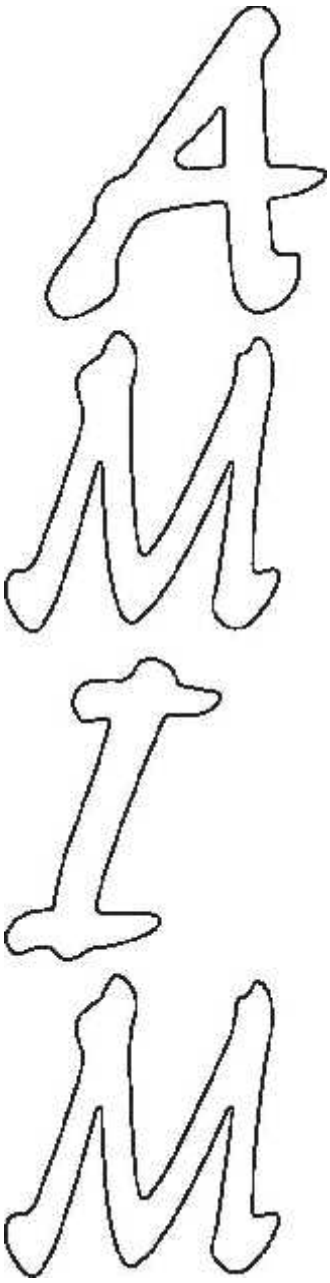
AMIM, *ტომი 25*, No2, 2020 ISSN 1512-0074



გამოყენებითი მათემატიკა

ინფორმატიკა და მექანიკა

თბილისის უნივერსიტეტის გამომცემლობა
თბილისი

ივ. ჯავახიშვილის სახელობის თბილისის
სახელმწიფო უნივერსიტეტი
ი. ვეკუას სახელობის გამოყენებითი
მათემატიკის ინსტიტუტი

გამოყენებითი მათემატიკა, ინფორმატიკა და მექანიკა

In this issue of the journal, dedicated to the memory of the founder of the school of classical mathematical logic in Georgia, the honor scientist of Georgia Shalva Pkhakadze, is published the reports of the First Tbilisi International Summer School "Logic, Language, Artificial Intelligence", which was funded by the project №MG-ISE-19-1315 of Shota Rustaveli Science Foundation of Georgia.

ჟურნალის წინამდებარე გამოცემაში, რომელიც ეძღვნება საქართველოში კლასიკური მათემატიკური ლოგიკის სკოლის ფუძემდებლის, მეცნიერებათა დამსახურებული მოღვაწის, შალვა ფხაკაძის ხსოვნას, ქვეყნდება შოთა რუსთაველის საქართველოს ეროვნული სამეცნიერო ფონდის №MG-ISE-19-1315 პროექტით დაფინანსებული თბილისის პირველი საერთაშორისო საზაფხულო სკოლის „ლოგიკა, ენა, ხელოვნური ინტელექტი“ მოხსენებები.

თბილისის პირველი საერთაშორისო საზაფხულო სკოლა
„ლოგიკა, ენა, ხელოვნური ინტელექტი“
09 – 15 სექტემბერი, 2019
FIRST TBILISI INTERNATIONAL SUMMER SCHOOL IN
„LOGIC, LANGUAGE, ARTIFICIAL INTELLIGENCE“
Dedicated to the memory of the founder of the school of classical mathematical logic in Georgia, the honor scientist of Georgia, Shalva Pkhakadze
09 – 15 September, 2019
/ ORGANIZERS
/ LECTURERS
/ SPONSOR
/ SUPPORTERS

AT THE ORIGINS OF THE FOUNDATION OF I.VEKUA INSTITUTE OF APPLIED MATHEMATICS PROF. SHALVA PKHAKADZE - 100

George Jaiani, Khimuri Rukhaia

I. Vekua Institute of Applied Mathematics
I. Javakhishvili Tbilisi State University
University str. 2, 0186 Tbilisi, Georgia.
george.jaiani@gmail.com

The Institute of Applied Mathematics of I.Javakhishvili Tbilisi State University was founded in 1968. Last autumn the jubilee to mark the 50^{th} anniversary of the Institute was widely celebrated at the University. At first there were 10 departments at the Institute. Among them was the department of Mathematical Logic and Algorithms. It was headed by the doctor of phys.-math. sciences, Prof. Sh. Pkhakadze.

To the centenary of his birthday was dedicated the one day scientific session at the Institute as well as the section of applied logic and programming at the XXXIII International enlarged session. On September 09-15, 2019, at the Technical University the Tbilisi International summer school "logic, language, artificial intelligence", dedicated to the same event, was held.

Shalva S. Pkhakadze was born on April 7, 1919, in the village of Upper Sakara (Zestafoni district). He successfully finished the Upper Sakara primary school (1930) and later Zestafoni secondary school (1933) and Zestafoni pedagogical school (1936) and finally the physics and mathematics faculty of Tbilisi State University (1941).

He defended his graduate thesis "On the geometrical theory of differential equations" under the supervision of Acad. Giorgi Chogoshvili. In 1942-1952 Shalva Pkhakadze worked as a teacher in different schools of the Zestafoni district. At the same time, in 1944-49 he was a methodist of the educational department of the Zestafoni district and was head of the mathematical section of the above department.

In 1949 Shalva Pkhakadze began to study the theory of integrals and the set theory under the supervision of the Corresponding Member of the Academy Vladimer Chelidze and a year later he started his studies in the measure theory.

