ACTA DIDACTICA NAPOCENSIA
Volume 5 Number 2, 2012

ICT COMPETENCES: ALGORITHMIC THINKING

László Zsakó, Péter Szlávi

Abstract: A lot has been said about what to teach in ICT in primary and secondary education. There are serious discussions even debates about it. Much less has been said about why ICT should be taught. [1] Competences are related actions and tasks done by people (somebody is competent in a certain field if they are able to solve common tasks related to that field). ICT curricula contain these competences, together with development tasks, activities, abilities and skills, always assuming a knowledge system you can rely on. One of the most important competences, which is gathering more and more ground in primary and secondary education of dynamically developing countries, is *algorithmic thinking*. This paper is aimed at dealing with its depths.

Key words: Informatics and education; Curriculum; Primary and secondary education; Algorithmic thinking; Problem solving

1. Introduction

The key competences set in the Hungarian National Curriculum are based on the Recommendation of the European Parliament and of the Council of 18 December 2006 on Key Competences for Lifelong Learning (2006/962/EC). [2]

The European Union has developed a key competence framework system consisting of 8 elements, which all individuals need for a successful life in today's modern economy and society, for acquiring and renewing knowledge, for evolving the paradigm of lifelong learning and the demand for being well-educated, and for personal self-realization (communication in your mother tongue, communication in a foreign language, mathematical and scientific, digital, learning, social competence, entrepreneurship and cultural expressiveness).

The Hungarian National Curriculum, rephrasing the above a bit, defines the following 9 key competences: [3]

1. Communication in mother tongue
2. Communication in a foreign language
3. Mathematical competence
4. Scientific competence
5. Digital competence
6. Efficient independent learning
7. Social and civil competence
8. Initiative and entrepreneur's competence
9. Aesthetic-art awareness and expressiveness

Relying on these key competences and subjects, competences can be defined for certain subjects and knowledge areas.

Possessing ICT competencies, you are able to apply basic ICT tools and methods for learning and problem solving in everyday life, at home and at work. Your knowledge can be applied practically in learning and operating new technologies and methods, in problem solving, in acquiring individual and community objectives as well as in making decisions that require understanding the possibilities of information society. [2]

Therefore, ICT competence includes main computer applications such as word-processing, data tables, databases, information storage and management, communication through the internet and electronic

Received 2 May 2012.

media in the fields of free time, information sharing, cooperative networking, learning and research. In addition, you must be aware of the problems concerning the credibility and reliability of accessible information and the ethic principles related to the interactive use of ICT tools.

The necessary skills cover finding, collecting and processing information, its critical application as well as distinguishing real and virtual relationships. It also involves the use of devices and tools that help producing, demonstrating and understanding complex information as well as access to Internet-based services, their use in research, applying ICT methods in critical thinking, creativity and innovation.

Many competences partly overlap and entwine: elements necessary for one area support the competences of another one.

2. The components of ICT competences

Since our starting point is the National Curriculum, ICT competences are also defined for the participants of primary and secondary education. Naturally, these competences could be expanded with respect to ICT specialists' training, but that could be the topic of another article.

2.1. Algorithmic thinking

In everyday life, both in learning and at work you continually make and execute algorithms, design series of activities and information flow processes, which make this world fully comprehensible only for those who are familiar with the basics of these activities. [4]

A Model Curriculum for K–12 Computer Science (Final Report of the ACM K–12 Task Force Curriculum Committee, October 2003): Computer science students learn logical reasoning, algorithmic thinking, design and structured problem solving all concepts and skills that are valuable well beyond the computer science classroom. [5]

National Curriculum Statement Republic of South Africa: The learner is able to design, implement, test and deliver efficient and effective solutions to problem situations. [6]

2.2. Data modelling

It is quite a common task for everybody to fill in forms. Moreover, you also make such things for others. While doing so, your aim is always to collect or pass information, which has to be supported by data sets and data structures. That is why it is so important to describe the objects of the world with data.

2.3. Modelling the real world

Real phenomena are often studied through models. In order to do so, you need to be familiar with the basic concepts and steps of modelling, and the methods of applying models. Beyond cognition, you are expected to consciously use the models for forecasting real phenomena, as well. It is a peculiarity of modelling that even the actuation of models is a complex creative process.

