Bilingual Logical Analysis of Natural Language Sentences

Marek Medveď, Aleš Horák, and Vojtěch Kovář

Natural Language Processing Centre,
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00, Brno, Czech Republic
{xmedved1,hales}@fi.muni.cz

Abstract. One of the main aims of logical analysis of natural language expressions lies in the task to capture the meaning structures independently on the selected "mean of transport," i.e. on a particular natural language used. Logical analysis should just offer a "bridge between language expressions."

In this paper, we show the preliminary results of automated bilingual logical analysis, namely the analysis of English and Czech sentences. The underlying logical formalism, the Transparent Intensional Logic (TIL), is a representative of a higher-order temporal logic designed to express full meaning relations of natural language expressions.

We present the details of the current development and preparations of the supportive lexicons for the AST (automated semantic analysis) tool when working with a new language, i.e. English. The AST provides an implementation of the Normal Translation Algorithm for TIL aiming to offer a normative logical analysis of the input sentences. We show the similarities and the differences of the resulting logical constructions obtained via both the input languages with a view towards wide bilingual logical analysis.

Key words: semantics, semantic analysis, logical analysis, Transparent Intensional Logic, TIL

1 Introduction

Meaning representations are usually understood as a "dense and shared" form of input sentences. In practical systems, many machine translation tools search for a kind of *interlingua*, a semantic representation used to convey the message content from source to the target language [1,2]. From the theoretical point of view, the meaning of natural language sentences can be fully expressed by using a kind of intensional logic formalism, which allows to represent the clausal relations inherently connected with the referents of real-world concepts. In the current paper, we lean on the formalism of the Transparent Intensional Logic, or TIL [3,4], that allows for a clear logical interpretation of many complex language phenomena.

In the following text, we present the latest results in the design and implementation of automated language-independent semantic analysis of natural language sentences. We show, how the AST tool [5] is adapted for the work with a new syntactic analyser and a new language of the input text. Primarily, AST worked in connection with the Czech language parser `synt` [6], which uses a strict meta-grammar of Czech to strictly decide the correctness of the input. This inevitably leads to lower coverage of borderline phenomena. The modular design of AST is proved via extending the input to processing another parser, namely the SET parser [7], that is based on a flexible pattern-matching dependency concept allowing to process all kinds of input sentences. The SET parser is extendible to other languages, which is demonstrated in this text by comparing the resulting logical constructions of the application of AST and SET to both Czech and English sentences.

In the following sections, we first briefly describe the `synt` and SET parsers with most emphasis on the differences between the parsers, then we enlist the requirements of the AST tool for obtaining the language-dependent lexical information, and finally present the first results of bilingual, i.e. Czech and English, logical analysis within the AST tool.

2 Parsing

The semantic processing of natural language sentence builds upon the result of structural syntactic analysis or *parsing*. As a prevalent type of presenting the hierarchical organization of the input sentence most parsers are able to provide a comprehensive representation in a form of a syntactic tree, which is also the form processed by the AST tool.

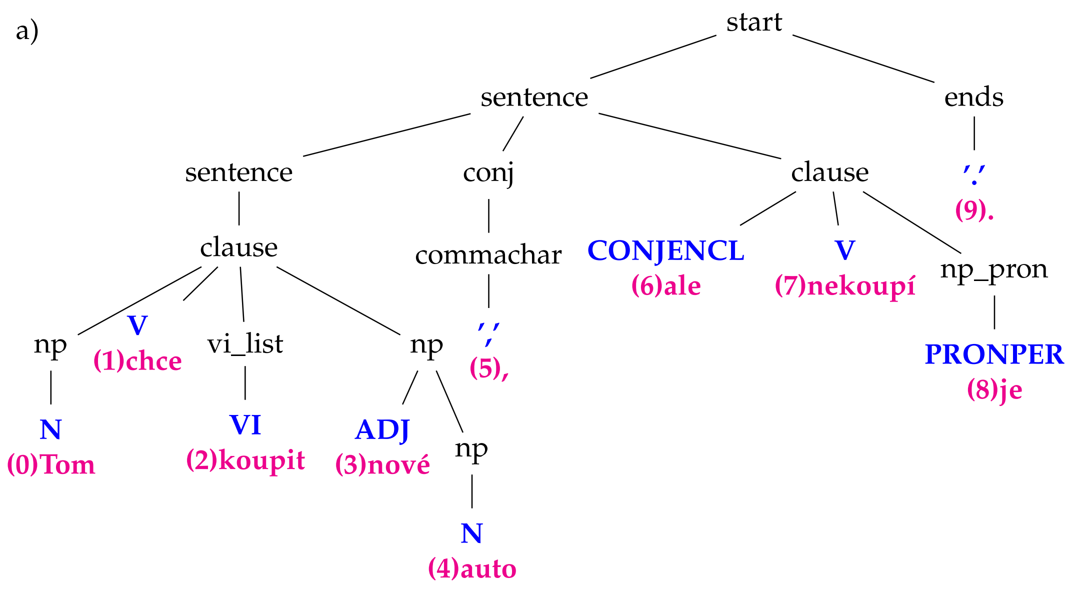
Both `synt` and SET parsers are rule-based systems but they use different approach in providing syntactic analysis, which also influences the differences of their syntactic trees.