In 1952 he read his lecture at the seminar of the functions theory department of A. Razmadze mathematical institute, thus announcing the main

results of his investigations. At the same time the new problems were set forth, concerning the theory of Lebesgue type measures. Taking V. Chelidze's advice, Sh. Pkhadze decided to present part of his works, namely, the one, in which he deals with the question of validity of The Fubini Theorem, concerning the change of integration order in double integrals as a dissertation thesis. After delivering the above paper, he was invited to work at the A. Razmadze mathematical institute as a junior research worker (1952) where he finished his candidate's thesis and defended it successfully in 1953. The same year he made several reports at the seminar of Academician P.S. Novikov in Moscow (V.M. Steklov Mathematical Institute) where he formulated the problems, posed by him and presented his own results. The positive evaluation of his works on the side of P.S. Novikov and the members of the seminar was a powerful incentive for him to go on with his fruitful scientific studies. Since 1953 Shalva Pkhakadze has been the senior research worker of the A. Razmadze mathematical Institute.

In 1959 he successfully defended his doctoral thesis "On the theory of Lebesgue type measures" and in January, 1965 was granted the title of professor. During this period Shalva Pkhakadze got important results in the general set theory and namely, in the theory of measure. Since as a result of development of mathematics and theory of automata the need for specialists, working in the field of mathematical logic increased in the 60ies of the last century, Shalva Pkhakadze did his best to found and promote the theory of mathematical logic in Georgia.

Since 1966 he had been reading general and special courses of lectures in mathematical logic at the physics and mathematics department of the University. At the Institute of Applied Mathematics of Tbilisi State University Sh. Pkhakadze, with the help of Acad. Ilia Vekua, founded (for the first time in Georgia) the department of mathematical logic and theory of algorithms which was staffed mainly by his own students. At the same time Shalva Pkhadaze began developing a new original direction in mathematical logic where his pupils – O. Chankvetadze, Kh. Rukhaia, V. Pkhakadze, Z. Khasidashvili were involved. Besides, in the department of mathematical logic and theory of algorithms investigations were carried out in the proof theory (M. Rogava, O. Tskhadadze), in the theory of algorithms (R. Omanadze, M. Tetruashvili), theory of automata and automated proof (N. Kalandarishvili, K. Pkhakadze), theory of measure and sets (A. Kharazishvili, A. Kipiani).

The basic results, obtained in this direction, are given in Sh. Pkhakadze's monograph "some problems of the notation theory". This work is of great theoretical and practical significance and in fact, creates the theory of contracting symbols which had existed only as a part of the definition theory before. In the monograph the rational system is found which introduces

contracting symbols, terminology is worked out and basic theorems on the main properties of contracting symbols and contracted forms are proved. On the basis of the actual problems, set by him, his pupils are involved in developing the theory of contacting symbols even today (Kh. Rukhaia, K. Pkhakadze, L. Tibua).

The results of Shalva Pkhadze and his pupils arose great interest not only among theoreticians but among specialists, working in applied fields of mathematics. For example, in 1979 the agreement was signed between the I. Vekua Institute of Applied Mathematics and the Department of digital automata of the Institute of Cybernetics at the Ukrainian Academy of Science. It provided cooperation between the members of the mathematical logic and algorithm theory of the Institute of Applied Mathematics and department of digital automata at the Ukrainian Institute of Cybernetics. The results, given in the above monograph and those, obtained by his pupils at the same period of time, were mentioned at the All-Union conference "Artificial intelligence and automatization of mathematical investigations" (Kiev) and in 1979 at the conference, dedicated to Gottlob Frege (Jena). The interest after these reports was great and as a result close contacts were established between the local group of logicians and group of mathematicians from Irkutsk, headed by Acad. V. Matrosov. The significance of a certain direction in logic is testified by means of Sh. Pkhakadze's monograph in the doctoral thesis of F. Van Ramsdok (Amsterdam, 1966) where Pkhakadze's theory of contracting symbols together with the work of the famous mathematician P. Axel is evaluated as the primary source in term rewriting for the case of connected variables.

Shalva Pkhakadze's critical report about Church's thesis arose special interest at the VI All-union conference of logicians (1983). The results, obtained along these lines, are mainly contained in his scientific work "An intuitively computable, everywhere defined function and Church's thesis" (Tbilisi University Press, 1984). In it he considers a very principal and important issue of the validity of Church's famous thesis.

Besides intensive scientific activities he led an active pedagogical life. For more than 10 years he worked at the Tbilisi Polytechnical Institute as Professor at the Chair of higher mathematics (1962-1972). Since 1962 he had been teaching at the Tbilisi State University. His activities as a scientist and a teacher were especially effective during his work at the Institute of Applied Mathematics of TSU, since the main goal of the initiator of founding this institute, I. Vekua was to create the base for the joint research work of scientists, teachers, postgraduates, candidates and students. At the Institute Sh. Pkhakadze was a supervisor of post graduate students and candidates. Among his students are 2 doctors of science (Acad. A. Kharazishvili and R. Omanadze) and 6 candidates of science

(M. Tetruashvili, Kh. Rukhaia, G. Kobzev, Z. Khasidashvili (working now in Israel), O. Chankvetadze and T. Kutsia (now in Austria). The students of mechanics and mathematics faculty of the University at all three stages are using his manuals and textbooks even today.

Sh. Pkhakadze took an active part in the social life of the country. He had been member of different scientific councils, specialized councils among them, granting doctor's and candidate's degrees. He was a member of the problematic council of mathematics and mechanics in Georgia, namely, chairman of the section of mathematical logic at this council and member of the methodology section. He was also a member of the education overseers council at V.M. Komarov mathematical school.

For his long and fruitful scientific and pedagogical activities Shalva Pkhakadze was granted the title of the honoured scientist and in 1979 at his 60th birthday jubilee – I. Javakhishvili medal.