2.4. Problem solving

Creative human activity is often a kind of problem solving from the – as precise as possible – phrasing of the problem to the evaluation of the solution, which makes it necessary to analyse the problem in order to decide whether it is necessary to use an IT tool for the solution; what IT tool(s) can be used and how; and if such a tool does not exist, how to make one. [7]

According to George Polya the problem-solving process consists of the following phases [4,7]:

- Identifying the problem
- Understanding the problem
- Representing the problem
- Solving the problem
- Communicating the results

2.5. Communication skills

By now man-to-man and man-to-group communication have changed fundamentally: intelligent devices have stepped among the communicating partners; and these intelligent devices have either created new communication opportunities or simplified the old ones. This new kind of communication should be used in everyday life in a proactive way and in the meantime respecting the law: in learning, at work, in making contacts, relaxing, self-study, resting and development. [8]

2.6. Application skills

Problems arising in everyday life can often be solved more easily with IT tools than with traditional ones, which requires familiarity with basic general applications as well as with the related IT tools and methods.

2.7. Team work, collaboration, interoperability

ICT creates an opportunity to solve problems that are not solved by one person: starting with the ability to use others' results in problem solving through participating in project work to designing and realizing projects, which requires the skill of using ICT tools that support teamwork and understanding teamwork methodology.

2.8. Creative skills

ICT problem solving is often a creative process where you use ICT tools in the process of making something. Just like every act of creation, it starts with repeating elementary creations, continues with making new ones from these creations, and ends with making independent creations by request lists. The development of creativity goes from simple miming/pattern pursuing through varying patterns to real creativity.

2.9. Orientation and information skills

One of the important features of the information society is to guarantee the right of access to information. However, it is fairly difficult to find the necessary information in a huge set. So it is to arrange and place useful information in a way that it could be found and utilized effectively.

2.10. Systemic thinking

There are countless systems in your life, including the whole ICT system around you, the various network systems and communication systems. You have become used to thinking logically and understanding the ways of life through analysis, which often results in success but not always. When facing a system, you may have to apply another way of thinking. Simple classical logic might fail when it tackles a system (see chaos theory). A system can also operate by itself owing to the interaction of its parts. The output is often determined by the structure of the system and not by the efforts of people. If you can recognize the connections that control the events, you might become able to affect your life and work. To examine, handle, develop and create these connections requires another kind of knowledge and examination.

3. The specifications and levels of algorithmic thinking competence

In the beginning, algorithmization is not about computational implementation. Just think of a classic example, the age-old Euclidean algorithm for computing the greatest common divisor of two integers. In many cases the executor of an algorithm - the processor – is the human being himself/herself who has created and is interpreting the algorithm. (Moreover, when trying to solve a new problem, an experienced programmer also tries to walk in the computer’s shoes, thus testing the operability of the devised solution.) Besides carrying out algorithms day after day, people also make algorithms for themselves and for others.

It is only after this that the implementation of a precisely defined algorithm is entrusted to an automaton, to a computer. It basically requires a different way of thinking, because an intelligent algorithm implementer implements our algorithms reasonably, it controls the implementation and

sometimes corrects the mistakes made. An automaton, however, does not think but carries out (if it can) what is written in the program.

It can be stated that most people devise algorithms for others (and for themselves), while implementing algorithms is a common task for everybody. [4]

3.1. Recognising and Understanding Algorithms (a sequence of activities)

The most elementary level of algorithmic thinking is recognising algorithms and the problems that can be solved by using them.

What is an algorithm? It is a procedure that can be divided into steps; it is implementable (and there is an implementer belonging to it); it has a fixed order of executing operations; during the steps something happens to something; the steps are either elementary actions (that the implementer understands) or they are algorithms themselves (which implies further specification).

Understanding has two stages:

- You understand what you should do,
- You understand why you should do that and not something else.

The question "why" belongs to a higher level of thinking, to the analysis of algorithms, which means recognising, rejecting, varying and selecting alternatives.

Likewise, the ability to understand a known (i.e. it has already been explained) and the ability to understand an absolutely new algorithm that has never been seen before also represent different levels of knowledge. The latter is the third level of algorithmic thinking.

You perform many routine activities in your everyday life. Some of them like getting up and going to bed are programmed into you, whereas there are others that you create yourself by modifying similar ones in a changed – often "emergency" situations (e.g. going to school).

