THE AUTOMATED TRANSLATION OF NATURAL LANGUAGE SENTENCES INTO INTENSIONAL LOGIC AT THE TYPE ANALYSIS AND CONSTRUCTION SYNTHESIS LEVELS

Branislav BEDNÁR*, Zuzana BILANOVÁ*, Adrián PEKÁR**
*Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Košice,
Letná 9, 042 00 Košice, tel. +421 55 602 2661
E-mails: branislav.bednar@student.tuke.sk, zuzana.bilanova@tuke.sk
**Department of Networked Systems and Services, Budapest University of Technology and Economics
1111 Budapest, Megyetem rkp. 3., Hungary, tel. +36 1 463 1111, E-mail: apekar@hit.bme.hu

ABSTRACT

This paper describes the theoretical design and implementation of the semantic machine of transparent intensional logic. Transparent intensional logic is used for the logical-semantic analysis of natural language, which is performed in three steps - type analysis, creation of construction, and type control (last step is optional). The semantic machine described in this paper allows analyzing natural language sentences in the first two of the mentioned steps. Transparent intensional logic analyzes the meaning of natural language sentences and therefore it was not essential to focus on the syntactic level of analysis. Syntactic analysis is performed using a Stanford CoreNLP parser, which was carefully selected after comparison with competing parsers. The functionality of the implemented semantic machine is demonstrated by a logical analysis of examples of natural language sentences.

Keywords: *intensional constructions, intensional logic, natural language processing, Stanford CoreNLP, semantic machine, syntactic analyzer*

1. INTRODUCTION

Czech logician Pavel Tichý created foundations of transparent intensional logic (TIL) in 1961 [1]. TIL uses an updated version of the object-oriented λ-calculus which has a ramified hierarchy of types. Tichý defined the TIL object base to analyze natural language expressions. Object base contains a set of individuals ι, a set of truth values o, a set of possible worlds ω and a set of time points τ [2]. $\{\iota, o, \omega, \tau\}$ objects represent a TIL intensional base. The intensions are objects of type $((\alpha\tau)\omega)$, defined as functions from possible worlds in time points for any type α.

A meaning of expression is expressed by its construction in TIL. Constructions are abstract, algorithmically structured procedures which represent the meaning of natural language expressions. Types of constructions [3]:

- variable x v-construct a construction, where v is a valuation parameter,
- trivialization $\hat{}0A$ cunstruct A without any change,
- composition $[AA]$ construct the application of a function to arguments,
- closure λxA construct a function.

The competitive logic system for TIL is Montague intensional logic (MIL) [4]. MIL has several drawbacks that TIL had overcome [5]. TIL is an effective tool for logical analysis of natural language because of its extraordinary expressive power [6], which makes it a suitable basis for the implementation of a semantic machine.

2. SYNTACTIC PARSERS

Analyze of the natural language sentences has two levels - syntactic and semantic [7]. To implement a semantic machine, it is necessary to first perform parsing of a sentence and, based on its output, to examine the meaning of a sentence by using semantic analysis [8]. The TIL principles are applied at the semantic level, so there was no need to implement a custom parser. We decided to use an existing tool for the syntactic analysis. Two of the most commonly used syntactic analyzers - Link Grammar Parser and Stanford CoreNLP, were compared to select the most appropriate tool as the basis of the semantic machine.

2.1. Link Grammar Parser

Link Grammar Parser [9] represents a syntax parser of the English language. This analyzer is written in C programming language and it is entirely open-source. The size of its verbal equipment is more than 60 000 words. In Figure 1 is an output from Link Grammar Parser for the sentence *„John is an American singer."*.

The output gives the display and naming of the links between individual words within the analyzed sentence. Each word of the tree is assigned to some word class. The unpleasant drawback of Link Grammar Parser is that the output is only in plain text form. For the programmer, it is hard to get some results for future semantic analysis.

```
        +---------Ost--------+
        |   +-------Ds-------+
  +-Ss-+   |       +----AN---+
  |    |   |       |         |
John is.v an American.n singer.n

Constituent tree:

(S (NP John)
   (VP is
       (NP an American singer)))
```

Fig. 1 Analysis of the sentence *„John is an American singer."* generated by Link Grammar Parser. The meaning of the used symbols is explained at [9].