Shalva Pkhakadze passed away on August 8, 1994 (Buried at the Saburtalo pantheon), but his ideas and activities which he started at the I. Vekua Institute of Applied Mathematics, are still going on successfully for the benefit of our country and on the centenary of his birth we are paying tribute to his memory.

Sitting (from left to right): Boris Khvdelidze, Elene Obolashvili, Ilia Vekua, Revaz Kordzadze, Tedore Tskhadaia, Gvanji Mania. Standing (from left to right): Guram Kharatishvili, Vakhtang Zhgenti, Amiran Getia, Alexander Khvolesi, Shalva Pkhakadze, Tengiz Gegelia. - Institute of Applied Mathematics, 1968.

SHALVA PKHAKADZE – A BRIEF OVERVIEW OF HIS LIFE

Khimuri Rukhaia[1], Konstantine Pkhakadze[2]

[1] I. Vekua Institute of Applied Mathematics
I. Javakhishvili Tbilisi State University
University str. 2, 0186 Tbilisi, Georgia.
[2] Center for Georgian Language Technology
Georgian Technical University
Kostava str. 77, 0160 Tbilisi, Georgia.
gllc.ge@gmail.com

Abstract

The paper is dedicated to the 100^{th} anniversary of the founder of the school of classical mathematical logic in Georgia, honored scientist of Georgia, Shalva Pkhakadze and it is mainly based on Khimuri Rukhaia's, Otar Chankvetadze's and Konstantine Pkhakadze's earlier publications, which are titled as "Shalva Pkhakadze" (1999, Tbilisi State University, Vekua Institute of Applied Mathematics, 1-50) and "A short overview of Shalva Pkhakadze's Scientific and Pedagogical Activities" (2014, Book of Abstracts of V annual international conference of the Georgian Mathematical Union, 31-38). Thus, the paper is a short overview of Shalva Pkhakadze's personal and scientific life, in the final part of which we present his general semantic program, in other words, a natural semantic program, created with the aim of the foundation of mathematics. Also, in the paper, we give a new theoretical view on the origin of different natural languages. It must be underlined here that this new view together with natural specifics of the Georgian language is mainly based on Shalva Pkhakadze's notation theory and natural semantic program.

Brief Biographical Notes – Shalva Pkhakadze, Founder of the School of Classical Mathematical Logics in Georgia, Honored Scientist of Georgia

1. Shalva Pkhakadze was born on April 7, 1919, in the Zeda Sakara village of the Zestafoni region of the Georgian Democratic Republic.

2. In 1941 he graduated with honors from the Faculty of Physics and Mathematics of Tbilisi State University (TSU).
3. From 1942 to 1952, he had been teaching at various schools in the Zestafoni district.
4. In 1952 he was invited to Razmadze Institute of Mathematics, where, in 1953, he defended his candidate dissertation.
5. In 1956, he completed his doctoral dissertation "The Theory of Lebesgue Measure", which was successfully defended in 1959;
6. In 1961 he was awarded with degree of the Doctor of Sciences, in 1965 – with title of the Professor, in 1967 – with title of the Honored Scientist of Georgia.
7. In 1969, at the Vekua Institute of Applied Mathematics, he founded the Department of the Mathematical Logic and Theory of Algorithms with the aim of formation of the school of classical mathematical logic in Georgia.
8. In 1970, with the above already mentioned aims, at the Faculty of Mechanics and Mathematics of the Javakhishvili Tbilisi State University, he founded a faculty seminar and specialized courses in mathematical logic and set theory.
9. In 1977, he published a fundamental monograph "Some Problems of the Notation Theory", which laid the foundation for a completely new mathematical theory today is named as Shalva Pkhakadze's notation theory.
10. In 1978, he was awarded with Ivane Javakhishvili Order of Honor, in 1979 – with Gottlob Frege Order of Honor.
11. He died on August 8, 1994. He is buried in Saburtalo Pantheon of Public Figures;
12. On February 23, 1995, according to decision №23 of the local government board of Zestafoni district, №3 Secondary School of Zestafoni, where he studied in his childhood, in his honor was named after Shalva Pkhakadze, a honored scientist of Georgia.
13. His university textbook "Mathematical Logic – Foundations" in three parts was published after his death (in 1996 (part 1), in 1999 (part 2), in 2009 (part 3)).
14. In 2019, with financial and institutional support of Shota Rustaveli National Science Foundation of Georgia, Georgian Technical University and Georgian National Academy of Sciences, the First Tbilisi

International Summer School in "Logic, Language, Artificial Intelligence" was held, which was dedicated to the memory of the founder of the Georgian school of classical mathematical logic, honored scientist of Georgia, Shalva Pkhakadze.

1 Family

This paper, as well as this edition of the present journal is dedicated to the 100^{th} anniversary of the founder of the school of classical mathematical logic in Georgia, honored scientist of Georgia, Shalva Pkhakadze.

Shalva Pkhakadze was born in the village of Zeda Sakara of Zestafoni region in the family of Samson Pkhakadze and Nino Putkaradze. Besides him four more children grew up in the Pkhakadze family – his sister Tamara and his brothers – Mikheil, Vasil, and Petre.

On the photo from left to right are Shalva Pkhakadze's: sister Tamara Pkhakadze, mother Nino Putkaradze, Shalva Pkhakadze himself, wife Mary Tskhadadze, father Samson Pkhakadze.

All the five, due to their professional and social activities, became respected and highly rated public figures.

Tamar Pkhakadze (1914 - 1983) was an honored geologist of Georgia. She discovered and studied a medical spring in the Terjola region of Georgia, which in her honor was named as Tamara's spring.

Mikheil Pkhakadze (1917 - 2007), an honored physician of Georgia, a retired colonel, a medical officer, who participated in the second world war, was granted by several orders, had devoted all of his life to the care of public health.