3.2. Implementing Algorithms (a sequence of actions)

If you received instructions from someone independent of you (especially in a dependency relation) step by step, you would have little scope for consideration but carry out the algorithms, whereas if it were you to form the instruction, it would be completely different.

The ability to implement algorithms is a level higher than the ability to understand them. It is not enough to simply understand a process but you have to monitor the interim stages during the implementation (keep in mind, register etc.) i.e. you should not pay attention to what is going on but to what you should do next depending on certain factors.

If you are in possession of this competence, you can perceive physical properties and their processes important for the implementation, and thus you can adjust the steps of actions if necessary.

3.3. Analysing Algorithms (a sequence of actions)

Analysing algorithms is partly about recognising the basic rules of constructing algorithms:

- Each basic step is to be executed (in the given order).
- You are to choose one of the basic steps and execute it.
- The basic steps are to be executed several times, repeatedly. [5]

Since the steps of algorithms themselves can also be algorithms, which can be labelled (actually, they should be labelled so that they could be quoted), the abstract notion of procedure can be formed.

On the other hand, the analysis of algorithms means that you understand what their parts are for, what your goal was when making them and how the whole problem solving process was broken down to smaller subtasks. … [9,10]

In addition to the above mentioned, the algorithm reading ability also belongs here. It means that you are able to understand a complex activity formed in an algorithmic language by someone else: you can

see the aims of the parts, their relations to other parts and you are also able to explain them. (Verbalization is not identical with understanding; in fact it is probably a new, higher level.)

3.4. Making Algorithms (a sequence of actions)

If you are able to understand and implement algorithms, you are not necessarily able to make new ones as that process requires some surplus. First you should consider

- what you know;
- what you would like to know;
- what will happen;
- what data you will have to work with;
- and how you can break down your tasks into smaller sub-tasks.

It partly means swapping reading activities for writing activities exchanging means, but there is much more to that. At lower competence levels you have guidelines, which help your thinking, but in this case you need to find out everything yourself. It cannot be done smoothly unless you acquire some systematic method for making algorithms (and their related data models). [10] You need to find a method to shortlist the infinite set of algorithms to a handy set with relatively few items. Obviously, you cannot do this without a large-scale restriction of tasks; the other pos-sible solution is to allow giving a small number of algorithm schemes instead of giving a small number of algorithms. The creative process in this case is selecting and combining the suitable algorithm schemes, and adapting them to concrete activities. The establishment of a scheme system requires analogical and abstract thinking. The distinction between essential and irrelevant features: abstraction; recognizing similarities: recognizing analogies.

3.5. Realising Algorithms (a sequence of actions)

It is now a real ICT task: the algorithm should be described with a tool in a way that the result could be executed by an automaton (e.g. computer). It also implies that you should learn the use of the tool (practically the programming language) you write the algorithm in.

The individual tools also have a specific way of thinking that you should acquire:

- How do you envisage the execution of the programs?
- How are the programs constructed?

On the other hand, you can rarely make flawless things at once. Therefore, you might need to see whether the algorithm created is correct or not, and to recognize and correct the mistakes if there are any. The latter would seemingly mean a simple analysis of the algorithm, but in ICT you have more possibilities. There exist such complex systems (programming environments) that you can use to make this activity more efficient than sheer thinking although, of course, it does not substitute thinking.

Encoding can be made much less tiresome if you formulate the rules that you use for writing a code in a given language from an algorithm which contains the substantive description of the solution. Such rules can be defined to any language, and their “transplantation” can be made mechanical to 90%. [11] The price is recognising and memorising the rules.

Testing is a “not liked” duty which requires much more thinking than you would expect. The purpose of testing is to give a schedule or scores to the discretion/controlling of the correction of the code. In order to achieve this goal, you should find a method on what data are needed to “move/stir” all the items of the code. It is obvious to try and rely on the analytical ability mentioned at Point 3, though it is not the only system. The apprehensive knowledge of the task (and not the solution!) could help you find relevant cases. After you detect an error, your analytical ability will also be needed for debugging.

It should be stated that encoding –in ICT education either – is not merely a tool for you to finally claim the correctness of the algorithm but often the culmination of problem solving (just think of simulation). [12]

3.6. Modifying and Changing Algorithms (a sequence of actions)

Understanding algorithms written by others is also a relatively easy task. Writing your own algorithm is not much more complicated, either when compared to the task of modifying and improving an algorithm produced by someone else. (Note: The literature often calls this an algorithm, as well.)