2.2. Stanford CoreNLP

Stanford CoreNLP (SCN) [10] appears as a more comprehensive and advanced open-source parser. SCN is an annotation-based natural language processing framework. In addition to English, this analyzer also supports other languages, such as German, Spanish, French, Arabic or Chinese. SCN is written in Java programming language. The great advantage that SCN offers is its interfaces. Thanks to them, it is possible to use SCN with programming languages as C#, Python or JavaScript and it is also cooperative with Docker. The parser can read a single sentence, as well as multiple sentences in a block of the text as an input. If the block of the text is used as an input, then every sentence is analyzed separately and SCN returns a package of analyzed sentences in the output.

Based on our analysis, SCN gives more possibilities to further data processing than Link Grammar parser (shown in Figure 2). SCN returns the result of sentence analysis in json, which is comfortable to parse for the programmer.


```
"tokens":[{
        "after":" ",
        "before":"",
        "characterOffsetBegin":0,
        "characterOffsetEnd":4,
        "index":1,
        "lemma":"John",
        "originalText":"John",
        "pos":"NNP",
        "word":"John"},
      { "after":" ",
        "before":" ",
        "characterOffsetBegin":5,
        "characterOffsetEnd":7,
        "index":2,
        "lemma":"be",
        "originalText":"is",
        "pos":"VBZ",
        "word":"is"},
      { "after":" ",
        "before":" ",
        "characterOffsetBegin":8,
        "characterOffsetEnd":10,
        "index":3,
        "lemma":"a",
        "originalText":"an",
        "pos":"DT",
        "word":"an"},
      { "after":" ",
        "before":" ",
        "characterOffsetBegin":11,
        "characterOffsetEnd":19,
        "index":4,
        "lemma":"american",
        "originalText":"American",
        "pos":"JJ",
        "word":"American"},
      { "after":"",
        "before":" ",
        "characterOffsetBegin":20,
        "characterOffsetEnd":26,
        "index":5,
        "lemma":"singer",
        "originalText":"singer",
        "pos":"NN",
        "word":"singer"},
      { "after":"",
        "before":"",
        "characterOffsetBegin":26,
        "characterOffsetEnd":27,
        "index":6,
        "lemma":".",
        "originalText":".",
        "pos":".",
        "word":"."
      }
    ]

"basicDependencies":[
      {
        "dep":"ROOT",
        "dependent":5,
        "dependentGloss":"singer",
        "governor":0,
        "governorGloss":"ROOT"
      },
      {
        "dep":"nsubj",
        "dependent":1,
        "dependentGloss":"John",
        "governor":5,
        "governorGloss":"singer"
      },
      {
        "dep":"cop",
        "dependent":2,
        "dependentGloss":"is",
        "governor":5,
        "governorGloss":"singer"
      },
      {
        "dep":"det",
        "dependent":3,
        "dependentGloss":"an",
        "governor":5,
        "governorGloss":"singer"
      },
      {
        "dep":"amod",
        "dependent":4,
        "dependentGloss":"American",
        "governor":5,
        "governorGloss":"singer"
      },
      {
        "dep":"punct",
        "dependent":6,
        "dependentGloss":".",
        "governor":5,
        "governorGloss":"singer"
      }
    ]
```


Fig. 2 Analysis of the sentence *„John is an American singer"* generated by SCN. The meaning of *the tokens*, *basic Dependencies* and *enhancedDependencies* is explained at [10].

Table 1 shown the comparison of Link Grammar parser and SCN. Based on it, we decide to choose SCN as the foundation of the TIL semantic machine implementation.

Table 1 Comparison of Link Grammar parser and SCN [11]

	Link Grammar parser	Stanford CoreNLP
API	-	+
multilanguage	-	+
open source	+	+
editable	+	+
output visualization	-	+
last released version	2005	2018
parser configuration	-	+
text analysis	-	+
output format configuration	-	+

2.3. Basic dependencies of SCN

The use of a third-party syntax parser requires an understanding of the output data before further processing. The most important data (captured in basic dependencies) are explained [12] in the following list:

- conj - conjunct,
- cop - copula,
- nsubj - nominal subject,
- det - determiner,
- dobj - direct object,
- neg - negation modifier,
- amod - adjectival modifier,
- advmod - adverb modifier,
- aux - auxiliary,
- poss - possession modifier,
- nmod - nominal modifier,
- nsubjpass - passive nominal subject.

