AST: New Tool for Logical Analysis of Sentences based on Transparent Intensional Logic

Marek Medveď and Aleš Horák

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

Abstract. Logical analysis of natural language is able to extract semantic relations that follow the underlying logical formalism. Transparent Intensional Logic (TIL) has been designed to capture even high-order relations between sentence elements and systematically work with all kinds of language references, i.e. extensions, intensions and hyperintensions.
In this paper, we introduce the first version of a new tool, called AST, for automatic semantic analysis of sentence. This tool is based on the TIL logic processing rules as they were implemented in the SYNT parser, in its logical analysis module. AST thus shares lexicons and semantic rules in the same format as in the SYNT parser, but allows to build upon the output of other syntactic parsers. AST is designed as a universal semantic analysis tool, which strictly separates the application logic and input data and strives for language independent analysis.
Within the evaluation, we present preliminary results of testing AST on selected problematic phenomena, which were not correctly processed by the SYNT logical analysis.

Keywords: semantics; semantic analysis; logical analysis; Transparent Intensional Logic; TIL

1 Introduction

Full semantic analysis of natural language (NL) texts still remains an open problem, although the problem is being partially solved from different points of view. The most comprehensive semantic systems build upon a mathematically sound formalism of a selected logical system. Mostly due to computability and efficiency, current systems work with the first order logic (or its variant). However, the low-order logic is not appropriate for capturing higher-order phenomena that occurs in natural language, such as belief attitudes, direct speech, or verb tenses [6].

In the following text, we present a new tool for automatic semantic analysis (AST) that emerged from (a module of) the Czech syntactic parser SYNT [3]. AST is now available as a standalone tool that is independent from the SYNT parser. AST works with the same input files (lexicons, semantic rules, ...) that

```
<tree>
{##start##
  {start
    {ss
      {clause
        {VL<leaf><idx>0</idx><w>Jedl</w>
         <l>jíst</l><c>k5eAaIgMnS</c></leaf>}
        {intr
          {adjp
            {ADJ<leaf><idx>1</idx><w>pečené</w>
             <l>pečený</l><c>k2eAgNnSc4</c></leaf>}
          }
          {np
            {N<leaf><idx>2</idx><w>kuře</w>
             <l>kuře</l><c>k1gNnSc4</c></leaf>}
          }
        }
      }
    }
    {ends
      {'.'<leaf><idx>3</idx><w>.</w><l>.</l><c>kX</c></leaf> }
    }
  }
}
</tree>
```

Fig. 1. Syntactic tree – text markup for "Jedl pečené kuře." (He ate a roasted chicken.)

were designed and developed in SYNT. AST can thus provide the semantic analysis in the form of Transparent Intensional Logic (TIL) constructions [1] independently on the input syntactic parser and language. Processing new language thus consists in a specification of four lexicon files that describe lexical items, verb valencies, prepositional valencies and a semantic grammar.

In the following sections, we describe the structure of the system, the language dependent files and the form of required input as processed by a syntactic parser.

2 The AST System

In this section, we introduce the main parts of the AST system, describe the content of language dependent files and formalize the required input of AST.

2.1 The AST Input

To create a semantic structure of a sentence, AST needs the output from previous NL analysis levels. A usual output is in the form of a syntactic tree

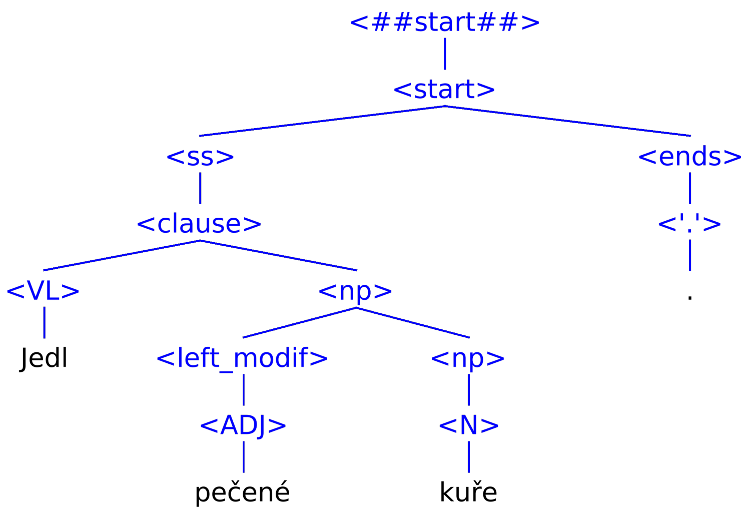

Fig. 2. Syntactic tree – visual for "Jedl pečené kuře." (He ate a roasted chicken.)

as provided by a syntactic parser. See Figure 1 for an example of a syntactic tree that is accepted as the AST input. The corresponding graphical tree representation is in Figure 2.

