iCTRL: Intensional conformal text representation language

Gábor Rédey

Hungarian Atomic Energy Authority, Nuclear Safety Directorate, P.O. Box 676, H-1539 Budapest 114, Hungary

Received 19 February 1997; received in revised form 13 March 1998

Abstract

A new compact and homogeneous symbolism is introduced to achieve a more general and exact representation of natural language texts. Traditional first-order and intensional logic cannot cope with numerous natural language phenomena such as the large variety of modalities, satisfactory interpretation of iterative application of modal operators or certain modelling problems like one-to-one sentence–formula mapping. The CTRL/iCTRL formalism can model them successfully and they are able to control many other different shades of meaning by applying only a minimal number of syntactic tools.

The most profitable and beneficial AI application of the presented natural language syntax consistent knowledge representation technique is automated knowledge acquisition: computer-aided textual data base generation and logical inference based information retrieval. CTRL/iCTRL applicability is demonstrated by various illustrative examples including a transparent graphical interpretation analogous to Frege’s graph language that help clarify new concepts and exemplify partial inappropriateness of traditional logical language.

The CTRL/iCTRL paradigm is based on a novel and interesting synthesis of the two traditional logic schools, the Stoic and the Peripatetic school, refuting a century long scientific prejudice against the latter stated to be completely outworn. An interesting issue of this analysis points out that expressing subordination unconsciously and simply by co-ordination causes a typical restriction of meaning in classical logic. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: Intensional logic; Aristotelian term logic; Sentence–formula proximity; Knowledge base validation; Natural language syntax conform text modelling; Computer-aided knowledge acquisition; Content relevant textual knowledge base query handling; Information retrieval systems

1 Email: redeyg@helka.iif.hu.

0004-3702/99 - see front matter © 1999 Elsevier Science B.V. All rights reserved.
PII: S0004-3702(99)00016-8
1. Introduction

A new logical language, CTRL, recalling Aristotelian term logic has been introduced formerly by the author [12]. The aim of this essay is to extend it to the intensional level, iCTRL, which faithfully and consistently formalises the classical Carnapian concept of intension also reconstructing Kripke semantics in order to arrive at a better expressive power. This latter characteristic of iCTRL implies a powerful logical type of knowledge representation method enabling text based information retrieval system applications, on the other hand it is also promising in modelling several particular natural language phenomena.

A detailed demonstrative analysis of classical S4 axioms, and the converse of the Barcan formula illustrates some of these capabilities of the new representation language revealing blind spots of the traditional intensional language of logic, which hinder it from accessing to certain distinctions and certain shades of meaning.

Additional conclusions are reached by clarifying and comparing the historical back-grounds of the competing formal structures of the conventional (Stoic school related) and text representational (continuing the Peripatetic line of tradition) language of logic that explain the causes of the differences in their expressive capacities. The presented arguments justify the fact that the Peripatetic approach to logic has unfairly been neglected since the end of last century.

The paper itself is confined to the indicated subject supposing the reader to be familiar with CTRL. Nevertheless, most of the subsequent sections on Orientation together with Section 3, that is a complete description of the new formalism imported from [12], offer a sufficient overview of ordinary CTRL, its basic concepts, techniques and results necessary to get to further conclusions. Appendix A, however, is a technical supplement to the proofs of Section 5 summarising definitions, theorems and their proofs essential to come to a correct adaptation of resolution to iCTRL, the most commonly known inference algorithm of theorem-proving systems.

2. Orientation

2.1. CTRL concepts—a retrospection

2.1.1. Motivation

The ordinary conformal text representation technique has been developed with almost fully pragmatic deliberations to achieve a suitably accurate textual knowledge representation method that allows logical type of information retrieval. Accuracy is particularly accentuated here and it is meant in both syntactic and semantic sense. A traditional logic based text analysis and representation method could result in a routine and correct solution with maximal accuracy concerning semantics, but not in the respect of syntax. The traditional language of logic generally needs a necessary preparation phase, i.e., a truth-value-invariant mapping of natural language statements, usually implying drastic changes

---

2 The qualitative terms of expressive power, capacity, capability or expressiveness all refer here to the attribute of a formal language relative to a corresponding natural one indicating how much of the latter can be emulated by that formal language.
in syntactic structures, before converting them to the corresponding first-order formulae. Therefore, as the latter results reflect the syntactic structures of the original statements in a relatively poor manner, this can make the essential validation tests, mapping fidelity tests, i.e., contrasting the source text with its formalised version, actually unrealisable on a large scale application. Text representation was designed to eliminate these problems by transforming natural language statements to formulae, which can be constructed solely on the basis of their elementary grammatical analysis relying on the principle that syntactic structures of natural languages should totally involve logical ones.

Traditional logic distinguishes logical rough structure of statements from that of logical fine structure according to the level of representation applying the tools of propositional and predicate logic, respectively. Keeping this terminology but not the decomposition principle unchanged, the main difference when comparing the devices of traditional logic to the method in question is that using propositional connectives for interpreting a subordinate type of relations is regarded as unfavourable for encoding logical fine structure.

Example 2.1. According to the routine representation technique the sentence “Horses are animals.” needs to be rephrased first like “For anything, if it is a horse, then it is an animal.” from which finally the following classical first-order formula can be obtained:

\[ \forall x. \text{is-horse } x \supset \text{is-animal } x. \]

This rephrasing stage itself and hence the import of the connective “\(\supset\)” is objectionable here, as the natural language statement does not use any co-ordinate (propositional) connective in this context either. A somewhat different problem is that actually there exists a rewriting of this statement which really uses the criticised conditional, and which is usually accepted as (but formally never proven to be) its truth value equivalent.

A rather much natural grammatical principle is regarded adequate instead. Pais referring to Paul builds the natural grammatical syntax on one single relation: the generalised subject–predicate relation [9]:

“The ground of the most ancient logical relation, or rather of any logical relation, is that a concept is compared to other ones, and as a consequence of this it is unified with certain ones and distinguished from others. This is the so-called subject-predicate: predicative relation, that is the most original structure in grammar carrying that kind of relationship that has a dominant role in the dynamics which makes a sentence to be a sentence. It is the utterance of a name of a concept after or before another one to express that one of them is similar to the other, moreover, according to certain other ways of comprehension, it is identical to that. . . .”

“Several other relations have grown out from the subject-predicate, i.e., predicative relation: attributive, determinative (adverbial), objective relations, in such a way that the attributive, determinative and objective structures are power lost subject-predicate: predicative structures. . . .”

\[ \text{is-animal} \]

3 The following extracts are translated from Hungarian by the author.
“We try to make an effort to clarify the development process of grammatical relation categories subsequently. . . . Paul’s fundamental idea will be accepted to set out from, according to which any syntactical relations: complement forming, other than copulative conjunction: co-ordination, are derivable from the subject-predicate relation in such a way that the relation of subject and predicate is applied repeatedly.

The following development possibilities can be pointed out.

Sentences can be constructed such as water warm, flow—one subject, two predicates. One of the two predicates, warm, has lost its predicative, being-in-relation-to-the-reality accent, and joined to the subject so as to create, instead of a predicative relation, which is essential in the sentence forming dynamics, another type of that, i.e., an attributive one. But warm could lose its predicative power by joining to the other predicate, flow, as becoming to its adverb, determiner, with the meaning ‘warmly flows’. Predicates of certain character have drawn other predicative power lost predicates to themselves as objects, i.e., ‘flow warm’. in such a way that ‘flow’, that has a similar meaning to ‘run’, changed its meaning to ‘make something flow’ or ‘make something run’. . . .”

According to the ideas outlined above the following principles can be settled:
– there exists a grammatical model of the natural language sentence structure describing the non-co-ordinate relations among its constituents as predicative, objective, determinative, attributive etc. ones, and
– grammatical constituents corresponding to these relation types are not distinct in logical sense: the roles of these constituents can be interchanged (what was object can be a subject later etc.), hence all other differences can be neglected, in other words the functional sentence model can be based on one single generalised subject-predicate relation.

The (generalised) predicative relation can easily be recognised in the previous example by identifying “(any) horses” as the subject and “are animals” as the predicate. Another subject-predicate (determinative in the strict grammatical sense) relational decomposition could be, of course, continued with the subject.

Two elementary components, predicates in traditional terms, of the set

\[ “horse x”, \quad “\forall x”, \quad “animal x” \]

are joined in predicative relation forming the formula subject: \((\forall x)(horse x)\), which group forms again a predicative relation with “animal x”, the formula/sentence predicate, resulting in the formula which can be represented as follows:

\[ (animal x)[(\forall x)(horse x)] \]

where \( x \) denotes a variable parameter referring to the objects of the given context and \((\ldots\ldots)\) denotes uniformly the predicative relation. Punctuation by spaces and commas may be preferred to brackets:

\[ animal x(\forall x, horse x), \]

\(^4\) The Hungarian original equivalent to flow is intransitive, too.
which can be shortened, keeping the right-to-left priority convention, to:

\[
\text{animal } x, \forall x, \text{ horse } x.
\]

The above CTRL formula of list form can intuitively be visualised by the graph in Fig. 1.

As comparing this one to the first-order formula above it is obvious that only the same main components are permuted, although their grounds are totally different. At this stage an apparent advantage of the graphical representation technique is also conspicuous: orthogonal branches represent the two main independent dimensions (elementary relations of the syntax).

Coming to the end of the informal introduction some new terms are to be initiated next to promote to a formal description.

2.1.2. Elementary relations of text representation

*Conceptual relation* plays the same role as the concept of predicate in traditional logic (in order to distinguish it clearly from the predicative relation introduced and discussed above, this new term is preferred). Conceptual relation is an ordered list of symbols set up by the name of the concept heading the list, and continued by a parameter list indexing the potential complements or subordinates of the concept. The so-called variable symbols following the concept name in the list identify the grammatical roles of the complements by their position characterising the conceptual relation: e.g., “\text{horse } x” (\(x\) is-a-horse), “\text{has } x\ y” (\(x\) has \(y\)).

*Predicative relation, sentence-forming relation.* Conceptual relations, which are the word-like elements of the language, can construct compound expressions (predicates, subjects, clauses, and sentences) making use of the predicative relation that may be applied iteratively. Two (or more, by using right-to-left priority convention of bracketing) conceptual relations form a predicative one, provided that there is a group of objects commonly referred to by them: they have a common variable parameter. The head element of the list of conceptual relations can be regarded as the “modifier section” of the rest, i.e., the group of objects referred to by the tail of the list, acting as the “modified section”.

**Example 2.2.**

- “\text{\underline{high} } x, \text{\underline{horse} } x”—\(x\) is-high, where \(x\) is-a-horse \(\equiv x\) is-a-high-horse—form together a potential compound predicate (the modifier-modified pairs of phrases are enhanced).
- “\(\forall x, \text{\underline{high} } x, \text{\underline{horse} } x\)”—any-of \(x\), where \(x\) is-high, and where \(x\) is-a-horse \(\equiv x\) any-of-high-horses—form together a potential subject.
• “animal $x$, $\forall x$, high $x$, horse $x$.”—$x$ is-animal, where $x$ is-any-high-horse $\equiv x$ any-high-horse, $x$ is-animal—is a sentence.

Contraction variable is a device for cross-sentential and cross-clausal anaphor resolution (also discussed in [3]). This problem does not seem to be crucial, however, to achieve a more natural decomposition of natural language sentences/clauses seems to be an acceptable motivation.

Example 2.3. Consider the following sequence of sentences: “Pedro has a donkey. He beats it.” Traditionally, this two separate sentences cannot be represented by separate first-order formulae. Personal pronouns, “he” and “it”, of the second one are expected to be interpreted first as referring to the terms “Pedro” and “a donkey”, respectively, and then the two sentences are represented by one single formula.