In the latter case it is not only the implementation that you have to figure out, but you need to understand the way the writer of the original algorithm was thinking and why he/she created the algorithm exactly that way etc.

You should also understand where you can enter the different world of thoughts of another person, what you can modify and what interventions will be effective etc. It may be much more difficult than writing an algorithm of your own starting from scratches.

You often receive inaccurately designed algorithms (or programs in the ICT world), which do not meet your expectations. Therefore, you must be able to change them so that they become useful and serve your purposes.

3.7. Designing Complex Algorithms (a sequence of actions)

The work you should invest does not grow linearly with the increase in the size of algorithms. Sooner or later you will reach a stage when the solution of the problem cannot be regarded as a single step. It is when abstraction and the systematic planning of algorithms could play a decisive role.

It is when you could consider setting sub-objectives and designing a series of activities for the particular sub-objectives. (It is quite common in everyday life when several people are working together. In programming, however, it is not always necessary though it is frequent during the development of large software systems.). Of course, it is a serious task to see that by meeting the sub-objectives and with a good synthesis, the task can be accomplished.

4. Algorithmic thinking and the Hungarian National Curriculum 2007 – Review

Table 1.

	1-4	5-6	7-8	9-12
Using IT tools	Understanding algorithms	Understanding algorithms	Understanding algorithms	Understanding algorithms
Application skills in informatics	—	Implementing algorithms	Analysing algorithms	Creating algorithms
Information technology	Implementing algorithms	Implementing algorithms???	Modifying algorithms	Complex algorithms
Info-communication	—	Implementing algorithms	Analysing algorithms	Creating algorithms
Media informatics	—	—	—	—
The information society	—	—	—	—
Library information	—	Understanding algorithms	Implementing algorithms	Implementing algorithms

The above table (see Table 1) shows the competence of *algorithmic thinking* for a given age group and a given topic at its highest level. Its appearance in IT is obvious (hence it is the field that is about algorithmization, datamodelling and problem solving), while the other fields need some explanation.

Using ICT tools (hardware and software) always requires the implementation of algorithms. For instance, if you want to install some software, you often have to face, understand then implement unknown algorithms.

For instance, this year's national ICT users' competition set a task where the participants received the raw data of a table in a simple text file. The text was not arranged in a table or divided by tabs, but spaces were used to make the printed table look good. If you wanted to convert this table into a real one, you either had to spend 30 to 40 minutes typing-copying or you could devise a clever algorithm which reduced the conversion to 3 minutes.

Logically, info-communication is also a kind of algorithm implementation, but there is much more to that here, as well. Creating communication networks and using them for delivering messages to an appropriate/selected group of addressees requires a serious knowledge of algorithm writing (and do not forget about data-modelling skills).

5. Other Ideas – Sándor Szántó

The levels of algorithmic thinking can be defined and related to other ways of thinking as follows (on the suggestions of Mariann Buda). [13 – The statements of the article are written in italics.]

5.1. Level 1

Application: When you only need to retrieve a particular (a given, named) procedure from your memory, which will help you overcome a given problem. (For example: if you are to find the common denominator, you will look for and select the named procedure, an algorithm.)

Implementation: deductive thinking (the application of an algorithm stored or given in a general format to a specific case)

We regard this level as two levels, actually. We firmly believe that there is a difference between understanding an algorithm and implementing it since implementation does not simply mean understanding. When you implement, you should carefully follow the states of the solution, you need to know which activities will take place in which state etc.

5.2. Level 2

Writing an algorithm i.e. the process when you are making an attempt to have steps recognized and start to formulate rules and generalisations basically requires inductive thinking.

Its main point: while solving a problem, the student observes a regularity (a rule), notices it several times and then encodes (records) it in some meaningful way.

We consider it to be two levels, as well: you have to tell the difference between analytical and creative skills. When you understand the elements of an algorithm, you know what role the elements play in it, but it does not necessarily mean that you are able to write such algorithms yourself, since the first one is a clear analytical activity, whereas the latter is a synthetizing one.

5.3. Level 3

It is a conscious effort for searching and selecting algorithms. When the student encounters a new but similar problem, he or she can recall, follow and use the procedure as a whole or in a slightly modified form.