Vasil Pkhakadze (1924 - 1994), an honored physician of Georgia and Abkhazia, author of a number of scientific publications, had been work-

ing for years in Abkhazia, where due to his particularly responsive nature and high professionalism he acquired many friends, who are respecting his memory even today.

Petre Pkhakadze (1928 - 1984), an honored meliorator of Georgia, was a scientific secretary of the melioration institute of the agricultural academy of Georgia. He was author of a number of scientific works. His doctoral thesis was dedicated to important problems of the drainage and utilization of Kolkheti lowland.

On the photo from left to right are brothers of Shalva Pkhakadze: Vasil, Petre, and Mikheil.

One could say much more about each of them since their merits are not limited to the above dry facts, but here the merits of their parents – Samson Pkhakadze (1888 - 1967) and Nino Putkaradze (1901 - 1987) must be stressed especially – in their family the cult of faith and honesty was a leading category – this was a main cause of that they raised their children with faithfulness, kindness and love to the humans and homeland.

2 Personal Life

Shalva Pkhakadze was born on April 7 – on the day of Annunciation – in 1919, in Zeda Sakara, a village of the Zestafoni region of Georgia. He graduated with honors from the primary school of Zeda Sakara in 1930, from the Zestafoni secondary school1 in 1933,[1] and from the Zestafoni high-

[1]In 1995, in honor of him, to this school was given his name

school in 1936.

In 1941 Shalva Pkhakadze graduated with honors from the department of physic and mathematics of the Tbilisi State University and later, in 1942-1952, he worked as a teacher of mathematics at various secondary schools of Zestafoni region. Simultaneously, in the same years, he was a Methodist of Zestafoni region department of education and the leader of mathematical section of schoolteachers of the region.

In 1949, under the guidance of Vladimer Chelidze, Shalva Pkhakadze began intensive investigations in the set theory and theory of integrals, and a year later he began his independent research in the measure theory. On January 1952 he delivered a report at the seminar of the department of function theory of Andria Razmadze Mathematical Institute of the Georgian Academy of Sciences. In the report he presented the obtained results, described the main directions of his researches and set up a series of interesting scientific problems in the theory of the Lebesgue measure. At the same session of the seminar he accepted the advice of Vladimer Chelidze to arrange as candidate dissertation a part of his results concerning to the validity of the Fubini theorem on the change of the integration order in iterated integrals of characteristic functions of plane sets.

As a result of above-mentioned report, the same year, he was invited to work at the Andria Razmadze mathematical institute as a Junior Researcher. Filling this position during a year he passed candidate exams and wrote final version of his candidate dissertation, which he defended on June 30, 1953. The same 1953 year, in October, Shalva Pkhakadze delivered several reports at Steklov Mathematical Institute at the seminar of Academician Pyotr Sergeyevich Novikov. In those reports he described the main direction of his research in detail, formulated main problems of this direction and presented obtained results. The positive opinion of the participants of the seminar and of the Academician Pyotr Novikov strengthened his belief in his forces. Thus, after this, he continued the fruitful investigations in the already chosen by him direction with increased motivation.

In December 1955 he filled the position of Senior Researcher and in 1957 he was given the official status of Senior Researcher in the speciality "Theory of function of a real variable". In 1956 Shalva Pkhakadze prepared the final version of his doctoral dissertation "Toward the Theory of Lebesgue Measure", which he has successfully defended in 1959. In January 1965 he was granted Academic Status of Professor.

Since at that stage of development of mathematics and automata theory the need of specialists in mathematical logic had strongly increased in general, Shalva Pkhakadze considered as his duty to devote all his forces to founding and developing mathematical logic in Georgia. Thus, in 1967 he began to study mathematical logic and from this year to the end of his

life he delivered lectures and special courses in mathematical logic at the Tbilisi State University. Parallel to this, due to an active support of the management of the institute of Applied Mathematics of the Tbilisi State University, especially due to support of the director and founder of this institute Academician Ilia Vekua in 1969 Shalva Pkhakadze founded a department of Mathematical Logic and Theory of Algorithms, which was first research unit of such type in Georgia.[2] – It must be underlined here that Shalva Pkhakadze, as a founder of the Georgian School of the Classical Mathematical Logic, overcomes all difficulties which were connected with the change of the area of investigations and, even more, it was able for him to create a new direction in the Classical Mathematical Logic.

For years Professor Shalva Pkhakadze successfully combined intensive scientific research with pedagogical work at the University. He was a member of various scientific councils, among them of special councils for considering candidate and doctoral dissertations. Also, he was a member Georgian Scientific Council for Problems of Mechanics and Mathematics, a member of its Methodic Section and the Head of the Section of Mathematical Logic. At the same time, as the member of the board of the Tbilisi Physical-Mathematical school of talented pupils, he was systematically helping the teachers of this school to overcome difficulties connected with the new program in mathematics.

For long and fruitful scientific, pedagogical and social activities Shalva Pkhakadze was awarded with the title of Honored Scientist of Georgia in 1967. In 1978 he was awarded with Ivane Javakhishvili's Honorary Medal and, also, in 1979 – with Gottlob Frege's Honorary Medal.