What one would like to have instead is a representation of the separate sentences individually also utilising the above interpretation. The formula representing the first sentence is clearly:

$$(\text{has } x \ y, \exists y, \text{donkey } y)\text{Pedro } x.$$ Subsequently, it is supposed implicitly that the subject and the object are known and can be referred to:

$$(\text{beat } u \ v, \forall u \rightarrow y)\forall u \rightarrow x.$$ Here $v \rightarrow y$ and $u \rightarrow x$ are called contraction variable occurrences of $x$ and $y$ in $\forall u \rightarrow y$ and $\forall u \rightarrow x$, respectively, indicating that the quantification of the subject and the object are predefined as shown earlier. The descriptive part of the subject and the object should be determined by the referred sentence omitting the superfluous internal quantification. An execution of the prescribed substitutions on the second formula yields the following equivalent rephrasing

$$([\text{beat } u \ v, \forall v, (\text{has } x \ v, \text{donkey } v), \text{Pedro } x] \text{Pedro } u) \equiv (\text{beat } u \ v, \forall v, \exists u \ v, \text{donkey } v)\text{Pedro } u.$$ The above formulae can also be illustrated (see Fig. 2).

2.2. iCTRL—an intensional extension of CTRL

2.2.1. Motivation

One of the commonly used intensional functors is necessity, $N$, causing interpretation problems even in the case appearing to be one of the most simple ones. It is not easy to find an equally simple dual formula for de re application of $N$ like the one valid for the de dicto mode: $N(p \ a)$. This kind of ambiguity of de re mode, studied earlier by Stalnaker and Thomason [17], can be eliminated either by using the correct but relatively complicated Churchian $\lambda$-conversion technique: $[\lambda x. N(p \ x)]a$, or by simply paraphrasing as in [14]: $\exists x [x = a \& N(p \ x)]$. 


A similarly equivocal, but even more confusing example demonstrating occasional ambiguity of the language of classical modal logic can be presented by the S4 scheme. Which of the readings of $\Box p$ should be taken correct: $\Box(\Box p)$ or $(\Box(\Box))p$ or none of them, but some other one that may be inexpressible in the classical language? These problems seem to be left out of reach for classical analysis motivating revision.

The succeeding analysis will consider choices of intensional extension of text representation language, leading also to a more exact reformulation and more convincing answer to the open problems summarised above.

2.2.2. Extension, intension, context and related problems

The widely known key concepts of intensional logic (contrasted with extensional one, which is associated with extension) are intension (meaning) and context (situation, alternative/possible worlds). It is a cliché yet a common sense understanding that any logical analysis of a natural language sentence, including the intention of its logical representation by reformulating it into a (or rather into a few) corresponding logical statement(s), needs to know the circumstances of its utterance, i.e., the context. In other words, the logical structure of a statement is impossible to be thoroughly described without having enough knowledge about the current denotation (extension) of the occurring designators, but at the same time information about their usage (context, meaning) may also be crucial. These latter type of relations characterise intensional logic compared to extensional one [14], the historical milestones of which are represented by the names of Frege (1892), Carnap (1947) and Kripke (1959, 1963), parallel with them Prior (1957) and Montague (1970) subsequently.

According to Carnap’s generally accepted terminology the meaning (intension) of any designator is a function, which orders its current extension relative to a situation/possible world. Approaching from the representational point of view the first problem faced is how to represent meaning as an object. As it is most commonly presented due to Montague, the intension of any designating expression, $p$, can be denominated, viz. it is the function, denoted by $^p$, attaching an extension to $p$ relative to an object domain for any elements of an index domain labelling situations, respectively. Hence, intensional functors operate
on the intension of their arguments. Special stress is on the fact that intension, taken to be a function, is referred to not by its value or argument, but by the function itself, while an intensional functor is applied on it.

Beyond other difficulties it is not easy to accept this principle. It was discussed previously with reference to modelling natural language sentence structure and Pais’ fundamental principles [9] that every statement, including intensional ones, too, have a structure which can be represented by simple or iterated subject-predicate relations, where the terms subject and predicate are meant in a generalised sense. Assuming this, intension can play the role of a subject with difficulties.

**Example 2.4.** It is hardly conceivable that the predicate: “Julia looks for”, is to be applied on the intension of its object: “a man”, in the classical example: “Julia looks for a man”. It is usually commented, as Julia may not find anybody, because she can be so pretentious that the very man she is looking for may not even exist, but in spite of this fact the statement itself can still be true. So it is often argued that the predicate “Julia looks for”, possibly not being able to refer to any real man, is not applied to its object, “a man”, but to the meaning of it, namely to a function. Against this reasoning the counter argument can be posed that Julia is apparently supposed to have no doubt about the meaning according to she is looking for “a man”, even if the phrase: “a man”, as projected to the present world, may have an empty extension.

A more favourable base for a more exact solution seems to be the observation that the present world and Julia’s imaginary world containing also that very person she is looking for may not be identical. But if such person really exists in Julia’s imagination, then that extension should be referred to by her search, and so the above sentence can be true in spite of the fact that those people in the present real world may never be found.

The above arguments point out that keeping the commonly used original concept of meaning, a modified way of representing intension may be more preferable for the specified purposes, which makes every component explicit.

The Montaguevian way of notation itself is based on the same Carnapian definition of meaning. According to that it is a function attaching the current extension to any designator in any situation. The meaning of \( p \) is denoted by \( \text{\textit{O}}p \) obeying to the classical function symbol conventions: \( \text{\textit{O}}p \) denotes the value of the function, \( \text{\textit{O}}p \text{\textit{a}} \text{\textit{t}} \text{\textit{s}} \), so \( \text{\textit{O}}p \) may be interpreted as the current extension of \( p \) in a situation unidentified so far, as it is argued for, or as the function itself, just as it is intended to by the traditional notion.

These ideas can fully be realised by the syntax of conformal text representation:

\[
[\alpha]x.
\]

Provided that \( \alpha \) denotes an arbitrary formula and \( x \) a contextual variable parameter, \( [\alpha]x \) designates the function mapping any index of the context domain referred to by \( x \) to the current extension of \( \alpha \). In this notation there appears every constituent of the scenario, namely: \( x \), the contextual variable parameter referring to any situation of the domain, \( [\alpha] \).
labelling the function itself, and the image of the function which is either referred by the unbound variable(s) occurring in \( \alpha \), or by its truth value.

**Example 2.4 (Continued).** Returning to the previous example the following efforts can be made for a better representation of the original content.

\[
[[\text{look-for } x \ y]z, [\text{Julia } x]w][\exists y, \text{man } y]z[[J z]w]0 w.
\]

This formula expresses that:

(a) Beside the present world, indexed by “0 \( w \)”, there exists Julia’s one, “\( J z \)”, also playing the role of an alternative to the present world, referred to by the expression “\([J z]w, 0 w\)” (with the intended literal meaning: “a world, denoted by the function \([J z]w\), the current image world of which is ordered to the world indexed by 0 \( w \) acting as its alternative”, i.e., the current extension of the world “\( J z \)” depends on the current reference of the present world “0 \( w \)”).

(b) In Julia’s world there are some men who are stated to be looked for by Julia (the extension of the individual [Julia \( x \)]\( w \) should of course belong to the present world, although the relations [look-for \( x \ y \)]\( z \) and [man \( y \)]\( z \) have both their references from Julia’s world).

(c) Hence, Julia’s search is implicitly hypothesised to be embedded into a wider world than the present one also comprising the men who may not exist in reality but extending her search on them, according to intuition, the sentence in focus can after all be evaluated true.

According to the graphic language presented previously the formula above can be visualised by the graph in Fig. 3.

### 3. The CTRL/iCTRL formalism

A formal language reconstructing the previously outlined grammatical relations is described in the subsequent sections.
3.1. CTRL syntax and semantics

3.1.1. Syntax

Alphabet. Let $T$ be a set of symbols (parameters) the elements of which are categorised as follows: variable (also contraction/reference variable) symbols denoted by lower cases indexed if necessary, $n$-ary conceptual relation symbols denoted also by lower cases, names and function symbols denoted by upper cases. “∀” and “∃” are regarded as special unary conceptual relation parameters referring to universal and existential quantifiers, respectively. “~” indicates the logical constant of negation. “ ” (space) and “,” are used as the connectives of conceptual and predicative relations, “,” “(“ and ”)” as punctuation symbols, respectively.

Rules of formula construction. If $p$ is an $n$-ary conceptual relation parameter and $x_1, \ldots, x_n$ are variable symbols then the conceptual relation

$$ p \ x_1 \ldots x_n $$

is an atomic formula. Names formed by a name parameter, $P$, and a variable symbol, $x$,

$$ P \ x $$

are taken as singular conceptual relations. Similarly, provided that $F$ is an $n$-ary ($n \geq 1$) function symbol with $x_1, \ldots, x_n$ as domain and $y$ as image variable symbols, the function

$$ (F \ y) x_1 \ldots x_n $$

is also termed as a singular conceptual relation with respect to the variable symbol $y$ while for domain variables $x_1, \ldots, x_n$ it is still treated as any other atomic formula.

If $\alpha$ is atomic, then it is also a formula. Provided that $\alpha$ is a formula and $\alpha(x)$, $\beta(x)$ denote arbitrary formulae containing neither quantifier nor singular conceptual relations with respect to the variable symbol $x$, so are formulae

$$ \sim \alpha, \quad \alpha(x), \quad \beta(x). $$

Expressions like

$$ \exists x, \quad \alpha(x), \quad \forall x, \quad \alpha(x) $$

including singular atomic formulae are called subjects. If $\sigma(x)$ denotes a subject and $\alpha(x)$ a non-subject, i.e., predicate type of formula regarding the variable symbol $x$, then

$$ \alpha(x), \quad \sigma(x). $$

is also a formula.
This language can be extended by a special type of variable, the so-called reference or contraction variable denoted by “\( x \rightarrow y \)” (see [10–12]). If \( \sigma(y) \) denotes a subject, then

\[
\alpha(x), \forall x \rightarrow y
\]
\[
\alpha(x), \exists x \rightarrow y
\]

are also formulae.

No other type of formula exists.

The concept of free/bound variable appeared in the preceding paragraphs implicitly. An occurrence of a variable symbol, \( x \), in \( \alpha(x) \) is said to be free, if there is no subject binding \( x \) in \( \alpha(x) \), the variable occurrence is bound otherwise. A formula with no free variable occurrences is closed, open otherwise.

Before proceeding with semantics, an example may help to become more familiar with formula construction and discussing variable scopes.

**Example 3.1.** Verify that the expression “\( a x . b x \); \( \forall x . c x \)\)” is a formula and if it really is, which are its subformulae.

According to the construction rules above the following formula decomposition can be realised “\( a x . b x \); \( \forall x . c x \)\)” showing clearly that variable \( x \) of “\( a x \)” cannot refer to the same object as the one in the subformula “\( b x . \forall x . c x \)”, since \( x \) is already bound in it, i.e., variable \( x \) of “\( a x \)” and that of “\( b x . \forall x . c x \)” can actually refer just to different objects. Therefore it would be clearer to choose two distinct variable parameters in this case, e.g., “\( a y . b x . \forall x . c x \)”. Consequently, the scope of a variable symbol is at most that closed subformula which includes it. \(^5\)

This situation changes radically if parentheses were neglected: “\( a x . b x . \forall x . c x \)\)”. This expression cannot be interpreted as a formula, unless it is made unambiguous by suitable parentheses, either just as above or possibly as: “\( (a x . b x) \forall x . c x \)\)”.

### 3.1.2. Semantics

Turning to the semantics, which is always the question of central interest, first an interpretation is built up to assign meaning to any conceptual relational parameters.

**Interpretation.** Let \( U \) be a given non-empty set that is the object universe. Then an extension is joined to each \( n \)-ary conceptual relation parameter, \( p \in T \) \((n = 1, 2, \ldots)\), as its reference:

\[
r(p . x_1 \ldots x_n) \subseteq U^n.
\]

In case of names, \( P \in T \):

\[
r(P . x) \subseteq U.
\]

(Reasonably, \( r(P . x) \) consists of one single element.) If \( F \) is an \( n \)-ary function parameter of \( T \) \((n \geq 1)\) then

\(^5\) From outside of this subformula only a reference variable is able to access to a variable of this type (see also footnote 11 and [11]).
With reference to quantifiers, “∀” and “∃”, their meanings are fixed as:

\[ r(\forall x) = U, \]
\[ r(\exists x) \in 2^U \setminus \{\emptyset\}. \]

**Formula evaluation.** The evaluation of open formulae based on a certain interpretation comes next at issue. Let \( \alpha \) be atomic, \( \alpha(x, y) \) and \( \beta(y, z) \) be arbitrary formulae, where \( x \) and \( z \) may represent lists of variables: \( x = x_1, \ldots, x_n \); \( z = z_1, \ldots, z_m \); and let \( (P y)z \) be a singular conceptual relation with respect to \( y \).