3. TIL SEMANTIC MACHINE

The semantic analysis of sentences is the second step of the logical analysis of natural language. The output of TIL semantic analysis is a sentence represented by TIL construction.

Before our semantic machine implementation, there was one similar solution created by Czech logician Aleš Horák [13]. His semantic machine works with the Czech language and covers almost the entire conceptual equipment [14]. The semantic machines of TIL (Horák and also ours) execute natural language processing in three successive steps (shown in Figure **??**):

1. type analysis - to each object of the sentence is assigned corresponding type,
2. synthesis of construction - construction describes the meaning of an analyzed sentence,
3. type control - proving if the earlier steps were precisely made (voluntary part).

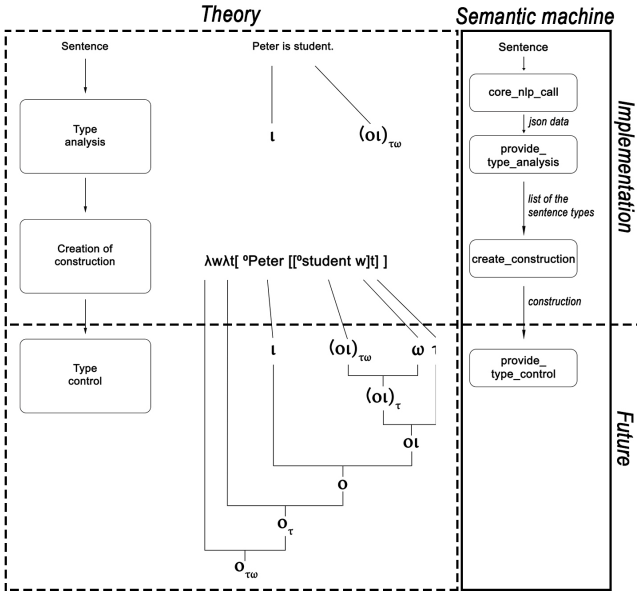

Fig. 3 Three-step sentence analysis in TIL [11].

The left side of Figure 3 represents a formal analysis of the sentence in TIL. The right side captures input processing. The input sentence/block of sentences is at the beginning sent to SCN. SCN provides syntactic analysis and returns analyzed text in json form. In the module *provide_type_analysis* of the semantic machine, a type analysis is made. All the identified types of objects are saved into a list, which should be passed to the module *create_construction*. Successfully generated construction is printed on the output of the semantic machine. Finally, the necessary data are provided to the module *provide_type_analysis*.

3.1. Automated type analysis

The type analysis is the first step of the semantic machine implementation. The analyzed types extend basic TIL types o, ι, ω, τ.

Individual - one of the basic types in TIL. It always starts with a capital. SCN indicates individual as *NNP* - a proper noun (Figure 4, part a).

Truth value - one of the basic types in TIL. In the sentence *„Sun is shining."* word *shining* expresses truth value because it describes the state of the sun in some possible world and time point. The truth value can be also achieved as an independent element of the nominal subject *nsubj* (Figure 4, part b).