2.1 The `synt` parser

The `synt` parser [6,8] was developed around the meta-grammar concept based on traditional linguistic rule systems. The core of the parser uses a context-free grammar with contextual actions and performs a stochastic agenda-based head-driven chart analysis.

The internal representation concentrates on fast processing of very ambiguous syntactic structures. The parser is able to process sentences with syntactic combinations resulting in millions of possible syntactic trees with an average processing time of 0.07 s per sentence. The analysis trees are stored in a *packed shared forest*, which can be transformed to several kinds of output such as an ordered list of syntactic trees, a dependency graph or an annotated list of extracted phrases.

The `synt` parser is also able to perform a TIL-based logical analysis of a sentence as a first implementation of the Normal Translation Algorithm a)



b)

$$\lambda w_1 \lambda t_2 \Big((\exists x_3)(\exists i_4)(\exists i_5)\Big([\mathbf{Does}_{w_1t_2}, i_5, [\mathbf{Imp}_{w_1}, x_3]] \wedge [[\mathbf{nov\acute{y}}, \mathbf{auto}]_{w_1t_2}, i_4] \wedge x_3 = [\mathbf{cht\acute{\imath}t}, i_4]_{w_1} \wedge [\mathbf{Tom}_{w_1t_2}, i_5]\Big) \wedge \Big[\mathbf{Not}, \Big[\mathbf{True}_{w_1t_2}, \lambda w_6 \lambda t_7 (\exists x_8)(\exists i_9)\Big([\mathbf{Does}_{w_6t_7}, On, [\mathbf{Perf}_{w_6}, x_8]] \wedge [\mathbf{on}_{w_6t_7}, i_9] \wedge x_8 = [\mathbf{koupit}, i_9]_{w_6}\Big)\Big]\Big]\Big) \ldots \pi$$

Fig. 1: The synt outputs for the sentence: “Tom chce koupit nové auto, ale nekoupí je.” (Tom wants to buy a new car, but he will not buy it.), a) the syntactic tree, b) the logical construction.

(NTA [9]), see an example of both the synt tree output and the logical construction in Figure 1. The logical analysis in synt is tightly connected to the parsing process and is not directly transferable to other representations, which was the reason for design and implementation of the parser-independent AST tool.

2.2 SET parser

The SET parser[1] [7] is a rule-based parser based on pattern-matching dependency rules. The SET grammar consists of a set of pattern matching specifications that compete with each other in the process of analysis. From the best matches, SET builds a full coverage syntactic dependency tree of the input sentence. Currently, the system includes grammars for Czech, Slovak and English, each with few dozens of rules that sufficiently model the syntax of the particular language, thanks to the expressive character of the formalism.

[1] http://nlp.fi.muni.cz/projects/set

```
TMPL: $ATTR $...* noun  AGREE 0 2 cgn   MARK 0  DEP 2   PROB 6000
    LABEL modifier
    LABEL rule_sch ( $$ $@ "[#1,#2]" )
```

Fig. 2: SET rule for adjective-noun modifier (e.g. "black dog") supplemented with a TIL schema. The `$ATTR` and `noun` aliases are defined in other parts of the grammar. Actions require agreement in case, gender and number (`cgn`), create an *adjective* $\rightarrow$ *noun* dependency and set up heavy weight (`PROB`) for this rule. The TIL schema says that the attribute (#1) is applied as a function to the noun (#2) – these indexes refer directly to the resulting structured tree.

Due to the nature of the parsing process (each word is in the end included into the tree structure), the parser has 100% coverage, i.e. it provides an analysis for any input sentence.