The extension of \( \sim \alpha \) is defined as:

\[ r(\sim \alpha) = U^n \setminus r(\alpha). \]

The extension of a predicative relation formed by the formulae \( \alpha(x, y) \) and \( \beta(y, z) \) is:

\[ r(\alpha(x, y), \beta(y, z)) = \{ (\xi, \eta, \zeta) : (\xi, \eta) \in r(\alpha(x, y)), (\eta, \zeta) \in r(\beta(y, z)) \}. \]

\[ r(\sim(\alpha(x, y), \beta(y, z))) = \{ (\xi, \eta, \zeta) : (\xi, \eta) \in r(\sim(\alpha(x, y))), \text{ or } (\eta, \zeta) \in r(\sim(\beta(y, z))) \}. \]

The evaluation of closed formulae expects that references attached to “∀y, \beta(y, z)” and “∃y, \beta(y, z)” types of expressions, i.e., quantified subjects, are already at hand:

\[ r(\forall y, \beta(y, z)) = \{ (\eta, \zeta) : (\eta, \zeta) \in r(\beta(y, z)), \eta \in r(\forall y) \}, \]
\[ r(\exists y, \beta(y, z)) = \{ (\eta, \zeta) : (\eta, \zeta) \in r(\beta(y, z)), \eta \in r(\exists y) \}. \]

Then for closed formulae: “\( \alpha(x, y), (P y)z \)” (\( z \) may be missing: names can be treated as 0-ary functions), “\( \alpha(x, y), \forall y, \beta(y, z) \)” and “\( \alpha(x, y), \exists y, \beta(y, z) \)” with respect to the variable \( y \):

\[ r(\alpha(x, y), (P y)z) = \begin{cases} \{ (\xi, \eta, \zeta) : (\xi, \eta) \in r(\alpha(x, y)), (\eta, \zeta) \in r((P y)z) \}, & r((P y)z) \subseteq, r(\alpha(x, y)) \}, \\
\text{nil otherwise,} & \end{cases} \]

\[ r(\alpha(x, y), \forall y, \beta(y, z)) = \begin{cases} \{ (\xi, \eta, \zeta) : (\xi, \eta) \in r(\alpha(x, y)), (\eta, \zeta) \in r(\forall y, \beta(y, z)) \}, & r(\forall y, \beta(y, z)) \subseteq, r(\alpha(x, y)) \}, \\
\text{nil otherwise,} & \end{cases} \]

\footnote{It should be reminded that in this construction the reference of “∃x” is not unique.}
It is to be observed that $r(x, y) \in R(x, y)$ may occur even if $r(x, y)(x, y) = \emptyset$, since $r[\forall y, (\beta(y, z)] = \emptyset \subseteq y \in r[\forall y, (\beta(y, z)]$. The same holds for names without reference, $r[x, y). (P y) = \emptyset$ only if $r[(P y) x] \subseteq y \in r[\forall y, (\beta(y, z)]$.

For negations of formulae let the following hold:

$$
r(\neg(x), (P x) z) = r(\neg(x), (P x) z),
$$

$$
r(\neg(x), \forall x, (\beta(x)]) = r(\neg(x), \exists x, (\beta(x))),
$$

$$
r(\neg(x), \exists x, (\beta(x))) = r(\neg(x), \forall x, (\beta(x))).
$$

As to formulae containing contraction variables, the referred subject gets its reference from that open subformula which is marked by the variable following “!” in the contraction variable symbol. Let $\alpha(x \to y) = \alpha’(x), Qx \to y$ denote a formula containing the “$x \to y$” contraction variable with $Q$ as quantifier symbol and “$\pi(y)$, $\sigma(y)$” the referred subformula of the considered text. Apparently, $y$ is treated as a global variable. Then the reference of the associated subject containing the contraction variable can be determined as $r(x, y) = r(x, y) \subseteq y \in r[\forall y, (\beta(y, z)]$ where $\sigma(x)$ is meant as follows:

$$
\sigma(x) = \begin{cases} 
\beta(x), \gamma(x) & \text{if } \pi(y), \sigma(y) = \beta(y), \forall y, \gamma(y) \text{ or } \beta(y), \exists y, \gamma(y), \\
(A x) z, \alpha(z) & \text{if } \pi(y), \sigma(y) = [\pi(y), (A z) \alpha(z)]
\end{cases}
$$

Self-reference, i.e., a contraction referring to the same subformula in which it is contained can be ignored.

Finally, for any double negated formula:

$$
r(\neg(\neg x)) = r(x).
$$

The truth-value evaluation process of any closed formula, $\alpha$, can be reduced to checking the non-emptiness of its extension:

$$
|\alpha| = \begin{cases} 
\text{true} & \text{if there exist } r(\alpha) \neq \emptyset, \\
\text{false} & \text{otherwise}.
\end{cases}
$$

Some unusual features of the evaluation process may be worth mentioning. According to the definition both open and closed types of formulae can have an associated extension which is not like the case in classical logic. Furthermore, formulae with the same truth values can often be evaluated with different references (e.g.: $|\alpha \land \forall x, b x| = |\neg(\neg x).)

---

$^7$ $a(\ldots, x, \ldots) \subseteq y \in \beta(\ldots, x, \ldots)$ abbreviates the relation $a(\ldots, x, \ldots) = \{x: (\ldots, x, \ldots) \in a(\ldots, \ldots)\} \subseteq (y: (\ldots, x, \ldots) \in R(\ldots, x, \ldots)) = R(\ldots, x, \ldots)$. On the other hand, $\emptyset$ denotes formally the same as $\emptyset$, i.e., the empty set, with the same properties, except that it is reserved only for identifying the case when the inclusion stipulation does not hold.
Proposition 3.1 (Contraposition rule). If \( \pi(x, y), \sigma(x, z) \), "\( \pi(x, y), \forall x, \sigma(x, z) \)" and
\[ \sim \{\sim \pi(x, y), \sigma(x, z)\}\forall x \]
are formulae, then
\[ \sim \pi(x, y), \forall x, \sigma(x, z) \] and \( \sim \{\sim \pi(x, y), \sigma(x, z)\}\forall x \)
are logically equivalent in the sense that they have simultaneously nil or non-nil reference in any context, respectively.

Proof. Obviously, \( r(\pi(x, y), \forall x, \sigma(x, z)) \neq r(\sim \{\sim \pi(x, y), \sigma(x, z)\}\forall x) \) although \( r(\pi(x, y), \forall x, \sigma(x, z)) \subseteq r(\sim \{\sim \pi(x, y), \sigma(x, z)\}\forall x) \). However, \( r(\forall x, \sigma(x, z)) \subseteq r(\pi(x, y)) \equiv r(\forall x) \subseteq r(\sim \{\sim \pi(x, y), \sigma(x, z)\}, \forall x) \), consequently \( \pi(x, y), \forall x, \sigma(x, z) \) and \( \sim \{\sim \pi(x, y), \sigma(x, z)\}\forall x \) have both nil or non-nil reference simultaneously, according to the definition of formula evaluation. Concerning the unbound variable parameters \( y \) and \( z \), it is easy to see that
\[
\begin{align*}
  r(\pi(x, y), \forall x, \sigma(x, z)) & =_{y,z} \{ (\eta, \xi, \zeta) : (\xi, \eta, \zeta) \in [r(\pi(x, y)) \cap r(\sigma(x, z))] \} \\
  & \cup \\
  & r(\sim \sigma(x, z)) \\
  & r(\forall x, \sigma(x, z)) \subseteq r(\pi(x, y)) \\
  & = \{ (\eta, \xi, \zeta) : (\xi, \eta, \zeta) \in r(\pi(x, y)) \cup r(\sim \sigma(x, z)) \\
  & \cup \\
  & r(\forall x, \sigma(x, z)) \subseteq r(\pi(x, y)) \} \\
  & = \{ (\eta, \xi, \zeta) : (\xi, \eta, \zeta) \in r(\pi(x, y)) \cup r(\sim \sigma(x, z)) \\
  & \cup \\
  & r(\forall x) \subseteq r(\sim \{\sim \pi(x, y), \sigma(x, z)\}) \} \\
  & =_{y,z} r(\sim \{\sim \pi(x, y), \sigma(x, z)\}\forall x)
\end{align*}
\]
also holds according to the rules of formula evaluation that completes the proof. \( \Box \)

3.2. \textit{iCTRL syntax and semantics}

An intensional extension of conformal text representation can easily be realised according to the principles summarised roughly in the previous section.

3.2.1. Syntax

The only modification of the syntax is based on (1). Let \( \alpha \) denote an arbitrary formula and \( x \) a contextual variable parameter, then
\[
[\alpha]_x
\]
denotes a context function that is also a formula. Additionally, if \( w \) is an \( n \)-place contextual conceptual relation parameter (including the logical constants of \( \forall \) and \( \exists \)) and \( A \) is a context
identifying index, i.e., singular conceptual relation parameter then, provided that at least
one of the \( x_i \) variable parameters is a contextual one

\[ w, x_1, \ldots, x_n \text{ and } A, x \]

are also atomic formulae.

### 3.2.2. Semantics

The interpretation of the new syntactic elements introduced above is the following. Beside \( U \), the object domain, another domain for the contexts, \( W \), should be postulated. The extension of any \( n \)-ary contextual conceptual relation, \( w \), index, \( A \), and context function of any formula, \([\alpha]x\), can be given relative to them:

\[
\begin{align*}
\tau(w, x_1 \ldots x_n) & \subseteq V^n \quad (V = U, W), \\
\tau(A, x) & \subseteq W, \\
\tau([\alpha]x) & = \{ (\xi_1, \ldots, \xi_n, \xi) : \xi \in W, (\xi_1, \ldots, \xi_n) \in \tau(\alpha) \subseteq V^n, \\
& \quad (\xi_1, \ldots, \xi_n) = w_\alpha(\xi), \ w_\alpha : W \rightarrow V^n \} \quad (V = U, W).
\end{align*}
\]

The following additional properties of compound context function interpretation are postulated:

\[
\begin{align*}
\tau([\alpha, \beta]x) & = \tau([\alpha]x, [\beta]x), \\
\tau(\neg[\alpha]x) & = \tau(\neg[\neg \alpha]x).
\end{align*}
\]

The former equivalence expresses that context function scopes can be decomposed or
reduced, and vice versa, the latter one declares the principle of excluded middle expressible
as:

\[
\neg[\alpha]x, \forall x, \neg[\neg \alpha]x.
\]

\[

\neg[\neg \alpha]x, \forall x, \neg[\alpha]x.
\]

All other rules of formula evaluation remain unchanged with respect to ordinary text
representation.

Before proceeding and considering more complex applications, a simple illustrative
ordinary language example is discussed that probably makes the concepts introduced above
more easily understood.

**Example 3.2.** Let the sentence: “What is very useful is useful.”, be examined in view
of formula construction and its validity verified by using the resolution algorithm (see
Appendix A for details).

Having a brief look at this sentence, there is no doubt that the constituent “very useful”
should be considered as intensional, hence it is expressible as:

\[
\text{very } y, \text{[useful } x\text{] } y.
\]

So as to understand better this formulation step it should be noticed above all that “useful x” may certainly refer to a natural object while “very y” cannot. Instead of referring to a natural object “very” modifies the meaning of “useful” like the following: if “useful x” can be interpreted in some worlds, referred by [useful x]y, then these worlds can also be referred to by “very y” so as to be able to refer those very worlds in which something is really “very useful”. 8

The corresponding formula with the same content as the example sentence above appears seemingly like

\[ \text{useful} \times \text{U} \times \text{very} \times \text{y} \rightarrow \text{useful} \times \text{U} \times \text{very} \times \text{y} \]

It should be read as: for any natural object, referred to by x, and for any world (or context), indexed by y, in which something, referred to by x, is very useful (in the sense given above), those are also useful objects according to the same context. For testing validity its negation should be proven to be unsatisfiable:

\[ \text{useful} \times \text{U} \times \text{very} \times \text{y} \rightarrow \text{useful} \times \text{U} \times \text{very} \times \text{y} \]

Then the clause transforming process, that is just reduced now to the skolemisation stage of the formula, results in the following clause set:

\[ \neg \text{useful} \times \text{U} \times \text{very} \times \text{y} \rightarrow \text{useful} \times \text{U} \times \text{very} \times \text{y} \]

which is obviously unsatisfiable, hence the original formula is valid.