Analogical thinking.

We believe that finding and selecting an algorithm for a task needs more elementary thinking than making a new algorithm. If you want to apply an algorithm you learnt before, you do not necessarily need to design a new one. It is true even if the algorithm is to be adapted to a particular task. Let us illustrate it with an everyday example: if you can make soft-boiled eggs, you are able to prepare hard-boiled eggs, as well, after modifying the cooking algorithm. However, it does not mean that you can create the recipe – i.e. the preparing algorithm – of any dish.

5.4. Level 4

Modifying and adapting an algorithm: the efficient and flexible transformation and joining of algorithms. Using the basic algorithm to develop a new procedure.

Creativity.

It corresponds to our Levels 6 and 7, and regarding the handling of complexity, we believe that it means a higher level of algorithmizing skill.

6. Other ideas – The Committee on Logic Education

Here are some of possibilities for what "algorithmic thinking" might be or involve:

i) application of an algorithm

ii) development of algorithms

iii) analysis of algorithms

iv) recognition of problems that have no algorithmic solution. [14]

It is an interesting, but at the same time a very good grouping. On the one hand, it does not deal with the implementation of algorithms or the checking their correctness. On the other hand, it clearly defines a competence on the fourth level that would or should rather belong to IT specialists' training.

7. Algorithmic thinking and program development

Several researchers study algorithmic thinking competence as a skill necessary for developing programs. Since it does not belong to the set aims of this article, we are only referring to this interesting food for thought with a short citation from a very interesting study:

The general concepts of algorithmic thinking, including functional decomposition, repetition (iteration and / or recursion), basic data organizations (record, array, list), generalization and parameterization, algorithm vs. program, top-down design, and refinement. Note also that some types of algorithmic thinking do not necessarily require the use or understanding of sophisticated mathematics.

Algorithmic thinking is key to understanding many aspects of information technology. Specifically, it is essential to comprehending how and why information technology systems work as they do. To troubleshoot or debug a problem in an information technology system, application, or operation, it is essential to have some expectation of what the proper behaviour should be, and how it might fail to be realized. Further, algorithmic thinking is key to applying information technology to other personally relevant situations. [15]

8. Tools Improving Algorithmic Thinking

There have been devised several tools that are independent of programming languages and can improve algorithmic thinking. One of the earliest such tools was Karel, the Robot. [16,17,18].

Karel is a "street urchin" that moves by stepping on corners of a grid of streets and avenues. At some corners there are walls Karel cannot climb over. On other corners there are pebbles (one on a corner) that Karel can pick arbitrarily and later can put them down. Being the implementer of the algorithm, Karel understands the following language:

Karel's actions:

- Start, Karel
- Turn right
- Turn left

What you can ask Karel:

- Face north
- Face south
- Face east

- Step
- Pick the pebble
- Throw a pebble
- Stop
- Face west
- Stand in front of a wall
- Check pebble/Is there a pebble here?

Examples:

```
Turn northward:
 WHILE NOT facing north
 ITERATE
  Turn right
 END ITERATE.
```

```
Go to the wall:
 IF NOT check wall
 THEN Step; Go to the wall
 END IF.
```

```
Go forward 10:
 ITERATE 10 times
  Turn right
 END ITERATE.
```

```
Swap:
 ITERATE 10 times
  IF check pebble
  THEN Pick up the pebble
  ELSE Throw a pebble
  END IF
 END ITERATE.
```

The essence of Karel, the Robot:

- it supports top-down approach in design; the procedure notion is primary;
- hiding the notion of data (substituted by state query operations), algorithms without variables;
- performable algorithms.

Recently, Karel's "three-dimensional sister", Alice has been defined:

Alice is primarily a scripting and prototyping environment that allows the user to build virtual worlds and write simple programs to animate objects (e.g., animals and vehicles) in those worlds. Objects in Alice can move, spin, change color, make sounds, react to the mouse and keyboard, and more. By writing simple scripts, Alice users can control object appearance and behavior. During script execution, objects may respond to user input via mouse and keyboard. Each action is animated smoothly over a specified duration. In the rest of this paper, we will describe the Alice programming environment, and then propose how Alice may also be used to assist in supporting the development of algorithmic thinking for novice programmers. [19]