The input format for the parser is a morphologically annotated sentence in the vertical format.[2] The output options include hybrid trees,[3] dependency trees, phrase structure trees, or a "bush" output.[4]

2.3 SET modifications for TIL analysis

The phrase structure output of SET was the most suitable for the TIL analysis, because the current AST system also works with a phrase structure tree. However, as opposed to the `synt` system, there is no formal grammar (in the Chomsky's sense) according to which the phrase structure tree was created; it is just a conversion from the hybrid tree (that was created according to the SET pattern-matching grammar). Especially, it is not clear what types of phrasal sub-trees can appear in the result, e.g. what components can a noun-phrase non-terminal have; in theory, the number of options is unlimited.

This was a problem, as the TIL analysis algorithm processes the tree level by level and it is to these levels (that correspond to grammar rules in the `synt` parser) where the particular schemata for TIL are assigned. Therefore, we have developed a new type of SET phrase structure output[5] where each level of the tree corresponds exactly to one rule in the SET grammar. This adaptation allows to assign the TIL logical construction schemata to be assigned to the rules in the SET grammar.

With this concept in mind, the Czech and English SET grammars[6] were supplemented with the schemata for TIL analysis exported to the new phrase

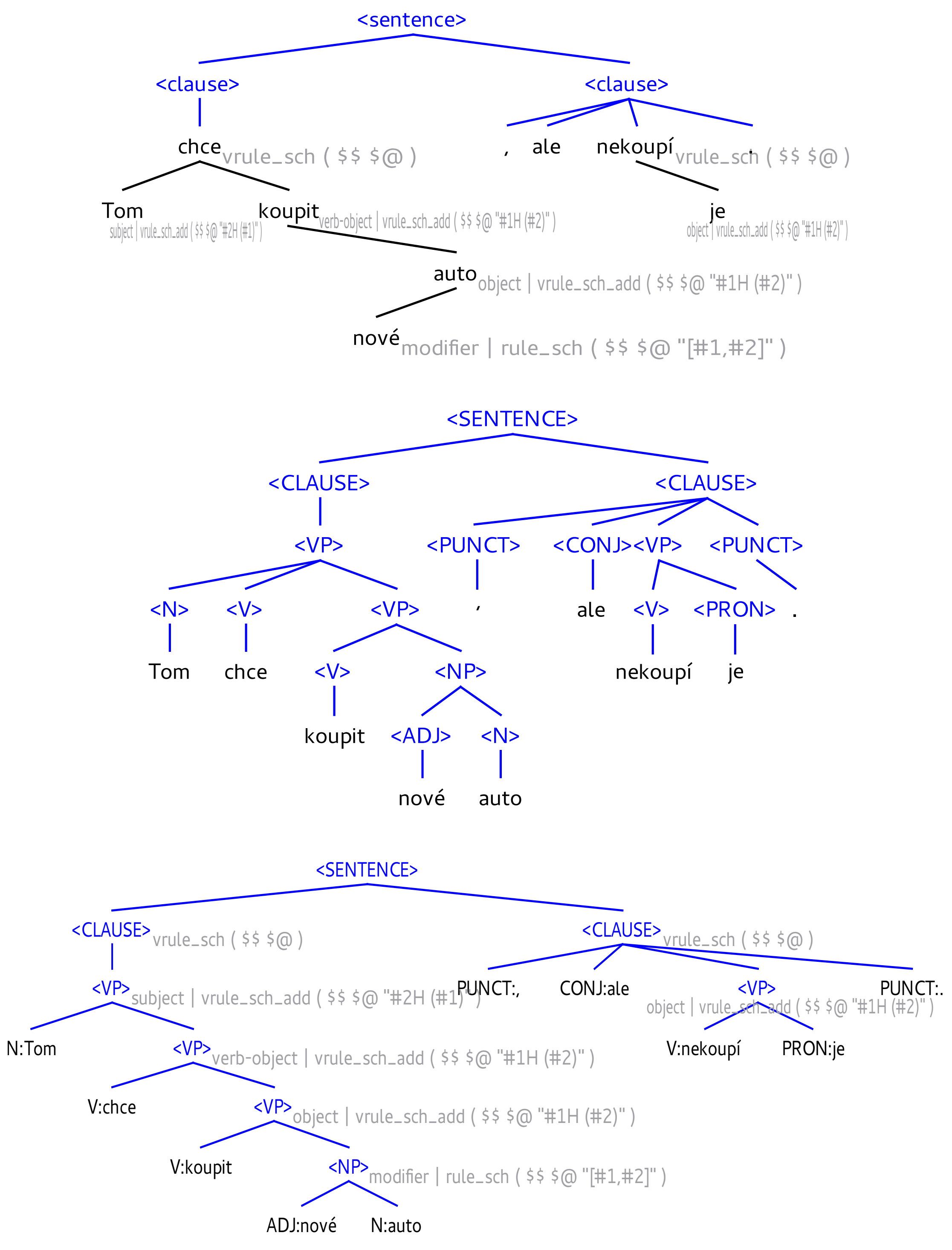
[2] Plain text tabular representation of a tokenized and annotated text.