He passed away on August 8, 1994. He is buried in the Saburtalo Pantheon of Public Figures. After his death, in 1996-1999, at the Ivane Javakhishvili Tbilisi State University, by his authorship and under the editorship of Konstantine Pkhakadze and Vakhtang Pkhakadze, the textbook for students "Mathematical Logic - Foundations" was published in three volumes. Also, On February 23, 1999, by decision №23 of the Local Government Council of the Zestafoni district, to the Zestafoni secondary school №3, where he studied in his childhood, was given the name of the Honored

[2]The first members of this newly established department were well known Georgian scientists Memed Rogava, Otar Chankvetadze, Otar Tskhadadze, Khimur Rukhaia, Rezo Tsakadze, Mikheil Tetruashvili, Nana Kalandarishvili, Alexsandre Kharazishvili, Roland Omanadze, Archil Kipiani, and Genadi Kobzev. At the same time, Most of them were members of the university seminar in mathematical logic led by Shalva Pkhakadze. Besides, among his followers – among his pupils and pupils of his pupils are also doctors of sciences Vakhtang Pkhakadze, Zurab Khasidashvili, Giorgi Pkhakadze, Temur Kutsia, Manana Pkhakadze, Lali Tibua, Besik Dundua, Mikheil Rukhaia, Konstantine Pkhakadze, Aleksandre Maskharashvili, Lasha Abzianidze, Merab Chikvinidze, Giorgi Chichua, and Shalva Malidze.

Scientist of Georgia Shalva Pkhakadze.

3 Scientific Life

3.1 Measure Theory

The first series of works at the first stage of Shalva Pkhakadze's scientific activities is dedicated to the theory of integrals [1, 2]. The results of these works are mainly included in the paper "On Iterate Integrals" [2], which is a short exposition of his candidate dissertation. Here the problem on the validity of the Fubini theorem on the change of the integration order is studied for iterated integrals of the characteristic functions of sets whose intersections with lines parallel to coordinate axes are Borel-measurable sets. Using a rather complicated arguments, the author obtains the following result: if the above-mentioned sets belong to the zero Borel class, then for the corresponding characteristic functions the Fubini theorem is valid. This result is final in some sense, namely, by means of an appropriate example it is shown that in the case where the intersection belongs to the first Borel class, the Fubini theorem, in general, is not valid.

Another series of the works of this stage is dedicated to the general set theory and measure theory [3–18]. A part of these works is included in his doctoral dissertation "The Theory of Lebesgue Measure" [9], which has become a classical monograph in the theory of invariant measure. It involves practically all results obtained earlier by the author in this direction. This monograph is a fundamental research in the theory of invariant measures. Therein a most productive notion – the notion of an absolutely null set is introduced. This notion is a basis of all the dissertation as well as of some research works carried on later by his followers.

It must be noted, that the introduction of this productive notion – the notion of an absolutely null set had required a very deep investigation. Really, the idea of introducing the notion of such sets for the Euclidean space R^n which may be neglected in the sense of measure, has occurred to several authors, for instance, there are a notion of a set of unconditional zero measure and others. However, they did not turn out to be productive and did not play any significant role in the development of the measure theory.

Unlike other authors, in defining absolutely null sets, Shalva Pkhakadze imposed restrictions not only on the set A under consideration, but also on any set which "naturally" has to be absolutely null along with A. Namely, he investigated four versions of the notion of absolutely null set and showed the unproductivity of the first three of them.

Thus, According to the first version, the restrictions "is of zero measure"

and "is not of positive measure" are imposed only on the set $A \subset R$ under consideration, according to the second version – on any subset of A, according to the third, respectively, fourth versions, on any finite, respectively, countable configuration of A (A set A^* is said to be a finite, respectively countable, configuration of A, if it can be represented as a union of a finite, respectively, countable, number of sets congruent to a subset of A) and it is proved that in the case of the first three versions the union of two absolutely null sets can coincide with the whole space, when in the case of the fourth version (which Shalva Pkhakadze adopted as a final definition), absolutely null sets have a series of very important and productive properties.

In particular, the class of such null sets is closed with respect to the finite union and is not closed with respect to a countable union. Both facts play an important role while finding various extensions of Lebesgue type measures thus making it possible to solve Sierpinski's problems on extendibility of solvable classes and extendibility of Lebesgue type measures. The last problem is solved by Shalva Pkhakadze on the ground of a most probable hypothesis, namely on the ground of the hypothesis that there is no unattainable cardinal number less or equal to the continuum. Moreover, Shalva Pkhakadze discovered a series of important properties of his absolutely null sets formulated as a necessary and a sufficient condition and proved that each from these conditions can be taken as a basis for defining the notion of an absolutely null set. Two of those properties can be formulated as follows:

(1) For a set $A \subset R^n$ to be absolutely null, it is necessary and sufficient that any countable configuration of A be vanishing (a set $X \subset R$ is said to be vanishing if there is a set of zero Lebesgue measure which can be represented as $\Pi_{i=1}^{n} X_i$, where each X_i, is congruent to X);

(2) For a set $A \subset R^n$ to be absolutely null, it is necessary and sufficient that for any Lebesgue type measure μ there exists its extension μ_1, with $\mu_1(A) = 0$.

On the basis of the property (1), Shalva Pkhakadze constructed a non measurable absolutely null set in R^n for $n = 1, 2$. In this connection, he poses the problem of constructing a non measurable absolutely null set in R^n for $n = 3, 4, \ldots$. This problem was solved by A. Kharazishvili on the basis of the same property (1).

In the fundamental monograph "The Theory of Lebesgue Measure", Shalva Pkhakadze poses also other important problems. His followers were working successfully on these problems. For instance, M. Tetruashvili generalized Shalva Pkhakadze's results for topological groups, and

A. Kharazishvili along with the above-mentioned problem also solved three other ones (see the problems II, III, and IV in [9]).

The main item of the fundamental monograph under consideration is the elaboration of powerful methods for finding the Lebesgue measure extensions with various properties. However, the results in the general set theory obtained using the methods elaborated in this fundamental work are also of major interest. Namely, in [9] results of Sierpinski on the existence of a set of almost invariant zero measure and of a set of the first category are essentially generalized, some important examples are constructed and general covering theorems are proved. Besides this in [9], the notion of almost completely asymmetric set is introduced and a decomposition of the space as a union of completely asymmetric and almost Π_n invariant sets are found. On the basis of these facts, in [9], results of Paul Erdös on the number of solutions of some linear equations in R^n are essentially generalized and some other significant results are also obtained.