4. Relation between the classical and text representational language of logic

4.1. CFOL and CTRL

One of the very first questions arising in this context is what kind of relation can be verified between text representational and the classical first-order language, which is just the next issue.

Proposition 4.1. Classical language of first-order logic is totally involved in the language of text representation.

Proof. According to the well-known grammar and interpretation rules of first-order logical language the following rewriting rule set can be proposed for mapping first-order expressions to the corresponding text representational ones. 9 Name and predicate/conceptual

8 This way of reflection resembles a literary model very much. Sei Shonagon a lady-in-waiting of the Japanese Imperial Court was especially fond of making records in her diary (Makura no soshi, 991–1000 A.D.) in a way like listing her experiences explicitly under titles as: unpleasant situations, amusing situations, etc. (A. Waley, The Pillow-Book of Sei Shonagon, London, 1928.)

9 Note that these rules operate on that type of propositional connectives, which are used to formalise logical fine structure.
relation symbols of classical logic and text representation, respectively, can be mapped to each other as:

\[
\begin{align*}
& a \leftrightarrow A x; \\
& y = f(x) \leftrightarrow (F y)x; \\
& p(x, \ldots) \leftrightarrow p' x \ldots.
\end{align*}
\]

For any open formulae, \( \alpha \) and \( \beta \), of classical logic, there are corresponding ones, \( \alpha' \) and \( \beta' \), of text representation as shown below:

\[
\begin{align*}
& \neg \alpha \leftrightarrow \neg \alpha'; \\
& \alpha \& \beta \leftrightarrow \alpha', \beta'; \\
& \alpha \Rightarrow \beta \leftrightarrow \neg (\neg \beta', \alpha').
\end{align*}
\]

Similarly, for any classical formulae, either for closed ones, \( \alpha \), or for open ones with respect to the variable symbol \( x \), \( \alpha(x) \) and \( \beta(x) \), and for the corresponding formulae, \( \alpha' \), \( \alpha'(x) \) and \( \beta'(x) \), of text representation the following rules are valid:

\[
\begin{align*}
& \neg \alpha \leftrightarrow \neg \alpha'; \\
& \alpha(x)(a/x) \leftrightarrow \alpha'(x), A x. \quad (\leftrightarrow \alpha'(x), \forall x, A x.); \\
& \forall x.\alpha(x) \leftrightarrow \alpha'(x), \forall x; \\
& \exists x.\alpha(x) \leftrightarrow \alpha'(x), \exists x.
\end{align*}
\]

It is also clear that if there is a common interpretation to a set of corresponding pairs of closed classical first-order and text representational formulae, then their evaluation processes result in the same truth values: their evaluation processes are equivalent to each other. □

The above argumentation encourages the belief that the inverse mapping also exists, just as if the two languages of logic were truly equivalent, which statement however does not hold. It is not very hard to find counter examples proving the contrary, justifying the proposition that text representational language totally involves the language of classical first-order logic. So as to show this, some everyday natural language sample sentences are considered.

**Example 4.1.** Let the typical sentence scheme of predicate logic, “Any \( F \) is-\( G \).”, be considered and formalised both in classical first-order logic and also in text representation. Concerning classical first-order logic, *this sentence has to be transformed first into another one in a truth-value invariant manner*, that is a generally accepted technique in any process of formula construction, and which is verified to be correct just intuitively up to the moment. Consider the subsequent transcripts: “If something is-\( F \), then it is-\( G \).”, “For anything, if it is-\( F \), then it is-\( G \).” This latter paraphrase yields the following first-order formula:

\[
\forall x. f(x) \supseteq g(x). \quad \text{For anything, if it is-} \ F, \text{ then it is-} \ G.
\]
Consequently, instead of the original natural language sentence one of its paraphrases was formalised in the classical logical language with the implicit hypothesis that they are all logically equivalent. On the other hand, both the original sample sentence and its paraphrases can be represented by separate text representational formulae:

\[ g(x), \forall x. f(x) \iff \forall x. f(x) \supset g(x). \quad \text{Any F is-G.} \]

\[ f(x), \exists x. \supset g(y), \forall y \to x. \quad \text{If something is-F, then it is-G.} \]

\[ \neg(g(x), f(x)) \forall x. \iff \forall x. f(x) \supset g(x). \quad \text{For anything, if it is-F, then it is-G.} \]

At this stage it is clear that \( g(x), \forall x. f(x) \supset g(x) \) and \( \neg(g(x), f(x)) \forall x. \) formalise the same sentence pattern (“For anything, if it is-F, then it is-G.”) in a synchronous way, i.e., both of them have the same subject, \( \forall x. \), and the predicates are structured in the same way, \( f(x) \supset g(x) \) and \( \neg(g(x), f(x)) \), respectively. At the same time truth-value invariant paraphrasing can be made explicit and evident:

\[ |\neg(g(x), f(x)) \forall x.| = |g(x), \forall x. f(x)|. \]

Actually, the natural language sentences “For anything, if it is-F, then it is-G.” and “Any F is-G.”, are truth-value-equivalent.

A similar statement holds in the dual case for the series of sentences “Some F’s are G.”

One more example of everyday language points out again the flexibility of CTRL and also the fidelity of CTRL formulae to natural language structures.

**Example 4.2.** Let the sentence “John walks.” (in Hungarian: “János sétál.”) be considered and formalised immediately as:

\[ \text{walk}(John) \iff \text{walk } x, \text{John } x. \quad \text{John walks.} \]

If the original sentence subject were negated, “Nem János sétál.” (Not John walks.), which can be interpreted approximately as “It is not John who walks.” (but someone else), then classical first-order language fails to formalise this aspect, while the corresponding text representation formula can represent it literally as follows:

\[ \text{walk } x, \exists x, \neg \text{John } x. \quad \text{It is not John who walks.} \]

10 It should be pointed out that the open subformula \( \neg(g(x), f(x)) \) stands exactly for \( f(x) \supset g(x) \). Actually, text representation language does not even have any other device to express the content of the “if …, then …” connective in this type of context (see argumentation in Section A.1.1).

11 According to the preceding definitions, the above example expresses the same content.

\[ f(x), \exists x. \supset g(y), \forall y \to x. \]

can be rewritten as follows:

\[ |f(x), \exists x. \supset g(y), \forall y \to x.| = |f(x) \exists x. \supset g(y), \forall y, f(x)| = |\neg f(x) \forall x, \forall y . \lor \neg(g(y), f(y)), \forall y|. \]

\[ = |\neg(g(y), f(y)) \forall y.| = |\neg(g(y), f(y))| = |g(y), \forall y, f(y)|, \]

which is just the expected result. Now it is proven formally that the sentences: “If something is-F, then it is-G.” and “Any F is-G.”, have indeed the same truth values.

12 For an exhaustive description of this type of grammatical problem the reader is referred to [2].
As a matter of fact this could be rephrased roughly by the next CFOL formula:

$$\exists x [ \text{walk}(x) \land x \neq \text{John}]$$

It is not John who walks.

The meaning of the original sentence (Nem János sétál) is much broader, and “~John x” is possibly not the optimal equivalent matching the natural language expression “not John”, but still referring to anybody except John covers more or less this meaning.

It should be noted that the subject is in focussed position in the Hungarian sentence (Hungarian has a topic-comment type of phrase structure [2]), which is not perceptible in English entirely expressing this by a preparatory subject construction. An obvious problem arises at this point: what happens if the predicate were focussed on instead in the sense that “Somebody walks, but it is not John.” Actually, the intended literal predicate focussing is ungrammatical in Hungarian: “Sétál nem János.” (find accurate reasoning in [2]), however, the literal translation of the English version exists: “Valaki sétál, de nem János.” This sentence can be formalised in CTRL as:

$$\text{walk } x, \exists x. \sim \text{John } y, \forall y \rightarrow x.$$  Somebody walks, but it is not John.

while the corresponding CFOL formula still remains the same as above, as it is unable to distinguish them:

$$\exists x [ \text{walk}(x) \land x \neq \text{John}]$$

It is not John who walks.

$$\text{Somebody walks, but it is not John.}$$

The above examples show clearly after all that text representation language really subsumes the language of classical first-order logic. The latter is a special case of the former, which means that only those natural language sentences can be formalised in classical first-order logic the generalised subjects of which are either one of the pure quantifiers (or which can be rephrased in this manner) or an individual. Two types of counter example were constructed demonstrating that CTRL is wider than the language of first-order logic. Example 4.1 shows that a whole class of truth-value-equivalent formulae can occasionally be ordered to a certain first-order formula; on the other hand Example 4.2 reminds that CTRL shorthand is not easy to be rephrased by the classical first-order language. To summarise this the following position is taken on this issue.

Conclusion 4.1. CTRL and classical first-order language can be made technically equivalent in the sense that those CTRL tools that have no direct mapping in classical first-order language must be rephrased truth-value-invariantly within CTRL lacking those tools, initiating an indirect translation of the formula that is concerned.

Beside numerous efficiencies the language of text representation still has some deficiencies a typical instance of which is the next one.

$$r(\text{high } x, \text{horse } x) = r(\text{horse } x, \text{high } x)$$

is apparently valid, however, the corresponding formulae certainly cannot even have any similar meaning, in fact, “horse high” kinds of constructions can hardly be interpreted as a meaningful natural grammatical phrase at all. Thus, predicative relation cannot be taken as commutative.
Furthermore, there is an application area where CTRL technique of logical text modelling seems to be absolutely inefficient. This is automated natural language translation. The obvious reason for that is just the thoroughly represented sentence structure of CTRL formalism that differs from language to language raising a similarly difficult translation problem as the original case.

4.2. Relation between intensional CFOL and iCTRL

Ordinary conformal text representation alone seems to have many advantages over classical first-order language of logic, outlined in Section 4.1 and earlier in [10–12]. Actually just ordinary CTRL itself leads out of the frames of CFOL. A similar behaviour can be observed as intensional logic is concerned. It clarifies many details due to its special qualities of descriptiveness that can be best appreciated by studying examples reminding of the limits of the classical language of intensional logic [13].

A short illustrative analysis of classical modal calculi (that of S4 and the converse of the Barcan formula) is taken next into focus. This is, however, by no means just technical or limited to syntactic manipulations only. On the contrary, it helps to understand why classical interpretation of S4 is inexact, and how a plainer reasoning for the validity of the converse of the Barcan formula can be recognised making use of the expressive character of iCTRL.

A classical resolution algorithm is chosen as a reasoning device [7], which is adapted to text representational context (summarised in Appendix A). Other well-known and prominent methods could evidently have been utilised or referred to, like, e.g., tableau proof system [16], or the sequent based proof method of Jackson et al. [4] that is worked out especially for modal predicate logic. Actually, the one suggested here seems to be simple and demonstrative enough, supplying clear and rational explanations to trace back.

4.2.1. Analysis of S4

Provided that the accessibility relation is transitive, according to Kripke-semantics, the validity of the S4 axiom is implied:

\[ N\alpha \supset N N\alpha. \]  

What seems rather problematic here is how \( N N\alpha \) should be interpreted in (2) in spite of the fact that traditionally it is believed to be correct as \( N(N(\alpha)) \). Yet, the following analysis of the relationship between the scheme \( N\alpha \supset N N\alpha \) and the transitivity condition is going to show that the case is unlike what is generally expected.