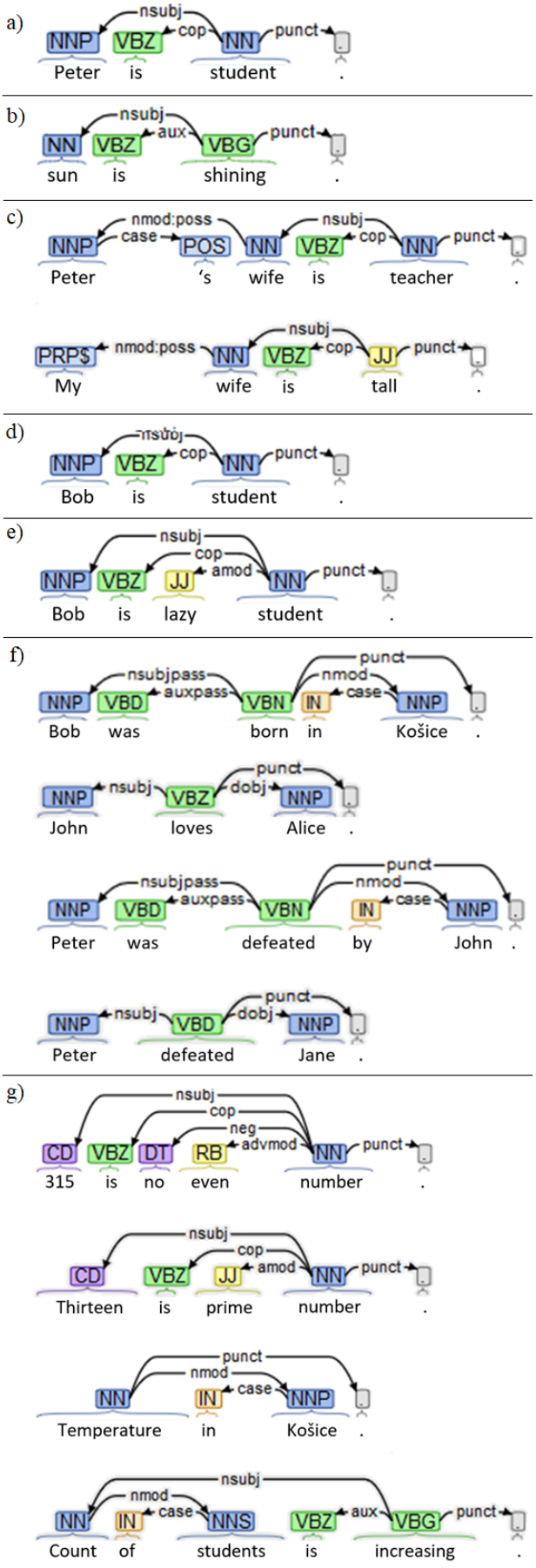

Fig. 4 Assignment of types to natural language expressions in *provide_type_analysis* module.

Attribute - an extended type in TIL. It is necessary to find a dependency *case*, where an individual is a dependent element and the indicator *'s* describes an independent element. Next, the dependency *nmod:poss* could be found - the dependent element is the individual and the independent element is the attribute that is owned by the individual. If there is no individual in the sentence, there is a need to find *nmod:poss* dependency directly (Figure 4, part c).

Property of the object - an extended type in TIL. To find out this type, there should be a dependency *nsubj*, where an individual is a dependent element and the truth value is an independent element. Sentence *„Bob is student."* has the property *student* which is assigned to the individual *Bob* (Figure 4, part d).

Property of the property - an extended type in TIL. To find out this type, there should be a dependency *amod*, where the dependent element is a property of property and the independent element represents property assigned to the object (Figure 4, part e).

Binary relation - an extended type in TIL. To find out this type, there should be a dependency *nsubj* or *nsubjpass*, where the dependent element is an individual and the independent element is a relation. Next a dependency *dobj* with another individual as the dependent element and relation as the independent element should be found (Figure 4, part f).

Value - an extended type in TIL. An expression has a type of value when it is a cardinal number *CD* or it can be expressed by the special nouns as *length, weight, temperature, height ...* (Figure 4, part g).

Property of the value - an extended type in TIL. It is related to all kinds of value. After identifying the value type in the sentence, the dependency *nsubj* should be found, where the dependent element is a value and the independent element is a property (Figure 4, part g, fourth sentence).

Each type specified in this chapter has been implemented in the *provide_type_analysis* module of the TIL semantic machine. If all the lexical objects of the natural language sentences have assigned the corresponding types, it is possible to proceed to the second phase of semantic analysis - the creation of the construction.

3.2. Automated synthesis of constructions

There is a dependency *ROOT* in the output from SCN (Figure 2). The benefit of this dependency is that it is in every analyzed sentence, so in our implementation, synthesis of construction always starts from it. After the type analysis is finished, a list of all found types of objects is made, which helps us to determine a dependency *ROOT* in the sentence.