Besides the tree nodes and edges, the tree contains morphological information about each word: a lemma and a PoS tag [4], which are used by AST for deriving implicit out-of-vocabulary type information.

2.2 Language Dependent Files

The AST system itself is universal and can be used for semantic analysis of any language. However the main system core also uses input files that are language dependent and that need to be modified for addition of another language. In this section, we describe the format of those files needed to build the resulting logical construction.

The Semantic Grammar The resulting semantic construction is built by bottom-up analysis based on the input syntactic tree provided by the syntactic parser and by a semantic extension of the actual grammar used in the parsing process. To know which rule was used by the parser, AST needs the semantic grammar file. This file contains specification of semantic actions that need to be done before propagation of particular node constructions to the higher level in the syntactic tree. The semantic actions define what logical functions correspond to each particular syntactic rule. For instance, the `<np>` node in Figure 1 corresponds to the rule and action:

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

```
rule_schema: 2 nterms, '[#1,#2]'
1, 3, +np -> . left_modif  np . @level 0
  nterm 1: 1, 2, +left_modif -> . left_modifnl  . @level 0, k2eAgNnSc4
       TIL: $^0$pečený...$((o\iota)_{\tau\omega}(o\iota)_{\tau\omega})$
  nterm 2: 2, 3, +np -> . N  . @level 0, k1gNnSc4
       TIL: $^0$kuře...$(o\iota)_{\tau\omega}$

Processing schema with params:
    #1: $^0$pečený...$((o\iota)_{\tau\omega}(o\iota)_{\tau\omega})$
    #2: $^0$kuře...$(o\iota)_{\tau\omega}$
Resulting constructions:

    [$^0$pečený/$((o\iota)_{\tau\omega}(o\iota)_{\tau\omega})$, $^0$kuře/$(o\iota)_{\tau\omega}$]...$(o\iota)_{\tau\omega}$
```

Fig. 3. Analysis of the expression "pečené kuře" (roasted chicken).

which says that 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. An example of processing such grammar rule is in Figure 3.

TIL Types of Lexical Items The second language dependent file defines lexical items and their TIL types. The types are hierarchically built from four simple TIL types [2]:

- o: representing the truth-values,
- ι: class of individuals,
- τ: class of time moments, and
- ω: class of possible worlds.

AST contains rules for deriving implicit types based on PoS tags of the input words, so as the lexicons must prescribe the type only for cases that differ from the implicit definition. A lexical item example for the verb "jíst" (eat) is:

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

The exact format of the lexical item in the input file is as follows: the lemma starts on a separate line. After the lemma there is a list of lines where an (optional) POS tag filter precedes the resulting object schema (here `otriv`, i.e. *o-trivialisation*) and TIL type (here *verbal object with one ι-argument*).

Verb Valencies The next language dependent file is a file that defines verb valencies and schema and type information for building the resulting construction from the corresponding valency frame. An example for the verb "jíst" (eat) is as follows

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

This record defines the valency of <*somebody*> *eats* <*something*>, given by the *brief valency frame* `hPTc4` of the *object* (an animate or inanimate noun phrase in accusative), and the resulting construction of the *verbal object* (`V(v)`) derived as an application of the *verb* (`#0`) to its argument (the sentence object) with possible extensification (`try(#1)`) and the appropriate possible world variable (`V(w)`).

Prepositional Valency Expressions The last file that has to be specified for each language is a list of semantic mappings of prepositional phrases to valency expressions based on the head preposition. The file contains for each combination of a preposition and a grammatical case of the included noun phrase all possible valency slots corresponding to the prepositional phrase. For instance, the record for the preposition "k" (to) is displayed as

```
k
3 hA hH
```

saying that "k" can introduce prepositional phrase of a *where-to* direction `hA` (e.g. *"k lesu"* – "to a forest"), or a modal *how/what* specification `hH` (e.g. *"k večeři"* – "to a dinner").

2.3 System parts

The AST system is implemented in the Python 2.7 programming language and consists of six main parts:

- the *input parser*: reads standard input, extracts tree structures and creates tree object for each tree from input,
- the *grammar parser*: reads the grammar file and assigns a grammar rule and appropriate actions to each node inside the tree,
- the *lexical item parser*: reads the file with lexical item schemata and TIL types and assigns the type to each leaf in the tree structure,
- the *schema parser*: according to a logical construction schema coming with a semantic action, this module creates a construction from sub-constructions,
- the *verb valency parser*: picks up the correct valency for given sentence and triggers the schema parser on sub-constructions according to the schema coming with the valency, and
- the *prepositional valency expression parser*: reads the possible valency expressions assigned to prepositional phrases used as (optional) valency slots in the actual sentence valency frame.

3 Error Analysis

During the AST development, AST is continuously evaluated in comparison with the original SYNT TIL logical analysis. In these tests, a number of uncovered phenomena is described and implemented in AST. The number of NL constructs covered by the AST logical analysis is thus still growing and, already in the current version, surpasses the original logical analysis.