The second part of the second series of works of his first stage scientific activities consists of 9 papers [10–18]. Therein Shalva Pkhakadze continues research in the same direction and obtains deep results. One should especially note the paper "A general method of construction" [18], where a general method is elaborated for effective construction of such subgroups of the Abelian group of real numbers which have null measure and a continual set of symmetric residue classes. This method enables one to construct an infinite quantity of individual instances of such subgroups. It is proved that the cardinal number of the set of such subgroups is 2^n. Moreover, it is shown that each of such subgroups has a quite paradoxical property – the set obtained from it by two-times contraction can be decomposed into the continual quantity of sets congruent to it. There arises a natural question on the place of such sets among effective sets constructed by Lusin's school. Namely, it is very interesting to find out whether some of them lay out of the class of projective sets.

At the end of this part of the publication we are quoting an opinion of Academician Pyotr Novikov, which gives a better idea about the results of this series of works of Shalva Pkhakadze and which is mainly concentrated on his dissertation work, because he was one of the official opponents of this dissertation: *"In the theory of the Lebesgue measure there arose a series of problems among which the main one is the problem posed by Sierpinski on the existence for any extension of the Lebesgue measure of another extension which would be an extension of the first one. In the measure theory the problem of extension is of the fundamental ones and it naturally has been arising a great interest.*

However, there was only a little progress towards solution of the posed problems on the extension of the Lebesgue measure. As a matter of fact,

only separate examples of such extensions were obtained. The dissertation under consideration is a substantial forward step in the theory, and moreover, I would say that it shifted the theory away from the dead point. The numerous results of the author can be divided into the following groups:

1. *Finding of general methods of extension of the Lebesgue measure and of extension of solvable classes of sets;*

2. *Generalizations of the measure in the sense that the usual requirements of invariance are replaced by the requirement of preservation of the measure under one-to-one transformations of the space forming a group.*

3. *Investigation of the structure of the measure by its decomposition into a sum of measures and establishing of a canonical decomposition.*

4. *Application of the methods developed by the author to problems lying outside of the measure theory.*

The first circle of the named results forms a rather general theory of extension of the Lebesgue measure. In these investigations the author introduces a series of important notions such as of absolutely null set, of a proper almost invariant set, etc. The germs of some of these notions can be found in works of previous authors. In the dissertation, however, they are developed at the full extent, they are systematically studied, and there is no doubt that in the future they will be repeatedly used. The theory of measure extension created in the dissertation would remain incomplete if therein there would be no progress in the direction of solving the above-mentioned problems. The central one is the problem of Sierpinski on existence of a nonextendable measure. In the dissertation the following answer is given to this problem: every Mn-measure is extendable if there is no unattainable cardinal number which would be less than continuum." Analyzing this answer, Pyotr Novikov further writes: *"I think that the problem of Sierpinski is solved fully enough. The author shows that if one adds to the axioms of the Mn-measure any of two quite natural axioms (A) and (B), then for such measures the problem of Sierpinski is solved completely without additional hypotheses. Likewise, without any additional hypothesis can be proved the extendability of any solvable class."* Finally, Pyotr Novikov concludes: *"The work under consideration is a substantial advance in one of the important areas of the set theory. Therein powerful methods are created which made possible to overcome serious difficulties."*

In addition to the above mentioned we underline that in 1960, in "Referential Journal of Mathematics" (N11456) famous Russian mathematician Vladimir Abramovich Rokhlin appreciates Shalva Pkhakadze's dissertation

as follow: *"The work is a fundamental research dedicated to the investigation of invariant extensions of the Lebesgue measure. It is known that such continuations exist, but up to now this knowledge was very scarce and consisted mainly of separate examples. The author has managed to prove a series of structural and fundamental theorems on them."*

Also, it must be mentioned too, that Alexey Lyapunov, also very famous Russian mathematician, in his review about the dissertation has written: *"The work is a bright event in the set theory"*.

3.2 Mathematical Logic

Despite his important results in Measure Theory and not young age, in 1967-1969, Shalva Pkhakadze changes his scientific sphere and begins to study Mathematical Logic. Of course, it was not easy for him to change scientific sphere, however considering the necessity of founding and developing a very important field of mathematics – mathematical logic in Georgia, he decided to make this step. Thus, it can be said that the main result of Shalva Pkhakadze's non-standard career is the fact that together with other Georgian mathematical schools, the Georgian school of mathematical logic exists and develops till todays.

All these would not exist, if there were no significant results in his scientific activity at the second stage, stage of mathematical logic [19–26]. Namely, in his work "One Example of Intuitively Computable and Everywhere Defined Function and Church's Thesis" [24], published in 1984, where he described an intuitively computable everywhere defined function, which according to him is not recursive, have casted doubt on the Church's thesis, which was indubitable for that time.

In the mentioned work the author, using an original method called by himself the "complex diagonal method", constructs an intuitively computable everywhere defined function f and studies its properties. On the basis of these properties, the author, as a hypothesis, expresses his full belief that, contrary to Church's thesis, the function f is not recursive.

At that stage of development of mathematical logic and algorithm theory, when the work was published, a claim against the Church's thesis can be considered as a purely speculative act, because of the point, that almost every logician has considered as a doubtless fact impossibility to solve problem of Church's thesis. Therefore, there was no risk in proposing the hypothesis on invalidity of Church's thesis, but in the case of the author's hypothesis, the situation is completely different, because it was very audacious and risky to propose a hypothesis, that a concrete intuitively computable function f is not recursive.