In order to be able to decompose accurately the structure of \( N N\alpha \), simple necessity, \( N\alpha \), should be examined first. According to Kripke-semantics \( N\alpha \) is true in an interpretation if in any interpretation accessible from that one \( \alpha \) is evaluated true. The text representation syntax has the advantage that this definition can be formalised literally as follows: \([\alpha] x\) represents the formula, \( \alpha \), with respect to an interpretation, \( x \); \([Vx]y\) refers to any interpretation, \( x \), accessible from an interpretation, \( y \); hence, provided that “0 y” stands

13 Unfortunately, this method is proven to be incomplete due to the restrictive presupposition of seriality on the accessibility relation.
Table 1
Ways of expressing $NN\alpha$

<table>
<thead>
<tr>
<th>Classical first-order formula</th>
<th>Text representation formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N(N(\alpha))$</td>
<td>$[[[\alpha]x, \forall x]y, \forall y]w, 0 w.$ (i')</td>
</tr>
<tr>
<td>$N(N(\alpha))$</td>
<td>$[[[\alpha]x, \forall x]y, [\forall y]w]0 w.$ (i'')</td>
</tr>
<tr>
<td>$N(N(\alpha))$</td>
<td>$[[[\alpha]x, [\forall x]y]y]w, 0 w.$ (ii')</td>
</tr>
<tr>
<td>$N(N(\alpha))$</td>
<td>$[[[\alpha]x, [\forall x]y][\forall y]w]0 w.$ (ii'')</td>
</tr>
<tr>
<td>$N(N(\alpha))$</td>
<td>$[[[\alpha]x, [\forall x]y]y][\forall y]w, 0 w.$ (iii')</td>
</tr>
<tr>
<td>$N(N(\alpha))$</td>
<td>$[[[\alpha]x, [\forall x]y]y]w, 0 w.$ (iii'')</td>
</tr>
<tr>
<td>$N(N(\alpha))$</td>
<td>$[[[\alpha]x, [\forall x]y]y][\forall y]w]0 w.$ (iii'')</td>
</tr>
</tbody>
</table>

for the present interpretation, “$[\forall x]y, 0 y$” represents those ones which are accessible from the present one; and finally “$[[[\alpha]x, (\forall x) y], 0 y$” formalises the prescribed content, i.e., $\alpha$ is true in any interpretation accessible from the one taken as the present interpretation.

Formalising double necessity is a bit more complicated. Table 1 summarises the possible candidates for expressing this content. There are numerous formally feasible scope patterns. Without intending to determine in advance which are the ones that suit intuitively best the requirements, it should be noted that the intensional extension of CTRL leads out of the frames limited by classical language of intensional logic; i.e., the latter is totally involved in the language of intensional text representation.

It must be indicated clearly that neither the * expressions, which are of course ungrammatical yet no better notation appears to be able to paraphrase the corresponding concept, nor the expressions marked by ** are acceptable relative to the traditional interpretation of the problem.

To keep the discussion compact only the (ii) option is chosen for a detailed examination, the other alternative, (iii), does not produce novelty any more. It should be pointed out that options (i') and (i'') are obviously out of question, although, this embarrassing situation can be resolved by postulating a more or less self-evident additional scheme:

\[
[[[\alpha]x, \forall x]y, [\alpha]x]\forall y]w, 0 w.
\]

specifying that for any proposition that is asserted in a reduced context, indexed only by $x$, the same can be stated in a compound one, indexed by $x$ and $y$ respectively.

The validity of (2) relative to both conditions of transitivity (3)

\[
(((\forall x]w, [\forall x]y]w)[\forall y]w), \forall w.
\]

\[
(((\forall x]w, [\forall x]y][\forall y]w), \forall w.
\]

requires to show that both of the formula sets composed by the variants ' and " are unsatisfiable.

\[
[[\alpha]x, \forall x]w, 0 w.
\]

\[
\sim([[[\alpha]x, [\forall x]y]y[\forall y]w, 0 w.
\]

\[
\sim([[[\alpha]x, [\forall x]y][\forall y]w]0 w.
\]
The transitivity conditions of (3) express the fact that for any world, referred by “∀w”, the alternatives’ alternatives, referred either by “[[∀x]y]w, [∀y]w∀w” or by “[[∀x]y, [∀y]w]∀w”, are also the alternatives of “∀w”, i.e., are included in “[∀x]w, ∀w”.

Of course the ”0 option is sufficient to study, the ”00 case can be treated similarly. The set of formulae at issue regarding satisfiability is 

3, 4, 5/0, which after routine operations of clause decomposition is equivalent to the following:

\[ T \cup x \cup w; T8 \cup x \cup y; 0 \cup w; \]

(6)

\[ T \cup x \cup w; T \cup x \cup y; [A x]y \cup w; [B y]w; 0 \cup w. \]

(7)

\[ [[∀x]w, [∀x]y]w, [∀y]w, ∀w. \]

(8)

Then from (6) and (7) directly follows

\[ (\sim[∀x]w), [∀x]y \cup w, [B y]w, 0 w., \]

(9)

which with (8) makes contradiction immediately.

\[ (\sim[∀x]y]w, [A x]y \cup w), [B y]w; 0 w. \]

(10)

This formula states that in spite of presuppositions the world index “[[A x]y]w, [B y]w, 0 w” cannot exist.

**Conclusion 4.2.** The previous example reminds that the S4 axiom, as it is traditionally interpreted, cannot be proven simply by the assumption that the accessibility relation is transitive, although a supplementary presupposition can restore this state. However, the iCTRL reconstruction of the S4 axiom clears up the case by offering several additional feasible interpretations some of which obey really to that principle that is classically presupposed.

4.2.2. Analysis of the converse of the Barcan formula

The subsequent analysis demonstrates how efficiently iCTRL can be applied for detecting validity conditions under which a certain (set of) formula(e), now the converse of the Barcan formula can be verified. The converse of the Barcan formula expresses commutativity of existential quantifier and the possibility relation in a certain direction:

\[ \exists x. M[α(x)] ⊃ M[∃x. α(x)]. \]

(11)

So as to find out these validity conditions the following pair of text representational formulae should be considered instead obtainable simply by translating (11) literally:

\[ \{[[α(x)]y, ∃y]∀x]w, 0 w. \]

(12)

and

\[ \sim[α(x), 3x]y, ∃y]w, 0 w., \]

(13)

which are analogously achievable like in the case of simple necessity in the previous section. These are equivalent to the next clause set concerning satisfiability:

\[ \{[[α(x)]y, A y]B x]w, 0 w. \]

(14)

\[ \{[[\simα(x)]y]w, [[∀x]y]w]∀y]w\}0 w. \]

(15)
The most simple and obvious one of several possible answers to the problem posed here can be reached by considerations appearing to be rather formal at first sight, mostly because it can be resolved from the current scene immediately:

$$\left\{ ([\forall x]y)w, ([\forall x]w)[\forall y]w \right\} 0 w. \tag{16'}$$

The encoded meaning can be recognised easily: $$(16')$$ expresses the fact that any accessible alternatives of the present universe, “$([\forall x]y)w, ([\forall y]w)0 w$”, should contain the universe of the present world, “$[\forall x]w, 0 w$”. $$(16')$$ itself also declares a philosophical self-evidence which in its present form rephrases an ontological interpretation of the principle of self-identity, $$(\forall x.x = x$$, asserting due to Coreth that “Seiendes ist, sofern es ist, notwendig Seiendes” (existent is, provided that it really exists, necessarily existent), or in other words: for everything that exists, if it really exists, the non-existence of that is impossible, [1, p. 228], [18, no. 124]. As a verification of $$(16')$$, put (14) and (15) together concluding to

$$\left\{ ([\forall x]y)w, [Bx]w[Ay]w \right\} 0 w. \lor \left\{ [\neg [\forall y]w, [Ay]w \right\} 0 w., \tag{17}$$

which in the subsequent step with $$(16')$$ results in a contradiction.

Another commonly known solution suitable for the same purpose is the condition

$$\left\{ ([\forall x]y, \forall x, [\alpha(x)]y)w, [\forall y]w \right\} 0 w. \tag{16''}$$

(that can be found in reference books, see, e.g., [14]) stating that any accessible universe alternatives should contain the corresponding predicate extensions. This, even in its interpreted form, seems also to be a rather formal prescription. The proof can be performed analogously composing (17) with $$(16'')$$. It yields:

$$\left\{ ([\neg [\alpha(x)]y)w, [Bx]w[Ay]w \right\} 0 w. \lor \left\{ [\neg [\forall y]w, [Ay]w \right\} 0 w. \tag{18}$$

Then (18) with (14) makes a contradiction.

**Conclusion 4.3.** There is a transparent ontological ground for the converse of the Barcan formula, stating that any accessible alternatives of the present universe should contain the present one, in other words, the accessible world alternatives can only be wider than the present real one: even imagination cannot be independent from reality.

**Conclusion 4.4.** It is easy to prove that the ontological precondition, $$(16')$$, and the conventional one $$(16'')$$, are equivalent:

$$[[\forall x]y, \forall x, \forall y]z, 0 z. \equiv \left\{ ([\forall x]y, \forall x, [\alpha(x)]y\right\} 0 z, 0 z. 15$$

---

14 Note that both “$[[\forall x]y]w$” and “$[\forall x]w$” refer to the normal object domain by the variable symbol $x$, “$[\forall y]w$” to the world of alternatives of the present world, 0, by $y$.

15 Observe that $\neg[[\forall x]y]z/[[\alpha(x)]y]z$ is a valid substitution for $$(16'')$$, where $\neg[[\forall x]y]z$ stands for the universal predicate associated with the current world alternative.
Table 2
Logic schools and their representatives

<table>
<thead>
<tr>
<th>Stoic school</th>
<th>Peripatetic school</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parmenides (540–480)</td>
<td>Aristotle (384–322)</td>
</tr>
<tr>
<td>Diodoros Chronos (?–307)</td>
<td>Theophrastos (372–287)</td>
</tr>
<tr>
<td>Zenon (336–264)</td>
<td></td>
</tr>
<tr>
<td>Chrisippos (278–204)</td>
<td></td>
</tr>
<tr>
<td>Boethius (480–524)</td>
<td>Boethius (480–524)</td>
</tr>
<tr>
<td></td>
<td>Avicenna (980–1034)</td>
</tr>
<tr>
<td></td>
<td>St Thomas Aquinas (1225–1274)</td>
</tr>
<tr>
<td></td>
<td>Leibniz (1646–1716)</td>
</tr>
<tr>
<td>Frege (1848–1925)</td>
<td></td>
</tr>
</tbody>
</table>

4.3. Reasons of discrepancies between traditional logic and text representation—a historical retrospection

It might be astonishing that a rather philosophical discussion finishes this section, however, in order to make the gist of the presented ideas more perceptible and assist to judge their significance more thoroughly, a draft evocation of the historical background and making use of its perspective will be inevitable. On the other hand there is an apparent reason for trying to get to conclusions this way: “According to an ancient methodological principle there are no sciences that are able to determine their own subjects and methods. . . . The stand-point from which the subject and method of a science can be described is a position always outside that science but inside the domain of preliminary knowledge preceding any scientific analysis.” 16

According to a stereotyped belief of logic, the triumphant of the two thousand year contest fought by term logic (researched by Aristotle and his followers the Peripatetics) with propositional logic (investigated by the representatives of the Stoic school) seems to have been definitely the latter since the final decades of the last century. (See Table 2.) It was due to above all Frege’s activity. Corresponding to historical researches of logic [8], a syncretism came to be felt later within the compass of which even the Peripatetics themselves acknowledged the superiority of propositional logic, too. It is also generally agreed that the widely known development of modern symbolic logic could not have been carried out on the grounds of term logic. The academic public belief proceeds further. It is stated that the Peripatetic way of approach is supposed to have hindered the development of logic until the day of Leibniz, the last committed representative of the Peripatetic school. Moreover, personally he can be blamed for the fact that he was not able to surpass the tradition and reach the aims he set for himself [6].

“The respect for the Aristotelian logic of subject and predicate had another unfortunate effect on Leibniz’s thought, namely, that of making him try to explain

away propositions of a relational form. (...) But there was a second reason, unknown to Leibniz, why he could make little progress in the construction of an ideal language, and that was his failure to shake himself free from the subject-predicate dogma of traditional logic."

The above deeply rooted convinced critique of the Peripatetics could be extended by several similar others from literature, which is tinged only by some scattered attempts of opposition. Maróth devotes his book [8] to prove by the recently accessible works of Ibn Sina (Avicenna) that the dichotomy of the Peripatetic and Stoic propositional logic is inappropriate. There existed a propositional logic of the Peripatetics, too, hitherto unrecognised by researches, which occurred earlier than that of the Stoics and also survived it at least by thousand years. Maróth in his book tries to reconstruct the whole system of Peripatetic propositional logic, and he argues that its utmost appearance is not subordinate to that of the Stoics at all, just on the contrary exceeds it in many respects.