[3] Syntactic tree format that combines dependency and phrase structure features; this is the primary output of the parser and all the other outputs are based on this tree

[4] A "bush" denotes a structure of phrases connected with dependencies, see [10] for details.

[5] as a new option `-s` which stands for "structured"

[6] denoted as `tilcz.set` and `tilen.set` in the repository



<sentence>
<clause>
<clause>
chce vrule_sch ($$ $@)
, ale nekoupí vrule_sch ($$ $@)
.
Tom subject | vrule_sch_add ($$ $@ "#2H (#1)")
koupit verb-object | vrule_sch_add ($$ $@ "#1H (#2)")
je object | vrule_sch_add ($$ $@ "#1H (#2)")
auto object | vrule_sch_add ($$ $@ "#1H (#2)")
nové modifier | rule_sch ($$ $@ "[#1,#2]")
<SENTENCE>
<CLAUSE>
<CLAUSE>
<VP>
<PUNCT>
<CONJ>
<VP>
<PUNCT>
<N>
<V>
<VP>
,
ale
<V>
<PRON>
.
Tom
chce
<V>
<NP>
nekoupí
je
koupit
<ADJ>
<N>
nové
auto
<SENTENCE>
<CLAUSE> vrule_sch ($$ $@)
<CLAUSE> vrule_sch ($$ $@)
<VP> subject | vrule_sch_add ($$ $@ "#2H (#1)")
PUNCT:,
CONJ:ale
<VP> object | vrule_sch_add ($$ $@ "#1H (#2)")
PUNCT:.
N:Tom
<VP> verb-object | vrule_sch_add ($$ $@ "#1H (#2)")
V:nekoupí
PRON:je
V:chce
<VP> object | vrule_sch_add ($$ $@ "#1H (#2)")
V:koupit
<NP> modifier | rule_sch ($$ $@ "[#1,#2]")
ADJ:nové
N:auto

Fig. 3: Different SET trees for the sentence "Tom chce koupit nové auto, ale nekoupí je." (Tom wants to buy a new car but he will not buy it.) The hybrid tree (1) has TIL schemata assigned to particular nodes (according to the SET grammar), in the default phrasal tree (2), it was not easy to decide where the schemata should be stored, so we introduced a "structured" tree (3) where the schemata are assigned to non-terminals (with the same semantics as in (1)).

structure output of the SET syntactic tree. Figure 2 shows an example of a SET rule annotated with a TIL schema exploited by the AST system. In Figure 3, we can see the difference between the default SET hybrid tree, the default phrase structure tree and the new "structured" tree. TIL schemata assigned to the particular rules are shown in grey within these trees.

As follows from Figures 3 and 1, the structure of the synt and SET trees is different. For example, the clause level in synt trees contains the main verb and the modal verb on the same level; the SET tree uses binary branching. Therefore, the AST tool needed to be customized to work with the SET "structured" trees.

3 The AST Modules

AST is a language independent tool for semantic analysis of natural language sentences. The ideas of the AST implementation emerged from the synt parser while targeting at easy adaptations for new languages and syntactic analyser. AST follows the principles of the NTA algorithm and uses the Transparent Intentional Logic (TIL) for the formalization of semantic constructions.

