Another approach, with a very small number of real implementations, is based on the fact that most of the human knowledge is encoded in the form of natural (non-artificial) language. Thus a straightforward way to handle such information is to build a system capable of analyzing the sentences directly with a machine parseable output and a connection to an automatic inference machine.

A system that fulfils such requirements is based on Tichý's Transparent Intensional Logic (TIL, [1]). The work [2] describes the logical system based on the Normal Translation Algorithm (NTA) for TIL. It follows the approach of full natural language analysis in the form of *normalized TIL constructions* (concept) as a meaning bearer for NL expressions and assertions.

2 Transparent Intensional Logic

When seeking for the elucidation of meaning, Tichý thoroughfully examined the most promising theories and, since all of them suffered from unacceptable inconsistencies in various places, he has introduced TIL with the fundamental conception of construction as a possible meaning naming tool.

TIL has come out of the ideas of Gottlob Frege, mostly following his fundamental predicament about meaning, known as the Compositionality Principle: the meaning of a compound is a *function* of the meanings of its constituents.

One of the main assets, TIL brings to NL analysis, is its intensionalistic feature, which is enabled through the possibility of denoting objects of complex type, i.e. not only objects of the universe (individuals), but also functions of individuals, functions of functions of individuals, etc.

The mechanism of intensions and extensions can be displayed with the following example of two expressions: *the rector of Masaryk University* and *Jiří Zlatuška*. The latter expression is a label of an individual, which is independent on the state of the world and on the time moment. In the case of the first expression the situation is different. The dependency of this expression on time moment is clear: the rector of Masaryk University was not the same in every time moment of the history. It was, for example, also *Eduard Schmidt* (until 1998). The fact that *Jiří Zlatuška* is now the rector of Masaryk University is only one of many possibilities, that could have happened. Objects depending on the state of the world and on time moment are called *intensions*, other objects are *extensions*.

Non-intensionalistic logic systems, like first order predicate logic, which is sometimes also used to represent real world facts, have no means to work with intensions.

Another powerful feature of TIL is the extended type hierarchy — TIL works with objects of higher order type. That means we can represent belief sentences in the same way as any other object and that the representations do not lead to any inconsistencies in further processing. A good exemplification is a statement about numerical calculation: if we say *Paul counts the sinus of* $\pi/2$, a first order type representation would make the expression *the sinus of* $\pi/2$ indistinguishable from the number 1. However, this obviously does not reflect the real meaning of the sentence — Paul does not really need to know what is the correct result of *the sinus of* $\pi/2$ and thus he may not even think about *any* particular number here.

2.1 TIL Types

The type system of TIL was based on the theory of types introduced by Church and was then further extended by Tichý [3]. In TIL, every object (as a representant of an entity described by the analyzed natural language expression) has its *type* which is defined over a firmly set type base.

Every object (which is not a construction) is assigned either one of the *basic types*, or a type that is formed by a *mapping* from one type to another type. Within this framework, we can obtain an infinitely nested hierarchy of types, i.e. mappings of basic types, mappings of mappings, mappings of mappings of mappings, etc. Nevertheless, how difficult soever the mapping is, the object of the respective type is still "flat". The flatness of mappings is predicated upon the way mappings are treated — as collections of $(n+1)$-tuples (in case of an n-adic mapping). This means that the mapping is (virtually) represented by a table of values without any possibility to find out the way (a procedure) which leads to those values. This is also one of the reasons why mere mappings cannot serve as surrogates for meaning — mappings lack a constructing *structure*.

For capturing this structural property, we do not work directly with TIL objects. We rather reference them by the schema describing, how (in correspondence with the NL expression) the TIL object is *constructed*.

A construction is usually displayed in the typed λ-calculus notation, as a λ-term, but the real construction is still the procedure described with the term, not the term itself.

Variables are the only simple constructions. There are three modes of forming constructions (from non-constructions and other constructions): trivialization, composition and closure. Classes of all constructions form another basic types in the hierarchy.

The idea of logical analysis of NL with TIL lies in the presupposition that every language has a definite *intensional base* — a collection of fundamental properties of objects (not only $(o\iota)_{\tau\omega}$but any world and time depending relations among objects like colors, heights, attitudes, ...), that are capable (without any need for other extra techniques) of describing a (thinkable) state of the world.

In TIL, such an intensional base of a NL is rigorously explicated in an *epistemic framework*, i.e. a typed system based on a set of four basic types of order 1 (simple types) and collections of all constructions of order n as types of order $n+1$ (higher-order type).

The four simple types are called a type *base* and are denoted with letters o, ι, τ and ω, with the following interpretations:

o is a set of two items representing the *truth-values* True (T) and False (F). These two objects behave exactly the same as their counterparts in the standard predicate logic especially in combination with standard logic operations such as conjunction, disjunction, implication or negation. These predicate logic operations can be represented as objects of type (oo) or (ooo), i.e. functions with one or two arguments of type o returning a value of type o.