Another contribution to this is Klima’s paper [5], which constructs a logical language embedded into the first-order one for an analysis of the Middle Age logic, continuing the Peripatetic school, from a modern aspect. His result is suitable to represent subject-predicate structure, and without pointing it out explicitly, it is equivalent to the language of classical first-order logic. This attempt, however, can only resolve the doctrine that since the modern symbolic logic was developed strictly on the grounds of propositional logic, which is no doubt a historical fact, therefore it is the only and necessary way as well.

By an argumentation detailed earlier can be proven that for the determination of the so-called fine logical structure the functional (subject-predicate) structure of a statement is sufficient to be analysed, i.e., the fine logical structure can also be reconstructed on purely Peripatetic grounds. In other words Peripatetic logic can also be generalised at least to the level of classical first-order logic. Henceforth, the affair around Peripatetic and Stoic logic just seems to be an aspect problem: the two thousand years old opposition was not genuine, it was more or less a matter of education or habit.

The difference between the Peripatetic and Stoic attitude to logic then can be summarised in the following. The common base for determining the fine logical structure of statements, no matter if it is admitted either explicitly or just implicitly, is the analysis of the functional structures of theirs, which means essentially the recognition of subordinate and co-ordinate relations in action. The Stoic way of regarding, contrasted with the Peripatetic one, does not seem to distinguish subordination from co-ordination, consequently, it is compelled to map truth-value invariant the subordinate relations into co-ordinate ones. On the other hand, the Peripatetic approach seems to have suffered from authentic treatment of polyadic relations, which as a matter of course is not necessarily an insoluble problem, even if it has appeared to be unsolved up to the present. Nevertheless, links between them should also be reminded, for instance the Peripatetic attitude is smuggled back to modern logic (of Stoic mind) via the theory of description.

If the two thousand year old opposition has already been described like coming to an issue in favour of Stoic attitude, a series of questions arise at once the most obvious of which is: what is the point of re-disturbing this problem?
Any answer appears to be highly idiosyncratic at first sight. Everybody who has got used to one of them will find that one the only intuitive and preferable way of thinking. This argument should be highly appreciated, but it also has to be reminded that natural language, that after all is always the starting-point to any logical analysis, actually applies subordinate constructions. From this prospect a logical language that is able to map both the subordinate and the co-ordinate relations of natural languages invariantly may certainly be preferable. The logic of Stoics only seems to prove by its existence that co-ordinate relation alone could be sufficient to be used for the language of logic, which resembles the commonly known fact that, e.g., Sheffer’s one is sufficient to express any other propositional function, not implying that the ones like $\&, \&, \neg, \lor, \ldots$, which also have their natural language equivalents, should or may be neglected.

Thus, there are several pragmatic reasons for the Peripatetic mind. One is its fidelity of a higher degree enabling the practical performance of validation tests on formalised texts. Another fact is that the Stoic school representative classical first-order language has been proven to be totally included in the one of text representation that itself belongs to the Peripatetic branch. Further reasons are comprehensible if intensional improvements are also drawn into focus, namely to create a coherent and consistent syntax for the language of intensional logic, which has been the primary range of interest of this paper.

5. Conclusions

A theoretically new and efficient, but not in the least unprecedented logical language was introduced that gives promising and practical responses to some since long unanswered questions of this domain. Natural language text modelling is a challenge to represent both syntax and semantics. To represent semantics is a necessity, while representing syntax is optional but might be beneficial. The validation problem of formal logical text models against the corresponding natural language texts is one of its application domain that is addressed and answered primarily by iCTRL. This point is essential in the knowledge representation phase of AI systems when completeness and soundness of text–formula translation are supposed to be verified. Besides, this area is highly interdisciplinary relating to computational linguistics, logic, AI and other fields concluding to several additional consequences.

iCTRL is a new, natural language syntax consistent, language of intensional logic moreover it is also an effective logical type of knowledge representation technique. It introduces a new dimension into logical text modelling. In contrast with the language of traditional logic utilising co-ordination exclusively, subordination is also represented implying more exact modelling of natural language utterances. Historically, it can be believed as a backtracking to term logic of the Aristotelian line of tradition preceding the Frege initiated present-day dominant approach. Similar to Frege’s graph language a transparent graphical interpretation of the CTRL/iCTRL formalism is outlined that helps to visualise and clarify the concepts introduced in the paper.
Just as it follows conscientiously and strictly the most essential syntactic structures of natural languages CTRL/iCTRL turns out to be closer to the common sense. The proximity attribute of the new formalism relative to natural languages can be identified that is essential in so far as knowledge representation validation problems of larger text bases are concerned. An adaptation of resolution technique is summarised as a proof aid, although, it is equally important as the backbone of AI system inference engines.

Conformal text representation and its intensional extension have been proven to be proper superset of classical first-order and intensional logic, respectively. As consistently keeping to the subject-predicate type of textual knowledge representation CTRL/iCTRL need not execute any preliminary truth-value-invariant transformation that is generally accepted in the traditional method of formula construction. This very step cannot be proven to be valid within the frames of CFOL exactly, just informally and intuitively, however, it can be verified to be correct formally by CTRL methodology. iCTRL representation of classical modal operators reveals some misinterpretation pitfalls that hitherto has avoided notice, and that can be escaped.

The most significant consequence of the presented formal logical language in respect to AI is that it offers an elegant knowledge representation frame, which realistically enables automated knowledge acquisition: a computer-aided generation of textual knowledge bases built from natural language texts that is fundamental for text based information retrieval systems. Automated knowledge acquisition procedure is feasible in view of the facts that CTRL/iCTRL is grounded on the grammatical structure of natural languages, which, however, can be parsed automatically granted by issues of computational linguistics. Regulation or legal texts are extremely favourable substances for this purpose for several distinct reasons, however, any other traditional application fields of large text based information retrieval systems as library catalogues, press archives or other type of textual databases can also be a question of interest. Similarly to other knowledge representation tools iCTRL presents just a framework supposing further research on particular natural language models in case of any definite application.

Concerning theoretical outcomes, iCTRL can be treated as an alternative of the classical intensional language of logic that can contribute to a better understanding of logic of natural language structures, additionally, it also provides a subject motivating new researches of logic and computational linguistics. Further applications on philosophical logic area are also conceivable owing to the homogeneous structure of the iCTRL formalism, e.g., exploring background of logical paradoxes and their resolution possibilities. As logical language, iCTRL is especially interesting from the perspective of logic history, since it refutes by its existence centuries long scientific prejudices.

Acknowledgement

The author would like to express his special thanks and indebtedness to Dezső Holnapy without whose stimulus this work could never have come to fruition. The author also thanks László Neumann and the anonymous referees for helping to improve the paper making it more correct and easier to follow.
Appendix A. Inference with text representational formulae

A.1. Inference with ordinary text representational formulae

The subsequent sections of Appendix A are devoted to show how to find the way back to the resolution method, the probably best-known classical algorithm to test formula sets for unsatisfiability. Definitions and proofs are going to be limited to the extent that cannot be traced from the literature, most of the well-known details are to be omitted, however, commonly known fundamental definitions are repeated in terms of text representation to make comprehension easier.

A.1.1. Clause

The resolution method operates on a special kind of formulae, clauses, hence its idea should be reconsidered, together with the algorithm converting arbitrary CTRL formulae into clause format. It has to be pointed out that the classical clause concept is still accepted as far as formulae of propositional logic are concerned, although some changes are needed beyond this level, i.e., if the fine logical structure of proposition is also taken into account.

Definition A.1 (Literal). Positive (non-negated) or negative (negated) atomic conceptual relations are called literals.

Definition A.2 (Skolem normal form). A CTRL formula is considered to be in Skolem normal form, if it does not have any indefinite subject. Namely, it does not have any subject that is existentially quantified either explicitly or implicitly, i.e., all of its subjects are either universally quantified (explicitly or implicitly), or singular atomic conceptual relations (individual name or function).

A CTRL formula is in strict Skolem normal form if it is in Skolem normal form and all of its subjects are pure universal quantifiers or singular atomic conceptual relations.

Definition A.3 (Clause). A CTRL formula in strict Skolem normal form is regarded to be a clause if for the predicate parts, $\pi(x)$, of any of its (possibly nested) subject-predicate decompositions, “$\pi(x), \sigma(x)$”, one of the following conditions holds: $\pi(x)$ is either one single open formula standing in the scope of any number of negation symbols or a composite one set up by more than one open formulae compound by a predicative relation which is within the scope of an odd number of negation symbols. However, the corresponding subject part, $\sigma(x)$, is in the scope of none or at most an even number of negation symbols.

17 A subject with implicit existential quantifier means a $\forall$ type of one in the negative scope of a $\forall$ type of subject or in the scope of a negation.
Clauses of course need not necessarily be in strict Skolem normal form. The contraposition rule can be applied to move any component of a clause predicate into subject position and vice versa.

**Proposition A.1.** Any clause of text representational syntax is also a clause in the conventional sense.

Verification is straightforward based on definition.

**Example A.1.** Discuss why the following formulae are not clauses:

\[(a \land b) \forall x \forall y,\]
\[(a \land b) \exists y, a \land b \land (b \land y).\]

As for the first one, its predicate part consists of two open formulae compound by a predicative relation and it is within the scope of an even number of negation symbols. Actually, it generates a pair of clauses: \[(a \land b) \forall y, a \land b \land (b \land y).\] The second one contains an explicit existential quantifier, therefore it is not in Skolem normal form. Finally, the last one has an implicit existential quantifier in its subject.

A rewriting rule set applicable recursively on the input CTRL formula that converts it into a corresponding set of clauses is next at issue. Let the input formula be presumed that its scopes of negation symbols have already been reduced to minimum. This operation can also be applied later if necessary. Let \[\pi(x)\] and \[\sigma(x)\] represent open formulae with respect to the variable symbol \[x\], let \[\lambda\] denote any literal, and let \[\ldots l\] identify the \(i\)th stage of the algorithm with \(l\) referring to the list of exactly those variable symbols, which have been universally bound at some earlier stage \(j<i\):

1. **Rewriting the positive scope of a definite subject:**
   
   (a) \[
   \left[\pi(x), \forall x \right]^\lambda \rightarrow \left[\pi(x), \forall x \right]^\lambda_{(x \land y)}, \forall x
   \]
   
   (b) \[
   \left[\pi(x), (A \land x) \land z \right]^\lambda \rightarrow \left[\pi(x), (A \land x) \land z \right]^\lambda
   \]

2. **Eliminating the negative scope of a universal quantifier:**

   \[
   \left[\pi(x), \forall x, \sigma(x) \right]^\lambda \rightarrow \left(\neg \pi(x), \sigma(x) \right)^\lambda
   \]

3. **Eliminating the existential quantifier (Skolemization):**

   \[
   \left[\pi(x), \exists x, \sigma(x) \right]^\lambda \rightarrow \left(\forall x, (A_i \land x) \land y \right)^\lambda
   \]

\((A_i \land x) \land y\) denotes the \(i\)th Skolem function symbol introduced at this stage. It must be new in the sense that it cannot be one already occurring in any previous stages. \[(\forall x, (A_i \land x) \land y)^\lambda\] is the explicit existence statement for the individual referred by \((A_i \land x) \land y\) supplementing the output of the conversion algorithm. The subject term, \(\forall y\), of this formula, \[\ldots \forall y\], is
an abbreviation for the universal closure of [...] with respect to its free variable symbols, which are all the members, \( y_j \), comprising the list \( y \), i.e., \( \ldots ([\forall y_1] \forall y_2) \ldots \) for all \( y_j \) belonging to the index list \( y \).

(4) Rewriting positive and negative open formulae (in a predicate part):

\[
\begin{align*}
(a) \quad [\pi(x), \sigma(x)]_{y}^{l-1} & \rightarrow [\pi(x)]_{y}^{l} \\
(b) \quad \neg([\pi(x), \sigma(x)])_{y}^{l-1} & \rightarrow \neg([\neg \pi(x)]_{y}^{l}, \neg[\sigma(x)]_{y}^{l})
\end{align*}
\]

(5) Rewriting literals:

\[ [\lambda] \rightarrow \lambda \]

For clauses made up of ground propositions alone the classical notation and transformation algorithm of propositional logic is preferable.