In the sentence *„John is a singer."*, the *ROOT* of sentence is the object *singer* which is the property type. It is necessary to discover the related types - if the object *singer* is related to an individual then the object *singer* has to be intensionalized in the output construction.

$$\lambda w \lambda t[[[\hat{}0singer\ w]t]\ \hat{}John]$$

After all related types are identified, the object is built for every analyzed word. Each object is in specific relation to the other objects of the sentence. Sometimes more lexical units represent one type. In sentence *„Count of apartments in Bratislava is increasing."* objects *count*, *of*, *apartments* are originally incorrectly analyzed by SCN as independent words. In reality, they have together value type. In the final construction, our algorithm put them together as one object - *count_of_apartments*.

$$\lambda w \lambda t[[[\hat{}0increasing\ w]t]\ \hat{}0count\_of\_apartments\_in\_Bratislava]$$

If there are connectives in the sentence, it is analyzed in a specific way. For sentence *„John's sister is a fabulous cook and skilled sportswoman."* all related types are discovered and then the algorithm determines if *ROOT* contains conjunction by finding dependency *conj*.

$$\lambda w \lambda t\ [[[[\hat{}0cook\ w]t]\ [[\hat{}0fabulous\ w]t]]\ \lambda w \lambda t[\hat{}0John\ [[\hat{}0sister\ w]t]]] \wedge [[[[\hat{}0sportswoman\ w]t]\ [[\hat{}0skilled\ w]t]]\ \lambda w \lambda t\ [\hat{}0John\ [[\hat{}0sister\ w]t]]]]$$

The last type of the analyzed expressions are sentences containing negation. For sentence *„John was not raised in Bratislava."*, the following output is created:

$$\lambda w \lambda t[\neg[[\hat{}0raised\ w]t]\ \hat{}0John\ \hat{}0Bratislava]]$$

The constructions described in this chapter are examples of outputs produced by the TIL semantic machine. The implementation of our semantic machine generates outputs correctly what makes it a tool suitable for automated logical analysis of natural language.

4. CONCLUSIONS

In the paper, a useful prototype of the TIL semantic machine (only the second in the world) is introduced. We consider the implementation of the semantic machine to be successful, but we plan to work on it in the future too - we want to improve and expand it. From Figure 3 it is evident that the voluntary step of the TIL analysis - type control has not been implemented. It would also be helpful to achieve a two-level solution that can replace SCN by a proper syntactic parser. Another extension is the creation of a universal semantic machine that can generate TIL constructions from several world languages or in other projects realized on Department of Computers and Informatics [15], [16], [17], [18].

ACKNOWLEDGEMENT

This work was supported by the following projects:

- Slovak Research and Development Agency under the contract No. SK-AT-2017-0012: Semantics technologies for computer science education,
- Slovak Research and Development Agency, project number APVV-18-0214 and by KEGA Agency of the Ministry of Education, Science, Research and Sport of the Slovak Republic under Grant No. 003TUKE-4/2017: Implementation of Modern Methods and Education Forms in the Area of Security of Information and Communication Technologies towards Requirements of Labor Market.

REFERENCES

[1] TICHÝ, P.: The Foundations of Freges Logic. De Gruyter, Berlin and New York, 1988, ISBN 3-1101-1668-5 .

[2] FREGE, G.: ber Sinn und Bedeutung. Zeitschrift fr Philosophie und philosophische Kritik, Vol. 100, 25-50; translated as On Sense and Reference by M. Black. Geach and Black (eds. and trans.), 1980, 56-78.

[3] MATERNA, P.: Hovory o pojmu. Academia, Praha, pp. 158, 2016, ISBN 8-0200-2536-4 .

[4] MONTAGUE, R.: The Proper Treatment of Quantification in Ordinary English. Approaches to Natural Language, 1970, 115–153.

[5] BILANOVÁ, Z. – UCHNÁR, M.: Comparison of the approaches of Montague and Tich within a logical analysis of an English sentence. POSTER 2017. - Prague : Czech Technical University, 2017, 1–6.

[6] DUŽÍ, M.: Structural isomorphism of meaning and synonymy. Computacin y Sistemas. Centro de Investigacin en Computacin del Instituto Politcnico Nacional, 2014, Vol. 18, No. 3, 439–453.

[7] GAMUT, L. T. F.: Logic, Language, and Meaning, Vol. II: Intensional Logic and Logical Grammar. University of Chicago Press, Chicago, London, 1990, ISBN 0-2262-8088-2 .