ι is a class of *individuals*. The designation "individual" must not entice us to imagine the members of this class as beings with all their properties. In TIL, the notion of an individual is best interpreted as a mean of a numerical identification of a (type unstructured[1]) entity. Any individual properties are ascribed to an object of type ι only by means of asserting a statement that contains the ascribing as its part — in the proposition 'John is a fat man,' we use the ι-object John[2] as an identification of an entity that is ascribed the property being a fat man (FatMan)

$$\lambda w \lambda t[\mathbf{FatMan}_{wt}\mathbf{John}]$$

The individual John itself is carrying no *a priori* properties, it serves as an identifier of a further unspecified object and is mainly used for references to this object.

τ is a class of *time moments*. Due to the continuity feature of this class, it may be regarded identical with the class of real numbers, in case we specify a fixed zero point and a unit. Functions working with arguments of type τ are usually used for expressing the *temporal dependency* of an entity.

ω is a class of *possible worlds*. Its members are intended for a transparent representation of *modal dependency* of described objects.

A possible world, in accordance with the Leibnizian claims, is in TIL defined as a *determination system* that, for each of the intuitively, pre-theoretically given features from intensional base, contains an assignment of all possible (consistent) distributions of those features.

Thus the selection (anchoring) of one such possible world allows us to work with values that are dependent on the *actual world* without the need of knowing exactly what the actual world is. This is necessary, since the knowledge of the actual world itself would bring severe inconsistencies into any theory that would rely upon it.

The non-basic types are classes of special entities build upon this base. They form an infinite hierarchy. The classes are formed by mappings from and into the initial objects, mappings from and into those mappings, etc. For every type a countable set is given, whose members are called *variables*.

The types can be also viewed as classes of TIL objects, i.e., if ξ is a type, $A \in \xi$ then A is called a ξ-object.

[1] i.e. whose type is not decomposable as a function to the types of the functional arguments and the type of its return value.

[2] we take John here (and on other places in the text) as an example of an individual despite the fact that proper names often need to be analyzed as a pragmatically anchored expression.

Type	Notation	Object
$((o\tau)\omega)$	$o_{\tau\omega}$, π	a *proposition* — an assertion whose truth-value depends on the world and time. **Example:** 'John is old.'
$((\iota\tau)\omega)$	$\iota_{\tau\omega}$	an *intensional role* — an individual office that may be engaged by different individuals in different worlds or times. **Example:** 'the rector of Masaryk University'
$(((o\iota)\tau)\omega)$	$(o\iota)_{\tau\omega}$	a *property* — an atomic intensional feature that an individual either has or has not according to a chosen world and time. Such feature is then represented by a class (its characteristic function) of those individuals that have the feature in a certain world and time moment. **Example:** 'being blue'
$(\xi\tau)$	ξ_{τ}	a ξ*-chronology* — a mapping that specifies the flow of changes of a ξ-object in time. **Example:** 'yesterday' is a time interval of 24 hours that ends at the last midnight, thus it represents a different interval every day.
$((\xi\tau)\omega)$	$\xi_{\tau\omega}$	an *intension* — expresses the dependency of the related ξ-object on the selected possible world and time moment. The application of an intension to the world and time (i.e. composition of a $\xi_{\tau\omega}$-object with argument of type ω and then with argument of type τ resulting in a construction of a ξ-object) is called the *intensional descent*. If ξ is not itself an intension, it is called an *extension* and represents an entity whose value does not change with world and time. All mathematical objects (numbers, operations, axioms) correspond to extensions.

Table 1. The most frequent derived types in TIL (ξ in the table stands for any type).

In TIL, we often consider particular types. The most frequent derived types are described in the Table 1.

3 Types in Easel

Easel [4] is a simulation system designed for gathering, processing and display of information about various active entities of the simulated world, about their interactions, and about their collective global effects.

For the description of the entities in the simulated processes, Easel uses a property-based type system. In the system a *type* provides a (complete) description of a template of an object. Any entity is then an *instantiation* of such template. The type can also be thought of as a class of all objects that comply with the set of properties stated in the type template.

Together with inheritance of type properties and arrangement of types in one hierarchy (with the root type *all*), Easel language resembles common object-oriented programing languages and formalism. However, as an extension to the usual OO classes, Easel emphasizes the need for type manipulations with symbolic techniques. The definitions of subtypes are described with adjective modifiers similarly as in NL expressions:

```
example( ): action is
    fruit: type;
    round(any): type;
    green(any): type;
    apple: type is round fruit;
    golden_delicious: type is
        green apple;
    my_apple: golden_delicious;

    confirm my_apple isa fruit;
example();
```

4 Using TIL and Easel in Applications

The specifications of Easel types is suitable for limited simulations with strictly specified bounds of the simulated world. However, in case of complicated descriptions of such extent that reaches the complexness of the real-world situations, the descriptiveness of the types in Easel would run against difficulties due to their underspecification.