**Proposition A.2.** The output of the above recursive algorithm is really a clause set.

**Proof.** Rules (1)–(3) ensure that the output is of strict Skolem normal form. Rule (4) guarantees that the subject-predicate decomposition of any input formula have either one single open formula or a set of open formulae compound by predicative relation in the scope of an odd number of negation symbols as its predicate part. Rule (5) stops the algorithm. \( \square \)

**Proposition A.3.** The original formula and the associated set of clauses are equivalent with respect to satisfiability.

**Proof.** Rules (1), (4b) and (5) do not make any change on the reference of the input formula.

Rule (2): (within the first-order equivalent sublanguage of text representation) both sides are nil-referenced or non-nil-referenced simultaneously, see the contraposition rule.

Rule (3): it is sufficient to see that if \( r[\pi(x), \exists x, \sigma(x)] \neq nil \) then a suitable reference set \( r \in r(\exists x) \) can be chosen such that \( r := r[(A x) y] \) would hold, and vice versa. \( y \) is the list of variable symbols occurring in the universal quantifiers holding \( \pi(x), \exists x, \sigma(x) \) in their positive scopes.

Rule (4a): the existence of \( r[\pi(x), \sigma(x)] \) implies the existence of both \( r[\pi(x)] \) and \( r[\sigma(x)] \), containing the reference of the corresponding subject simultaneously.

Finally, as far as the term “clause” is concerned, it is easy to see (cf. Section 4) that any clause of the text representational syntax within the first-order equivalent sublanguage of CTRL can be mapped to the corresponding clause of classical first-order logic, and vice versa. \( \square \)
Before proceeding, an example may help to clarify the old ideas newly expressed and to illustrate how the above rule set works.

**Example A.2.** Let the formula “\((b \land x \land y, \forall x \land y, \land o \land x \land y, \land z \land d \land y)\forall x((o \land x \land z, \land z) \land z) f \land x\)” be transformed into clause form. The sequence of formulae below follows exactly the above algorithm, figures in brackets identify the rules applied consecutively:

\[
\begin{align*}
(b \land x \land y, \forall y, \land o \land x \land y, \land d \land y)\forall x((o \land x \land z, \land z) \land z) f \land x & \\
\sim((b \land x \land y, \forall y, \land o \land x \land y, \land d \land y)\forall x) & \\
\sim((b \land x \land y, \forall y, \land o \land x \land y, \land d \land y)\forall x) & (2) \\
\sim((b \land x \land y, \forall y, \land o \land x \land y, \land d \land y)\forall x) & (1a) \\
\sim((b \land x \land y, \forall y, \land o \land x \land y, \land d \land y)\forall x) & (4b) \\
\sim((b \land x \land y, \forall y, \land o \land x \land y, \land d \land y)\forall x) & (5) \\
\sim((b \land x \land y, \forall y, \land o \land x \land y, \land d \land y)\forall x) & (2, 3) \\
\sim((b \land x \land y, \forall y, \land o \land x \land y, \land d \land y)\forall x) & (1a, 5) \\
\sim((b \land x \land y, \forall y, \land o \land x \land y, \land d \land y)\forall x) & (4b) \\
\sim((b \land x \land y, \forall y, \land o \land x \land y, \land d \land y)\forall x) & (5) \\
\end{align*}
\]

Finally, the converse of rule (2) can eliminate double negations and the result is the following pair of clauses:

\[
\begin{align*}
(b \land x \land y, \forall y, \land o \land x \land y, \land d \land y)\forall x((o \land x \land z, (A \land z)x) f \land x) & \\
(b \land x \land y, \forall y, \land o \land x \land y, \land d \land y)\forall x((o \land x \land z, (A \land z)x) f \land x) & \\
\end{align*}
\]

It may be a point of interest to have a brief look at these results and compare them to the corresponding classical clauses generated by the formula “\(\forall x((o \land x \land z, \land z) \land z) f \land x\)”:

\[
\begin{align*}
b \land x \land y \lor \sim o \land x \land y \lor \sim o \land x \land a(x) \lor \sim f \land x & \\
b \land x \land y \lor \sim o \land x \land y \lor \sim o \land a(x) \lor \sim f \land x & \\
\end{align*}
\]

Comparing them it is remarkable that the latter one is a standardised statement with an exaggerated size of the compound predicate part, namely the alternations of literals.
and with very poorly visible subjects, which are the Skolem functions and the unmarked universal quantifiers.

A.1.2. The resolution method

The problem that is to be faced next is how to adapt the well-known inference method and the associated concepts of substitution and unification.

**Definition A.4 (Substitution rule).** A formula \(\sigma^e(x), \tau(x)\) is taken to be a substitution with respect to the clause \(\pi(x), \sigma(x)\), provided that \(\sigma(x) = \forall x, \sigma^e(x)\), with the result of \(\pi(x), \tau(x)\).

**Proposition A.4 (Correctness of substitution rule).** The substitution rule represents a correct inference scheme.

**Proof.** Considering that \(r[\pi(x)] \subseteq r[\sigma^e(x)]\) and \(r[\forall x, \sigma^e(x)] \subseteq r[\pi(x)]\) are true, \(r[\tau(x)] \subseteq r[\pi(x)]\) is straightforward. \(\blacksquare\)

**Example A.3.** The result of the substitution \(b x, A x\) applied to \(a x, \forall x, b x\) is \(a x, A x\).

The following rephrasing of the basic resolution rule concerns only non-propositional clauses. Certainly, traditional resolution rule is still applicable for pure propositional purposes. The subsequent formal interpretation of basic rule of resolution seems rather complicated at first sight, however, its essence can be summarised in a quite illustrative way. Providing the predicate parts of the input clause pair are complementary, and their subject parts can be made common by a series of basic subject substitutions (i.e., the analogue of unification step) obeying the succeeding definition, then the case would conclude to a direct contradiction. So that it could be avoided, the negation of the unifying basic substitution set should be presumed that is the condition of preventing that potential contradiction. This direct consequence of the input clause pair is taken as their resolvent.

**Definition A.5 (Basic resolution rule).** The resolvent \(\theta\) for an input clause pair \(\pi\) and \(\rho\) is determined recursively as follows.

Let the output of the \(i\)th step of resolution algorithm be

\[
\langle[\pi_1(x_i), \sigma_1(y_i)]\rangle, \langle[\rho_1(y_i), \tau_1(y_i)]\rangle\rangle_{\alpha_i}.
\]

\(\alpha_i\) denotes that segment of the subject that has already been unified, and \(\theta_i\) the condition yielding this result. In the case of initial input it is \((\pi_1, \rho_1)\alpha_1, \alpha_1 = 0\) := \(\varepsilon\), \(\varepsilon\) denotes the empty formula and \(\pi_1 = \pi, \rho_1 = \rho\).

Let the variable parameters of the innermost predicate parts of \(\pi_1\) and \(\rho_1\) be unified: denoted by the same symbols, while the rest of the variables are standardised apart (denoted by symbols different from the ones unified earlier). Let \(L\) denote the common variable symbol list of the \(i\)th input clauses. Thus, \(L\) identifies by its variable symbol elements those subjects, which are to be unified.
Two main alternatives can be distinguished: either both subjects are bound by the corresponding innermost predicate part or at least one or possibly both of them is/are bound by a separate one. Additional alternatives get into the picture according to the subject types that are to be unified: none, one or both of the subjects may be either universally quantified or singular. These points yield the following rule set.

(1) **Subject unification:** if \( x_1, y_1 \in L \) (that means \( x_1 = y_1 \) according to the presumption) and

(a) **singular-singular subject unification:** if \( \sigma_i(x_i) = \tau_j(x_j) = (A x_i) \ldots \), then according to the convention that \( \omega_{i+1} \) is compound as \( \omega_{i+1} = \{([\ldots]\omega'_{i+1}\omega''_{i+1})\omega_i \) and similarly \( \theta_{i+1} = \{([\ldots]\theta'_{i+1}\theta''_{i+1})\theta_i \) the next input is \( (\pi_i, \rho_i)\omega_{i+1}, \omega''_{i+1} = \tau_j(x_j) \), \( \omega''_{i+1} = \theta''_{i+1} = \theta''_{i+1} = \varepsilon, \eta_{i+1} = (\ldots)\omega_{i+1} \); or

(b) **universal-universal/universal-singular subject unification:** if \( \sigma_i(x_i) = \forall x_i, \) \( \sigma^*_i(x_i) \) and if

(i) \( \tau_j(x_j) = \forall x_i, \) \( \tau^*_i(x_i) \), then \( \tau^*_i(x_i) = \exists x_i, \tau^*_i(x_i) \) or

(ii) \( \tau_j(x_j) = (A x_i) \ldots \), then \( \tau^*_i(x_i) = \tau_j(x_j) \), then the next input is \( (\pi_i, \rho_i)\omega_{i+1}, \omega'_i = \tau^*_i(x_i), \omega''_{i+1} = \varepsilon, \theta''_{i+1} = \tau^*_i(x_i), \theta'_{i+1} = \tau^*_i(x_i), \theta''_{i+1} = \tau^*_i(x_i), \eta_{i+1} = \varepsilon. \)

(2) **Subject omission:** if not (i.e., \( x_i \notin L \) or \( y_i \notin L \) or \( x_i, y_i \notin L \)) and

(a) **single subject omission:** if, for example, \( x_i \notin L \) but \( y_i \in L \), then the next input is \( \{\pi_i, \rho_i(y_i), \tau_i(x_i)\}[\omega_{i+1}, \omega'_i = \sigma_i(x_i), \omega''_{i+1} = \theta''_{i+1} = \varepsilon, \eta_{i+1} = (\ldots)\omega_{i+1} \}; or

(b) **double subject omission:** if \( x_i, y_i \notin L \), then the next input is \( (\pi_i, \rho_i)\omega_{i+1}, \omega'_i = \sigma_i(x_i), \omega''_{i+1} = \pi_i(y_i), \omega'_i = \theta''_{i+1} = \varepsilon, \eta_{i+1} = (\ldots)\omega_{i+1} \).

(3) otherwise the resolvent does not exist.

If the \( n \)th output \( (\pi_n, \rho_n)\omega_n \), reaches the state that \( \pi_n \) and \( \rho_n \) are complementary, \( \pi_n = \sim \rho_n \), then the resolvent for the initial input clause pair \( \pi \) and \( \rho \) is \( \theta = (\sim \rho_n)\eta, \eta = (((\ldots)\eta_n) \ldots)\eta_1, \) otherwise the resolvent does not exist.

The subsequent proposition characterises the relation between CTRL and conventional resolution rule.

**Theorem A.5 (Isomorphism).** The text representational rule of resolution and the traditional one are isomorphic.

**Proof.** The CTRL resolvent is a clause, since the subjects of the resolvent are either singular conceptual relations or existentially quantified predicative relations of literals, and the negation of a formula of this form just satisfies the criteria of being a clause.

The following bijective mappings can be defined between text representational and classical first-order terms, literals and clauses, respectively (they are similar to the ones introduced in Section 4).

**Terms:**

\[ a \leftrightarrow A x; \]
\[ y = f(x, \ldots) \leftrightarrow (F y)x \ldots; \]
Literals, conjunctive and disjunctive subformulae built up by literals:

\[ \lambda \equiv \lambda'; \quad (p(x, \ldots) \equiv p'x \ldots); \]
\[ \& \lambda_i \equiv \lambda_i'; \]
\[ \vee_i \lambda_i \equiv \sim \lambda_i'. \]

Clauses or closed disjunctive formulae:

\[ \forall x. \beta(x) \supset \alpha(x) \equiv \alpha'(x), \forall x, \beta'(x); \]
\[ [\forall x.]\alpha(x)](\alpha/x) \equiv \alpha'(x), A x, \]

where \( \alpha \) and \( \beta \) are a disjunctive and conjunctive subformulae, respectively.

It is clear at this stage that the two different clause types, i.e., the domains of operation, can one-to-one be mapped to each other. It will be shown that this mapping can be extended to the basic resolution rule, too.