The point is that one of the reasons in favor of the Church thesis reads as

follows: for any intuitively computable function constructed up to now the proof of its recursiveness was easily found, and the belief that this will be so in the future is so strong that many authors don't consider it necessary to seek for the proof of recursiveness of a concrete intuitively computable function thinking that if needed this will be done without any difficulty. In this case it must be noted that there were attempts to prove the recursiveness of the above-mentioned function f and thereby the invalidity of Pkhakadze's hypothesis. Moreover, there were announced claiming the contrary, i.e. claiming that the function f, defined in Shalva Pkhakadze's above mentioned work, is recursive, though the proof of recursiveness of Pkhakadze's function is not published anywhere yet. Even more, nowadays, attitude of specialists towards Church's Thesis has radically changed and often there are published papers with disclaiming considerations about it, which makes correctness of the opposite thesis – Pkhakadze's thesis more trustworthy. However, this is not his main result in mathematical logic. Namely, refusing such an important thesis as Church's thesis, cannot be compared to what notation theory gave to mathematical logic [20]– [26]. Thus, with Shalva Pkhakadze's fundamental monograph "Some Problems of Notation Theory" [21] was founded the notation theory, which gives formal developing ability to the Frege's and Hilbert's classical formalism.[3] Obviously, taking into account the goals and the necessity of further development of formalistic approaches, this novelty is fundamentally significant.

Thus, as it was mentioned, the basic results of the second stage of scientific activities of Shalva Pkhakadze are mainly set forth in the monograph "Some Problems of the Notation Theory". Therein rules for definition of contracted symbols are introduced and studied. The monograph is in mathematical logic, but it is similar to the above considered one by the fact that from the very beginning a particular attention is paied to the development of the fundamental concepts. Thus, the conceptual aspect of the work is to be particularly stressed.

Thus, the development of the formal mathematics has led us to significant discoveries and results. They belong not only to the foundations of mathematics but also to its special fields. At the contemporary stage of the development of mathematics it is doubtful to obtain such results without using methods of formal mathematics. Therefore, for development of the whole mathematics it is most important to improve the methods of the formal mathematics. In spite of this, the majority of fields of mathematics is developing on the nonformal level. This is due to the fact that it is difficult to master in the methods of formal mathematics since valuable expositions of formal mathematical systems are not available. To change this existing

[3] We will consider this issue in more detail at the end of the article.

expositions, deep scientific research is needed.

One of the essential peculiarities of formal mathematical systems is the limitation of the alphabet of the theory and its reduction to minimum. This makes the class of the propositions of the theory available for the examination as a whole, the concept of the proof can be defined exactly, a series of mathematical problems can be formulated clearly and their solving is facilitated. Modern mathematical theories which are developing on informally level are influenced by methods of formal mathematical theories, they reduce their alphabet to minimum and associate formal mathematical concepts to the intuitive ones. For instance, the concept of algorithm and other concepts, related with it, are introduced in such a way.

Although the limitation of the alphabet of the theory is very advantageous theoretically, it causes substantial difficulties. Namely, one has to introduce extremely long forms (formulas and terms), and it is practically impossible to write them down and perceive their meaning. To get over these difficulties there are introduced contracted, in other words, abridging symbols.

In the case of a rather rich theory (as the set theory), the rules of definition of contracted symbols are just slightly restricted or are not restricted at all. So it is impossible to establish general properties of abridging symbols and abridged forms. Because of this the principle is adopted according to which an abridged form is considered as the form it denotes. But the complete realization of this principle is impossible due to the above-mentioned difficulties.

This implies the necessity of changing of the above principle and adopting of one which would make possible to draw conclusions about forms when we are operating directly on their abbreviations. This, in its turn, gives rise to the need for general rules of operations on abridged forms. To this aim, it is necessary to establish general properties of abridging symbols and abridged forms by restricting their intuitional concepts by exact mathematical notions, i.e. by defining the notion of the abridging symbol on the basis of restriction of the rules of their introduction. Moreover, for the aims of developing a formal mathematic to found a rational solution of the here posed problem is under the obligatory needing. In the considered work a rather rational system of definition rules is found for contracted (abridging) symbols in the cases of classical formal and unformal mathematical theories. By the rationality of a system of rules is meant the fact that, on the one hand, it is so general that with its aid almost all symbols used in classical formal theories can be derived, and on the other hand, the symbols introduced according to these rules possess properties rich enough to ensure a considerable liberty in operating on abridged forms. Namely, by means of contracted symbols mathematical judgments can be transferred

into the enlarged theory using natural general laws.

The construction of a mathematical theory (formal or informal) with a limited alphabet and rational (in the above sense) rules of introducing of contracted symbols is of great theoretical and practical importance. It makes possible to study the main theory using effectively its various extensions obtained by adding contracted symbols; it reveals deep connections between various mathematical theories. It is practically necessary for the automation of mathematical research and the creation of special systems processing mathematical texts.

Moreover, in the work algorithmic processes of reconstruction of the form by its abbreviation are studied. In a rather naturally introduced class of algorithms, an algorithm is found for such a reconstruction with a minimal number of steps. Various numerical characteristics of contracted symbols and contracted forms are also studied and algorithms are found for their evaluation. It should be especially noted that a carefully thought out system of terms and concepts is used. Namely, there are introduced various types of abridging symbols and abridged forms, various reconstructing processes, various types of applications of definition of abridging symbols, various types of particular cases of abridging symbols, and so on.

It is clear from the above said that Shalva Pkhakadze's fundamental monograph "Some Problems of the Notation Theory" is a significant achievement in mathematical logic and a deep research in this field. It deals with the foundations of mathematics and has a great theoretical and practical value. It can be successfully used while writing monographs in those fields of formal and unformal mathematics which use a limited alphabet, namely, abridging (i.e. contracted) symbols of the types I-IV, II' and IV' are to be used. For them important properties are established facilitating to carry out mathematical reasoning and making possible to change intuitive judgment by mathematically exact ones.