The AST tool is a modular system and consists of seven main parts, which mostly correspond to the required language-dependent lexicons:[7]

– the *input processor*: reads the input with a syntactic tree in textual form (see an example of SET tree in Figure 4).
– the *grammar processor*: reads the list of grammar rules and schema actions to obtain a schema for each node inside the tree (only for synt parser). An example of such rule is:

```
np -> left_modif np
    rule_schema ( "[#1,#2]" )
```

In this case the resulting logical construction of the *left-hand side* np is obtained as a (logical) application of the left_modif (sub)construction to the *right-hand side* np (sub)construction.
– the *lexical items processor*: a list of lexical item specifications with TIL types for each leaf (word) in the tree structure. A lexical item example for the verb "koupit" (buy) is:

koupit
/k5/otriv $(((o(oo_{\tau\omega})(oo_{\tau\omega}))\omega)\iota)$

– the *schema processor*: general module for operations over the logical construction schemata. An example of creating the construction for the noun phrase "nové auto" (new car):

rule_schema: '[#1,#2]'
Processing schema with params:
    #1: ^{0}nový... $((o\iota)_{\tau\omega}(o\iota)_{\tau\omega})$
    #2: ^{0}auto... $(o\iota)_{\tau\omega}$
Resulting constructions:
[^{0}nový/$((o\iota)_{\tau\omega}(o\iota)_{\tau\omega})$, ^{0}auto/$(o\iota)_{\tau\omega}$]... $(o\iota)_{\tau\omega}$

[7] See [5] for details.

– the *verb valency processor*: identifies the correct valency frame for the given sentence and triggers the schema parser on sub-constructions according to the schema coming with the valency. An example for the verb "koupit" (eat) is as follows

```
koupit
hPTc4 :exists:V(v):V(v):and:V(v)=[[#0,try(#1)],V(w)]
```

– the *prepositional valency expression processor*: translates analysed prepositional phrases to possible valency expressions. For instance, the record for the preposition "k" (to) is displayed as

```
k
3 hA hH
```

The preposition "k" (to) can introduce a prepositional phrase of a *where-to* direction `hA`, or a modal *how/what* specification `hH`.

– the *sentence schema processor*: if the sentence structure contains subordination or coordination clauses the sentence schema parser is triggered. The sentence schemata are classified by the conjunctions used between clauses. An example for the conjunction "ale" (but) is:

```
("";"ale") : "lwt(awt(#1) and awt(#2))"
```

The resulting construction builds a logical conjunction of the two clauses as can be seen in the Figure 1 example.

3.1 AST Adaptations for the SET Trees

As explained above in Section 2.3, the dependency core of the SET parser defies to the logical schema prescription used in the case of the `synt` parser. The final solution implemented in AST is based on two ideas:

– organization of the SET output in an adapted phrasal-dependency tree, viz the "structured" tree in Figure 3,
– splitting the corresponding *verb rule schemata* to binary additive actions denoted as `vrule_sch_add`.

An example of the resulting labeled tree form is displayed in Figure 4, where each constructive edge obtains a logical schema label from the SET grammar.

In the `synt` tree, all constituent groups are reachable directly from the clause node (see Figure 1). In the SET tree, the additive actions `vrule_sch_add` build the arguments needed by the clause-level `vrule_sch` (*verb rule schema*) action. The action arguments denote different constituent types: a main verb, an auxiliary constituent or a clause constituent. The schemata also distinguish whether the reference denotes the whole (sub)phrase or applies to the headword only (tag `H`). By this process, the identified constituents are unified with the list of

```
id word:nterm lemma  tag          pid til schema
0  N:Tom      Tom    k1gMnSc1;ca14 p
1  V:chce     chtít  k5eAaImIp3nS 15
2  V:koupit   koupit k5eAaPmF     16
3  ADJ:nové   nový   k2eAgNnSc4d1 17
4  N:auto     auto   k1gNnSc4     17
5  PUNCT:,    ,      kIx          10
6  CONJ:ale   ale    k8xC         10
7  V:nekoupí  koupit k5eNaPmIp3nS 13
8  PRON:je    on     k3xPp3gNnSc4 13
9  PUNCT:.    .      kIx.         10
10 <CLAUSE>          k5eNaPmIp3nS 12  vrule_sch ( $$ $@ )
11 <CLAUSE>          k5eAaImIp3nS 12  vrule_sch ( $$ $@ )
12 <SENTENCE>                     -1
13 <VP>       koupit k5eNaPmIp3nS 10  vrule_sch_add ( $$ $@ "#1H (#2)" )
14 <VP>       chtít  k5eAaImIp3nS 11  vrule_sch_add ( $$ $@ "#2H (#1)" )
15 <VP>       chtít  k5eAaImIp3nS 14  vrule_sch_add ( $$ $@ "#1H (#2)" )
16 <VP>       koupit k5eAaPmF     15  vrule_sch_add ( $$ $@ "#1H (#2)" )
17 <NP>       auto   k1gNnSc4     16  rule_sch ( $$ $@ "[#1,#2]" )
```

Fig. 4: SET tree format

possible verb valency frames, which finally leads to the selected clause logical schema.