(1) If \( x_i, y_i \in L \) (i.e., \( x_i = y_i \)) then

(a) if \([\forall x_i.]\rho_i(x_i)\alpha(\ldots)/x_i \equiv \rho'_i(x_i),(A x_i)\ldots\)

then \([\ldots]\epsilon\theta_i \equiv [\ldots]\epsilon\theta_i'\];

(b) if \([\forall x_i.]\rho_i(x_i)\alpha(\ldots)/x_i \equiv \rho'_i(x_i),(A x_i)\ldots\)

and if

(i) \( \forall x_i.]\sigma_i^*(x_i) \supset \rho_i(x_i) \equiv \rho'_i(x_i), \forall x_i, \sigma_i(x_i), \ldots\)

then

\[ \theta_i = [\ldots]\sigma_i(x_i)\theta_i[x_i, \nu_i^*(x_i)] \equiv \theta_i' \equiv \theta_i \]

(ii) \([\forall x_i.]\rho_i(x_i)\alpha(\ldots)/x_i \equiv \rho'_i(x_i), (A x_i)\ldots\),

then

\[ \theta_i = [\ldots]\sigma_i(x_i)\theta_i[x_i, \nu_i^*(x_i)] \equiv \theta_i' \equiv \theta_i \]

(2) If \( x_i \) or \( y_i \notin L \) then \([\ldots]\epsilon\theta_i \equiv [\ldots]\epsilon\theta_i'\].

The rest of the proof will confirm that the outcome of the preceding algorithm, the resolvents, can also be mapped one-to-one to each other, i.e., \( \theta = \theta_n \equiv \theta' = \theta' n \). This can be proven by the following argumentation. \( \theta \) can be verified to be a most general unifier for \( \pi_n \) and \( \sim \rho_n \), and \( \sim \theta' \eta \) will be shown that it can be rewritten to the traditional resolvent of the input clause pair that can be achieved by applying \( \theta \).

\( \theta \) is a most general unifier for \( \pi_n \) and \( \sim \rho_n \), since it is a special kind of mesh substitution (see [7]), namely, which is a permutation of an ordinary one. In order to obtain that unifier the standard mesh substitution algorithm should be applied on \( \pi_n, \sim \rho_n \) except that the normal sequential access to the members of the parameter lists is altered according to the order, which is given by the corresponding CTRL clauses in their order of subject bindings. For this case the well-known unification theorem holds without any further restriction.

So as to rewrite \( \sim \theta' \eta \) to a classical formula the subsequent procedure can be followed:

(1) Move negation symbol inward to the minimum scope:

(a) \( \sim \sim \alpha' \rightarrow \alpha' \); 

(b) \( \sim [\alpha' (x), \exists x, \beta' (x)] \rightarrow \sim \alpha' (x), \forall x, \beta' (x); \)
(c) $\neg[\alpha'(x), \beta'(x)], \forall x, \gamma'(x) \rightarrow \neg\alpha'(x), \forall x, \beta'(x), \gamma'(x)$;
$\neg[\alpha'(x), \beta'(x)], \forall x, \gamma'(x) \rightarrow \neg\beta'(x), \forall x, \alpha'(x), \gamma'(x)$;
(d) $\neg[\alpha'(x), (A x) \ldots] \rightarrow \neg\alpha'(x), (A x) \ldots$;

(2) **Rewrite the resulting formulae into a classical one:**
(a) $\alpha'(x), \forall x, \beta'(x) \rightarrow \forall x [\beta(x) \supset \alpha(x)]$;
(b) $\alpha'(x), (A x) \ldots \rightarrow \alpha[a(\ldots)]$;
(c) $\alpha'(x), \beta'(x) \rightarrow \alpha(x) \& \beta(x)$;

(3) **Rewrite the result into classical clause format by using rules of classical propositional calculus.**

The correctness of the above rule set is straightforward based on earlier issues presented in Section 4.

At this point the only remaining task is to show whether the output of this procedure is the same clause as the one normally achievable. To perform this it is sufficient to see that both types of clause are made up by literals mutually corresponding to each other, and that they also have mutually corresponding structures.

Evidently, the CTRL resolvent cannot contain other than exactly those literals which can be mapped on the ones consisting of the corresponding normal resolvent, therefore, it is satisfactory to show that the input literals preserve their “signs” in the output. According to the resolution algorithm only (1b) has any effect on the resolvent structure. Hence, one of the input clause subjects should remain in subject position while the other gets into predicative status, but ultimately into the scope of a negation symbol, which means that its “sign” is still preserved. Consequently, the CTRL rule of resolution keeps the “sign” of the output literals unvaried, and since the resolvent itself is a clause, their final structure corresponding to the classical one is assured, too.

**Example A.4.** Find a resolvent of $(p x y, \forall x, q x) A y.$ and $\neg p x y, (A x) y \forall y.$

Having noticed that the input clauses have complementary predicates, rule (1b)(ii) can be applied first that yields: $(p x y, \forall x, q x) \neg p x y, (A x) y \forall y.$ with the substitution $(\ldots) A y.$ Then the same scheme is repeated once more producing $(p x y, \neg p x y) (A x) y A y.$ while the valid substitution is $[(q x, (A x) y \forall y)] A y.$ As the output includes a predicative relation of complementary conceptual relations as its predicate part the resolvent clause is the negation of the last substitution: $\neg[(q x, (A x) y \forall y)] A y.$ $\equiv [\neg q x, (A x) y] A y. \lor \neg A y.$ A y.

The corresponding traditional resolution process performed on traditional clauses $\neg q x \lor p x a$ and $\neg p a(y) y$ yields the same result: $\neg q a(a).$

Other rules utilising contraposition scheme may complete this single one of basic resolution so that it could be applied in any context.

The existence of one particular clause, the empty clause, is implied by resolution. This concept belongs to the level of propositional logic so it is sufficient to refer to the original definition.

**Definition A.6** (Empty clause). The empty clause is the empty proposition (the alternation having no arguments), except it is taken unsatisfiable.
Generally an additional elementary rule of inference, factoring [7], is considered separately.

**Definition A.7** (*Factor*). Let \( \pi, \sigma = (\ldots (\pi_i, \sigma_i) \ldots \sigma_n) \), be a clause with the literals \( \lambda_1 \) and \( \lambda_2 \) occurring in \( \pi \) or \( \sigma_i \), \( i = 1, 2, \ldots, n \). Then the result of applying a set of basic substitutions of the type: “\( \forall x, \forall x \to x_i \)” or “\( \forall x, \exists x \)” making \( \lambda_1 \) and \( \lambda_2 \) identical in the same (either predicate or subject) position is a factor of the clause \( \pi, \sigma \).

**Example A.5.** Find factors of \( (p x y, \forall x, \sim p x x) \forall y, \sim p y y \).

Let the input first transformed to “\( \lnot(\sim p x y, \sim p x x, \sim p y y) \forall x \forall y, \lnot p y y \)” by contraposition rule then the substitution “\( \forall y, \forall y \to x \)” be applied. The result is “\( p x x, \forall x \).” In a similar manner “\( \forall x, \forall y \)” can also be obtained, but clearly, it is just a variant of the former one.

**Definition A.8** (*Resolution algorithm*). Resolution refutation is the iterative application of the basic resolution- and factoring rules to a given input set of clauses, that is continuously extended by the resolvents and factors, until the empty clause is reached.

From this point on it is sufficient to refer the reader to the literature with respect to fundamental properties of resolution, e.g., soundness- and completeness proofs.

The following example will demonstrate how the discussed version of the traditional resolution method works on an easy case of natural language inference.

**Example A.6.** Prove the subsequent instance of reasoning to be valid:

\[
\begin{align*}
\text{Horses are animals.} \\
\text{Horse heads are heads of animals.}
\end{align*}
\]

Formalising the premise seems more or less evident with the comment that obviously “horses” should be taken as “any horses” in this context. Nevertheless, the conclusion part is worth some additional considerations. “Horse heads”, i.e., “heads of a horses”, that can be made more explicit by the expression “head owned by a horse”, can be formalised here as “\( (\text{own } x y, \exists x, \text{horse } x) \text{head } y \)”, just to keep variants of genitive case as standardised as possible. Thus the sentences above can be reproduced as follows.

\[
\begin{align*}
\text{animal } x, \forall x, \text{horse } x. \\
\forall x, \text{horse } x \supset \text{animal } x \\
[ \text{own } x y, \exists x, \text{animal } x] \text{head } y \forall y \\
\exists z [\text{own } z y \& \text{horse } z] \& \text{head } y \supset \\
\exists [\text{own } x y \& \text{animal } x] \& \text{head } y]
\end{align*}
\]
The conventional resolution refutation procedure can be performed after negating the conclusion and transforming the resulting formula set into clause form:

\[
\begin{align*}
\text{animal} & \; \forall x, \text{horse} \; x. \\
\neg \text{own} \; x \; y, \forall x, \text{animal} \; x)A \; y \lor & \; \neg \text{own} \; x \; a \lor \neg \text{animal} \; x \lor \neg \text{head} \; a \\
\neg \text{head} \; y, A \; y. \\
\neg \text{own} \; z \; y, B \; z)A \; y. & \; \text{own} \; b \; a \\
\text{horse} \; z, B \; z. & \; \text{horse} \; b \\
\text{head} \; y, A \; y. & \; \text{head} \; a
\end{align*}
\]

Applying the basic resolution rule to \((\neg \text{own} \; x \; y, \forall x, \text{animal} \; x)A \; y \lor \text{head} \; y, A \; y,\) and \((\text{own} \; x \; y, B \; x)A \; y,\) the following resolvent clause can be achieved:

\[
\text{animal} \; x, B \; x. \lor \neg \text{head} \; y, A \; y. \lor \neg \text{animal} \; b \lor \neg \text{head} \; a
\]

Then resolving this conclusion by \(\text{head} \; y, A \; y,\) and after that by \(\text{animal} \; x, \forall x, \text{horse} \; x,\) the result is

\[
\text{horse} \; x, B \; x. \lor \text{horse} \; b
\]

Finally, it concludes to the empty clause, \(\text{nil},\) with \(\text{horse} \; x, B \; x,\) that is a member of the input clause set coming to a contradiction.

### A.2. Inference with intensional text representational formulae

Intensional extension makes no relevant difference from the inference method outlined previously. The only difference appearing in the resolution algorithm applied to iCTRL formulae compared with the one of non-intensional case discussed above occurs in the formula-clause conversion algorithm.

Context functions can be decomposed into their elementary parts (and can also be reunited, if needed), that allows to extend the definition of literal as follows: the context function of any literal also remains a literal implying the invariance of any subsequent steps.

The formula–clause conversion algorithm presented in Section 1.1 should be supplemented by the following rule (\(\alpha\) represents a formula and \(x\) a context variable symbol):

(6) Moving context variable outward:

\[
[[\alpha x]_y]^{-1} \rightarrow [[\alpha y]_x].
\]

The validity of this rule is straightforward. The resolution algorithm still remains applicable in its previous form without any change.

### Example A.7.

Prove the following Aristotelian syllogism to be valid.

\[
\begin{align*}
[[a x]y, \forall x, b \; x]z, [\forall y]z)0z & \lor \forall x, b(x) \supset N(a(x)) \\
[b x, \forall x, c \; x]y, 0 \; y. & \lor \forall x, c(x) \supset b(x) \\
[[a x]y, \forall x, c \; x]z, [\forall y]z)0z. & \lor \forall x, c(x) \supset N(a(x)).
\end{align*}
\]
Conventional resolution refutation procedure is carried out after negating the conclusion and performing the formula-clause conversion.

\[
[[a \ x]y, \forall x, b \ x \ z, [\forall y]z]0 z.
\]

\[
[b \ x, \forall x, c \ x]y, 0 y.
\]

\[
(((\neg[[a \ x]y]z, [A \ x]z)[B \ y]z)0 z.
\]

\[
([c \ x]z, [A \ x]z)0 z.
\]

Now, resolving \((([[a \ x]y]z; [\forall x]z; [b \ x]z)[\forall y]z)0 z. with \(((\neg[[a \ x]y]z, [A \ x]z)[B \ y]z)0 z. yields:

\[
\neg[b \ x]z, [A \ x]z)0 z.
\]

Next is the resolution of the previous result with \((b \ x]y, [\forall x]y, [c \ x]y)0 y. which concludes to

\[
\neg[c \ x]y, [A \ x]y)0 y.
\]

Finally this and \((c \ x]y, [A \ x]y)0 y. resolve to \textit{nil}, that is a contradiction.

References