[8] DUŽÍ, M. – MACEK J.: Analysis of time references in natural language by means of Transparent Intensional Logic. Organon F, Filozofick stav SAV, 2018, Vol. 25, No. 1, 21–40.

[9] SLEATOR, D.: Link Grammar, `https://www.link.cs.cmu.edu/link/`

[10] MANNING, Ch. D. – SURDEANU, M. – BAUER, J. – FINKEL, J. – BETHARD, S. J. – MCCLOSKY, D.: The Stanford CoreNLP Natural Language Processing Toolkit. Association for Computational Linguistics (ACL) System Demonstrations, 2014, 55–60. `http://www.aclweb.org/anthology/P/P14/P14-5010`

[11] BEDNÁR, B. – BILNANOVÁ, Z.: Towards Automated Translating Natural Language Sentences into Intensional Constructions. Informatics 2019, Danvers: IEEE, 2019, 26–31.

[12] DE MARNEFFE, M. C. – MANNING, Ch. D.: Stanford Typed Dependencies Manual, 2008. `https://www.bibsonomy.org/bibtex/2487e67b81281e38900ec97bd2915ec45/butonic`

[13] HORÁK, A.: The Normal Translation Algorithm in Transparent Intensional Logic for Czech. Faculty of Informatics, Masaryk University, Brno, 2001.

[14] MENŠÍK, M. – KERMASCHEK J. – CIENCIALA L.: Existential Generalization in TIL. 17th International Multidisciplinary Scientific GeoConference: SGEM 2017: conference proceedings: 29 June-5 July, 2017, Albena, Bulgaria. Vol. 17, No. 21, Sofia: STEF92 Technology Ltd., 2017, 311–318.

[15] ÁDÁM, N. – BALÁŽ A. – PIETRIKOVÁ E. – CHOVANCOVÁ, – E. FECIĽAK, P.: The impact of data representationson hardware based MLP network implementation. Acta Polytechnica Hungarica, Vol. 15, No. 2, 2018, 69–88.

[16] ŠIMOŇÁK, S. – UCHNÁR M. – FECIĽAK, P. – CHOVANCOVÁ, E. – CHOVANEC M.: Abstraction-enriched Formal Methods Integration. Acta Polytechnica Hungarica, Vol. 15, No. 7, 2018, 179–200.

[17] GAZDA, J. – ŠIMOŇÁK, S. – PIETRIKOVÁ E. [et al.]: Agent-based model of the spectrum auctions with sensing imperfections in dynamic spectrum access networks. Acta Polytechnica Hungarica, Vol. 15, No. 7, 2018, 179–200.

[18] VOKOROKOS, L. – CHOVANCOVÁ, E. – RADUŠOVSKÝ J. – CHOVANEC M.: A Multicore Architecture Focused on Accelerating Computer Vision Computations. Acta Polytechnica Hungarica, Vol. 10, No. 5, 2013, 29–43.

Received December 10, 2019, accepted December 12, 2019

BIOGRAPHIES

Branislav Bednár was born on 08. 04. 1997 in Prešov, Slovakia. In 2019 he graduated (Bc.) at the Department of Computers and Informatics at the Faculty of Electrical Engineering and Informatics of the Technical University of Košice. His main scientific interests are natural language processing and transparent intensional logic.

Zuzana Bilanová was born on 13. 07. 1992 in Košice, Slovakia. In 2015 she graduated (Ing.) at the Department of Computers and Informatics at the Faculty of Electrical Engineering and Informatics of the Technical University of Košice. Since 2015 she is PhD. student at the Department of Computers and Informatics at the Faculty of Electrical Engineering and Informatics of the Technical University of Košice. Her main scientific orientations are focused on creating new approaches in logical analysis of natural language, non-classical logical systems in computer science, and resource-oriented logic programming. Her secondary research areas are educational technologies for the effective implementation of engineering education concentrating on project and team based teaching.

Adrián Pekár received the Ph.D. degree in computer science from the Technical University of Košice, Slovakia, in 2014. Currently, he is an Assistant Professor with the Department of Networked Systems and Services, Budapest University of Technology and Economics, Hungary. Prior to this, he held research, teaching, and engineering positions in New Zealand and Slovakia. His research interests include network traffic classification and management, network traffic measurement data reduction and visualisation, cloud computing, and software-defined networking.