On the sentence level, where the TIL constructions of all clauses are combined into the final construction, the synt grammar specified action for *sentence rule schema*, which consults the sentence schema lexicon to determine the construction schema. In case of SET, there is no top-level rule, that is why AST needs to automatically trigger the function for a recursive combination of clauses in the SET tree output.

4 Bilingual Logical Analysis

As we have mentioned above, the SET parser contains comparable grammars for the Czech and English languages which in combination with the above described modifications of both SET and AST systems enables bilingual logical analysis. The complexity of the two grammars is very similar – the Czech grammar currently contains 71 rules, the English one has 45 rules. Both the grammars were annotated with TIL schemata consistently.

The English language processing in AST is, however, still preliminary in the sense that the required logical lexicons are filled with testing data only. The example logical constructions obtained from translated sentences show promising regularities in the construction structures, which allow for isomorphic concept labeling between the two languages. An example of the

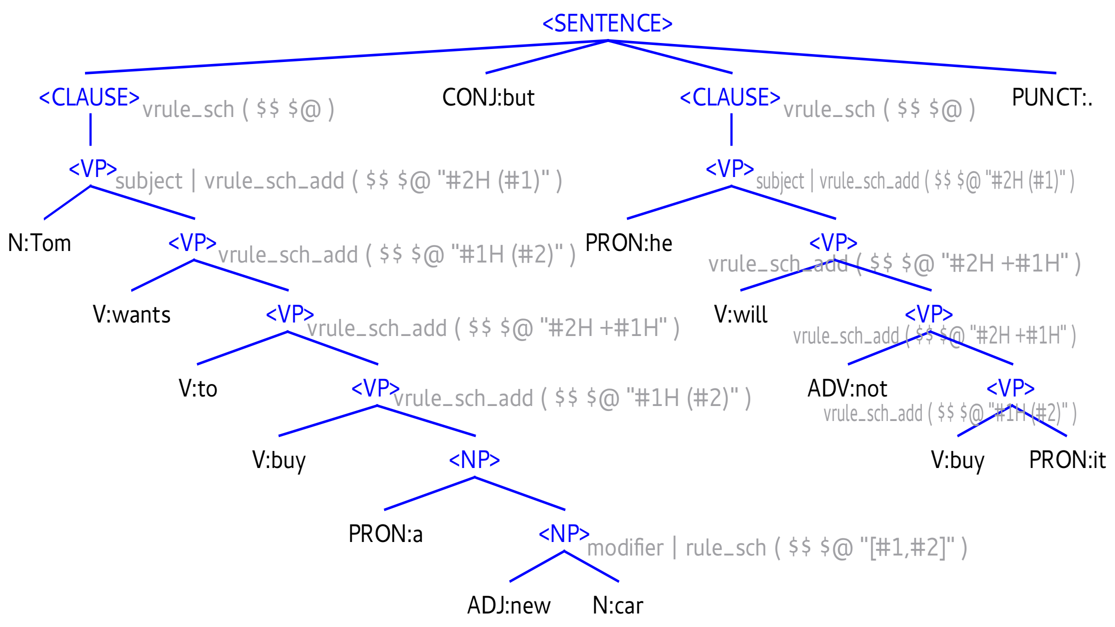

<SENTENCE>
<CLAUSE> vrule_sch ($$ $@)
CONJ:but
<CLAUSE> vrule_sch ($$ $@)
PUNCT:.
<VP> subject | vrule_sch_add ($$ $@ "#2H (#1)")
<VP> subject | vrule_sch_add ($$ $@ "#2H (#1)")
N:Tom
<VP> vrule_sch_add ($$ $@ "#1H (#2)")
PRON:he
<VP> vrule_sch_add ($$ $@ "#2H +#1H")
V:wants
<VP> vrule_sch_add ($$ $@ "#2H +#1H")
V:will
<VP> vrule_sch_add ($$ $@ "#2H +#1H")
V:to
<VP> vrule_sch_add ($$ $@ "#1H (#2)")
ADV:not
<VP> vrule_sch_add ($$ $@ "#1H (#2)")
V:buy
<NP>
V:buy
PRON:it
PRON:a
<NP> modifier | rule_sch ($$ $@ "[#1,#2]")
ADJ:new
N:car

Fig. 5: SET tree for English sentence “Tom wants to buy a new car but he will not buy it.”, according to SET grammar for English, and annotated with TIL schemata.