In the following paragraphs, we present the results of error analysis of selected 200 sentences and the corresponding problems related to the original analyses of these sentences.

3.1 Sentences with Two Items Divided by "and"

The construction for the following sentence:

Vidíte zásadnější rozdíly mezi přístupy českých a západních informačních firem?
(Can you see the main difference between Czech and west information companies?)

is (schematically) analysed as `[západní, [český, x8]] and [firma, x8]`.[1] This means that both words "*západní*" (west) and "*český*" (Czech) are modifiers of the word "firma" (company) which is not correct. The correct analysis for this sentence is `[západní, x8] and [firma,x8] and [český, x9] and [firma, x9]`.[2]

3.2 Verb Valency vs. Clause Valency

The concluding clause semantic construction is created according to the content of the syntactic tree of the clause. The clause valency (valencies) is built from the subtrees, and in the next step the clause valency is matched to the verb valency from lexicon and the schema assigned to the verb valency is used for creating the clause logical construction. However, if the system creates an incorrect clause valency or the verb valency has no suitable option, the clause construction is not created and the resulting semantic analysis is void.

In the analysis of the following sentence

Možná, že se tito lidé ani nesetkali.
(Maybe, these people never met each other.)

the clause schema does not contain the reflexive pronoun "*se*"[3], so the system finds only the verb valency for the word "*setkat*" (meet), which does not contain suitable schema for semantic analysis.

[1] transl. `[west, [Czech, x8]] and [company, x8]`.
[2] transl. `[west, x8] and [company,x8] and [Czech, x9] and [company, x9]`.
[3] here in the meaning of "each other"

Table 1. 200 sentences evaluated by the SYNT TIL system and the AST system.

system	correct	correct in %	incorrect	incorrect in %
SYNT TIL	131	65.5 %	69	35.5 %
AST	158	**79.0 %**	42	**21.0 %**

3.3 Verb Valency Schema Update

In some cases, the clause valency is created correctly but the verb valency file does not contain an option that can match with the created cause valency. To solve this type of problems, AST files must be updated to add new option for the missing verb valency to create the correct semantic analysis.

3.4 Verb Valency Schema Missing

The verb valency list is created from the Czech VerbaLex lexicon [5], which is a large database of more than 10,000 Czech verb lemmata and their verb frames. However, it can happen that some verbs are not included in VerbaLex thus the system does not contain the verb frame.

In the test data, there were three verbs that are not included in VerbaLex, that it why they have been added to the AST verb list.

3.5 System errors

The previous SYNT TIL analysis contains a construction checker that does not allow the dash character "-" in the name of an object construction, such as `[0(Si-an/`ι`)]` (a proper name). In such case, the construction is rejected and the semantic analysis fails. The AST analysis allows such object naming, so the analysis can continue successfully.

4 Evaluation

In this section, the AST system is compared with the original SYNT TIL logical module. For evaluation, we have picked up 200 randomly chosen sentences that were processed by both systems and the resulting analysis for each sentence was manually checked. For the final results, that shows 14 % improvement of AST to SYNT TIL, see Table 1.

5 Conclusions

In this paper, we have introduced new language and parser independent tool for semantic analysis, called AST, that is based on the SYNT TIL logical analysis. The new AST system is designed as a lightweight standalone module, that can be straightforwardly updated and improved. Already in the first version AST corrects several frequent errors of its predecessor, and presents a 14 % increase in the number of correctly analyzed sentences.

This new implementation of semantic analysis brings the necessary simplicity for future development and parser and language independence.

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

References

1. Duží, Marie and Jespersen, Bjorn and Materna, Pavel, Procedural Semantics for Hyperintensional Logic: Foundations and Applications of Transparent Intensional Logic, Springer Science & Business Media (2010)
2. Horák, Aleš: Types in Transparent Intensional Logic and Easel – a Comparison, Proceedings of the IASTED International Conference Artificial Intelligence and Applications, 833–837 (2004)
3. Horák, Aleš and Jakubíček, Miloš and Kovář, Vojtěch: Linguistic Logical Analysis of Direct Speech, RASLAN 2012 Recent Advances in Slavonic Natural Language Processing, 51–59 (2012)
4. Jakubíček, Miloš and Kovář, Vojtěch and Šmerk, Pavel: Czech Morphological Tagset Revisited, Proceedings of Recent Advances in Slavonic Natural Language Processing, 29–42 (2011)
5. Nevěřilová, Zuzana and Grác, Marek : Common Sense Inference using Verb Valency Frames, In Proceedings of 15th International Conference on Text, Speech and Dialogue, 328–335 (2012)
6. Tichý, Pavel: The Foundations of Frege's Logic. de Gruyter, Berlin, New York (1988)