The possible combination of both, TIL types and Easel types, in one application seems to be a promising completion of the type specification techniques. TIL allows to place no limits to the expressiveness of the descriptive language used to describe the types and subtypes used in the simulation and (possibly) can make use of any properties from the intensional base of natural language.

The confirmative claims that are currently allowed in the Easel language, would not need to be closed to browsing the type hierarchy, but, with the use of the inference mechanism in TIL, it could analyze any consistent proposition about the type. The above stated example could look as follows:

```
example( ):  action is
    # the primary properties
    Fruit is a class of individuals.
    To be round is a property of
       individuals.
    To be green is a property of
       individuals.
    # the definitions and instantiations
    Every apple is a round fruit.
    golden_delicious is a green apple.
    The variety of my_apple is
       golden_delicious;
    # statements
    If I eat a fruit, I am healthy.

    # claims to be confirmed
    confirm that my_apple is a fruit;
    confirm that if I eat my_apple,
       I am healthy;
example();
```

The form of the description of the situation, i.e. the type definitions, can be written in natural language statements about the discussed elements and their properties. An automatic parsing system can translate the specifications into the following objects and propositions:

```
example( ):  action is
    # the primary properties
    fruit ... (oι)_τω
    round ... (oι)_τω
    green ... (oι)_τω
    # the definitions and instantiations
    (∀x)(∀w)(∀t)apple_wt x ⊃ (fruit_wt x ∧ round_wt x)
    (∀x)(∀w)(∀t)golden_delicious_wt x ⊃
        (apple_wt x ∧ green_wt x)
    (∃my_apple)golden_delicious_{w1 t1} my_apple
    # statements
    (∀w)(∀t)(∀x)[(fruit_wt x ∧ Does_wt I [Perf_w[eat x]_w])
        ⊃ healthy_wt I]
    # claims to be confirmed
    confirm (find a match)...
        x : fruit_{w1 t1} my_apple
    confirm (find a match)...
        x : Does_{w1 t1} I [Perf_{w1}[eat my_apple]_{w1}])
            ⊃ healthy_{w1 t1} I
example();
```

In the case of argumentations about instances of a type, like my_apple here, we use the current reference time t_1 and the actual reference world w_1, which are related to the working environment of the processing system.

The mechanism of analyzing questions and inferring derivations from other facts is described in more detail e.g. in [3, 2]. Finding a *match* means a request to infer (from the underlying knowledge base) the value x of the specified TIL type, i.e. the truth-value o of the extensified proposition on the right hand side of the match, here.

Even if the above stated examples do not use more power, than a temporal first order logic system would provide, the TIL theory ensures correct analysis of much more complicated expressions, like *intensional roles* or *belief sentences*.

A straightforward reuse of the TIL analysis and TIL inference system is, however, not so easy as a simple incorporation of some other, fully described algorithm. TIL analysis needs to work with something as complex as the natural language and the ambitions of TIL do not allow to make any simplifications of the matter.

Pavel Tichý published his views on the language analysis in TIL in several papers (e.g. [5]) and started to write a thorough work on that matter [6]. However, he managed to write only the first out of the intended twelve chapters, and thus has left a lot of particular phenomena of NL without the prescription of their proper analysis. The first text, that tries to describe the algorithm of translation of NL sentence in its completeness, is the cited work [2]. Nevertheless, the algorithm is still in its first version only and its finalization is the work of Tichý's followers such as Materna, Oddie, Duží, Horák and others.

5 Conclusion

In the paper, we have introduced the main ideas and basic structures of Transparent Intensional Logic. We have also argued for the TIL constructions being the meaning bearer for any natural language expressions.

As a complement to the extended type system of TIL, we have chosen the Easel language with its property-based types. We have formulated the assets of using the two approaches in one application and showed the need for working implementations of TIL NL analyzer and TIL inference system based on the work of TIL's author and other researches.

Acknowledgements

This work was supported by Ministry of Education of the Czech Republic Research Intent CEZ:J07/98:143300003.

References

[1] P. Tichý. *The Foundations of Frege's Logic*. de Gruyter, Berlin, New York, 1988.

[2] A. Horák. *The Normal Translation Algorithm in Transparent Intensional Logic for Czech*. PhD thesis, Masaryk University, Brno, 2002.

[3] P. Tichý. Foundations of partial type theory. *Reports on Mathematical Logic*, 14:52–72, 1982.

[4] D. A. Fisher. Design and implementation of EASEL. In *Proceedings of MacHack 14, the 14th Annual Con-*

ference for Leading Edge Developers, Deeerborn, MI, 1999.

[5] P. Tichý. Cracking the natural language code. *From the Logical Point of View*, III(2):6–19, 1994.

[6] P. Tichý. The analysis of natural language. *From the Logical Point of View*, III, 2:42–80, 1994.