9. Conclusion

After reviewing the available literature on the subject, and considering our own experience, we can clearly state that everybody considers the skill of algorithmic thinking to be of great importance. Several researchers relate it to the mathematical competence, as well. (*We considered algorithmic thinking as one kind of mathematical thinking, and contrasted it with intuitive pattern recognition and analogical thinking. We think of algorithms as objects – namely as a list or sequence of steps to be performed, possibly given by a flow chart or diagram*). [14,15]

Relevant authors also agree that this competence is the skill of understanding, implementing, analyzing and designing algorithms. Expert's opinion differ partly on how these skills should be built on each other; on the other hand, very few colleagues study algorithmic thinking skills, which is necessary for all, independently of the programming activity. That is why we primarily laid the emphasis on it in this article.

References

[1] Horváth Gy.; Szlávi P. & Zsakó L. (2010). Informatics (ICT) competencies, *8th International Conference on Applied Informatics, Eger, Hungary*

[2] Recommendation of the European Parliament and of the Council of 18 December 2006 on Key Competences for Lifelong Learning (2006/962/EC)

[3] Nemzeti Alaptanterv (National Curriculum Statement Republic of Hungary), http://www.nefmi.gov.hu/kozoktatas/tantervek/nemzeti-alaptanterv-nat [2012.04.30.]

[4] Amorim, C. (2005). Beyond Algorithmic Thinking: An Old New Challenge for Science Education, *Eighth International History, Philosophy, Sociology & Science Teaching Conference, July 15 to July 18, 2005, University of Leeds, England*

[5] A Model Curriculum for K–12 Computer Science. *Final Report of the ACM K–12 Task Force Curriculum Committee, October 2003*, http://csta.acm.org/Curriculum/sub/CurrFiles/ K-12ModelCurr2ndEd.pdf [2012.04.30.]

[6] National Curriculum Statement Republic of South Africa, http://www.education.gov.za/Curriculum/SUBSTATEMENTS/Information%20Technology.pdf [2012.04.24.]

[7] Vass V. (2009). *A kompetencia fogalmának értelmezése,* Oktatáskutató és Fejlesztő Intézet

[8] Benczúr A. (2003). A communication fejlődése és az információs forradalom. (Evolution of Communication and the IT Revolution), *Természet világa (The World of Nature, special edition)*, pp 74-79

[9] Hvorecky, J. & Kelemen, J. (1983). *Algoritmizácia, elementárny úvod*, ALFA, Bratislava

[10] Hromkovic, J. (2009). *Algorithmic Adventures – From Knowledge to Magic,* Springer

[11] Szlávi P. & Zsakó L. (1986). *Módszeres programozás*, Műszaki Könyvkiadó

[12] Horváth L.; Szlávi P. & Zsakó L. (2005). *Modellezés és szimuláció*, ELTE IK

[13] Szántó S. (2002). Az algoritmikus gondolkodás fejlesztése, *Új Pedagógiai Szemle* May 2002, http://www.oki.hu/oldal.php?tipus=cikk&kod=2002-05-mu-Szanto-Algoritmikus [2012.04.30.]

[14] Algorithmic Thinking, *The Committee on Logic Education of the Association of Symbolic Logic*, http://www.ucalgary.ca/aslcle/nctm/Q2.html [2012.04.30.]

[15] Fluent With Information Technology by the National Research Council, *National Academy Press*, June 1999, http://www.nap.edu/html/beingfluent/ [2012.04.30.]

[16] Koster, C.H.A. (1984). *Systematisch leren programmeren*, Educaboek

[17] Hanák P. (1988). *Programozás ELAN-nal*, Műszaki Könyvkiadó

[18] Pattis, R. (1981). *Karel the Robot*, John Wiley & Sons

[19] Cooper, S.; Dann, W. & Pausch, R. (2000). Developing Algorithmic Thinking with Alice, *Information Systems Educators Conference 2000, Philadelphia, November 2000* http://www.stanford.edu/~coopers/alice/isecon00.PDF [2012.04.30.]

Authors

László Zsakó, Faculty of Informatics, Eötvös Loránd University, Budapest, Hungary, e-mail: zsako@ludens.elte.hu

Péter Szlávi, Faculty of Informatics, Eötvös Loránd University,, Budapest, Hungary, e-mail: szlavi@ludens.elte.hu