SET tree for the English variant of the original Czech sentence is displayed in Figure 5, which leads to the logical construction of

$$\begin{aligned}
&\lambda w_1 \lambda t_2 \Big((\exists x_3)(\exists i_4)(\exists i_5) \Big([\mathbf{Does}_{w_1 t_2}, i_5, [\mathbf{Imp}_{w_1}, x_3]] \wedge [[\mathbf{new}, \mathbf{car}]_{w_1 t_2}, i_4] \\
&\wedge\ x_3 = [\mathbf{to_want}, i_4]_{w_1} \wedge [\mathbf{Tom}_{w_1 t_2}, i_5] \Big) \wedge \Big[\mathbf{Not}, \Big[\mathbf{True}_{w_1 t_2}, \\
&\lambda w_6 \lambda t_7 (\exists x_8)(\exists i_9) \Big([\mathbf{Does}_{w_6 t_7}, He, [\mathbf{Perf}_{w_6}, x_8]] \wedge [\mathbf{it}_{w_6 t_7}, i_9] \\
&\wedge\ x_8 = [\mathbf{to_buy}, i_9]_{w_6} \Big) \Big] \Big] \Big) \ldots \pi
\end{aligned}$$

The resulting construction can be directly compared with the TIL logical construction from Figure 1b).

5 Conclusions and Future Directions

Representing the essence of the structural meaning via an automated process offers a valuable tool for semantic processing of natural language texts. The validity of such representation can be verified with the level of correspondence in the resulting logical formulae when processing direct translations between two natural languages.

In this paper, we have presented the current development of the language-independent automated semantic tool AST, which shows the capabilities of logical analysis for two languages – Czech and English. Even though the

English analysis currently does not cover standard vocabulary on the same level as the Czech analysis, the preliminary results are promising in the structural correspondences between the logical representations.

In the future research, the required English TIL lexicons will be connected to large available resources such as VerbNet, FrameNet and WordNet, which will allow to obtain the same level of coverage for both languages. The logical correspondences will be then thoroughly evaluated on a representative set of bilingual sentence pairs.

Acknowledgments. This work has been partly supported by the Czech Science Foundation under the project GA15-13277S.

References

1. Anismovich, K., Selegey, V., Zuev, K.: Systems for translating sentences between languages using language-independent semantic structures and ratings of syntactic constructions (July 3 2012) US Patent 8,214,199.
2. Popel, M., Žabokrtský, Z.: TectoMT: Modular NLP Framework. In: International Conference on Natural Language Processing, Springer (2010) 293–304
3. Tichý, P.: The foundations of Frege's logic. Walter de Gruyter, New York (1988)
4. Duží, M., Jespersen, B., Materna, P.: Procedural Semantics for Hyperintensional Logic. Foundations and Applications of Transparent Intensional Logic. Volume 17 of Logic, Epistemology and the Unity of Science. Springer, Berlin (2010)
5. Medveď, M., Horák, A., et al.: AST: New Tool for Logical Analysis of Sentences based on Transparent Intensional Logic. (2015)
6. Horák, A.: Computer Processing of Czech Syntax and Semantics. Librix.eu, Brno, Czech Republic (2008)
7. Kovář, V., Horák, A., Jakubíček, M.: Syntactic analysis using finite patterns: A new parsing system for Czech. In: Human Language Technology. Challenges for Computer Science and Linguistics. Volume 6562 of Lecture Notes in Computer Science., Berlin, Springer (2011) 161–171
8. Jakubíček, M., Horák, A., Kovář, V.: Mining phrases from syntactic analysis. In: Text, Speech and Dialogue. (2009) 124–130
9. Horák, A.: The Normal Translation Algorithm in Transparent Intensional Logic for Czech. PhD thesis, Faculty of Informatics, Masaryk University, Brno (2002)
10. Grác, M.: Case study of BushBank concept. In: The 25th Pacific Asia Conference on Language, Information and Computation, Singapore, Institute for Digital Enhancement of Cognitive Development, Waseda University (2011) 353–361