The further study of the problems dealt within the monograph in the direction indicated by the author is of great importance. It should be also especially noted that the notation theory – a well formulated independent theory has been already created on the basis of the already obtained results.

Finally note that the results of the monograph arise interest not only of theoreticians, but also of specialists of applied mathematics working in the theory of automata. Presently an intensive work is being carried out aiming at the automation of mathematical research, the creation of automatic systems for processing mathematical texts. There exist rather convincing reasons according to which mathematical texts of only those mathematical theories can be processed with the help of automatic systems which use a restricted alphabet and are created on the basis of the well-developed notation theory. – To illustrate better the results of this work,

below, we quoted some very famous Russian scientists.

Vladimir Mefodievich Matrosov: *"From our point of view, the representation of knowledge and the economy of "thinking" of a computer (in particular, a hierarchical organization of concepts and theories in the data structure of a computer and processing of mathematical texts) needs the application of results of the notation theory. In Shalva Pkhakadze's monograph "Some Problems of the Notation Theory" the first attempt is given of systematic development of the named theory. Here is considered theory in language of rather general type without concretizing axioms and are imposed restrictions on the rules of introducing of contracted symbols. This makes it possible to operate on contracted forms instead of forms they denote. General properties of contracted forms are studied and operations on them are defined. Also, here, the rules of introducing of contracted symbols are chosen so that rather good properties of contracted forms be obtained, which guaranteeing the homomorphism of the algebra of operations on contracted forms into the algebra of operations on forms of main theory. It is clear, that this is a very important result in the sence of developing and improving general formal approaches."*

Yuri Ivanovich Yanov: *"Although abridging notations, in other words, contracted notations have been constantly used in constructing mathematical theories, systematic study of problems arising in this connection was first made in the Shalva Pkhakadze's fundamental monograph "Some Problems of the Notation Theory". Therein a theoretical basis was laid down for introducing and using abridging notation in mathematical theories. The actuality of this problem is due to the fact that development and exposition of any, even rather simple, mathematical theory is practically impossible without introducing additional symbols, axioms and definitions playing a role of abridging notation. In this connection a series of important problems arise. Among them are conservativeness of the extension, i.e. whether the set of theorems which can be formulated in the language of the original theory remains unchanged, the problem of relation between properties of abridged and main forms and etc. The work of Shalva Pkhakadze, by means of a certain classification of abridging notation, makes it possible to obtain rather simply answers to mentioned problems. The basic ideas of the work were further developed in the works of the pupils of Sh. Pkhakadze. As a concrete application, the works in this direction may be used in problems of constructing an artificial intelligence system."*

Oleg Borisovich Lupanov: *"Professor Shalva Pkhakadze is a leading specialist in mathematical logic and set theory. During the last fifteen years, he has been developing an important direction in mathematical logic, connected with the notation theory. This direction formalizes the procedure of using abridging notation and is essential in constructing artificial mathematical*

languages, in problems of automated deriving of theorems, etc."

Sergey Vsevolodovich Yablonsky: *"Shalva Pkhakadze is developing an original direction related to the theory of formal systems. In constructing mathematical theories, we usually start from a limited alphabet and therefore we have to use abridged i.e. contracted notation not contained within the framework of the initial theory. Shalva Pkhakadze gives a common approach to operations with contracted symbols and forms, that is of great importance for constructing mathematical theories according to general princeples of the mathematical logic and formal mathematics."*

Sergey Mikhailovich Nikolsky: *"The monograph not only opens an actual direction, but also contains large possibilities in the area of automated processing of mathematical texts."*

Thus, as it was already underline above, that the main contribution in mathematical logic, which was done by Shalva Pkhakadeze, is the theory of formal notation, shortly, notation theory founded by his fundamental monograph work "Some Problems of Notation Theory" and, also, due to the works of his pupils and followers. Here, it must be noted that, the scientific society is still in the process of comprehending the above-mentioned fundamental significance of Shalva Pkhakadze's notation theory. This might be caused by the fact that the results ahead of the time need more time to be understood.

However, it must be noted as well, that Shalva Pkhakadze's notation theory already significantly contributed in the development term rewriting systems, which, in turn, is a very important theory for the aims of formalization of mathematical and natural languages and theories. Namely, in the Femke Van Raamsdonk's doctoral thesis "Confluence and Normalization for Higher-Order Rewriting" Shalva Pkhakadze's notation theory is already appreciated as one of first sources of the higher-order rewriting systems. In particular, at the first stage of the development of the theory of the term rewriting systems, it was possible to compute only expressions without bound variables, but on the basis of Shalva Pkhakadze's notation theory and on the base of the different types of substitution operators, which are defined in this theory, there was created an expression reduction system, on the basis of which term rewriting systems were equipped with possibility compute expressions with bound variables too.[4] This fact, and it is clear, is a very clear confirmation of the high theoretical significance of

[4]Expression reduction systems as initial theory was created in Tbilisi by Zurab Khasidashvili, which defended his doctoral thesis in Tbilisi State University in 1991 by leadership of Shalva Pkhakadze (for more details see publications: 1. Femke Van Raamsdonk, Confluence and Normalization for Higher-Order Rewriting, Vrije University, Amsterdam, 1-212, 1996; 2. Z. Khazidashvili, Expresion reduction systems. Proceedings of VIAM TSU, Vol. 36, 200-220, 1992).

Shalva Pkhakadze's notation theory. However, below, we will try to present the deeper conceptual meanings of this theory, but beforehand we underline that below presented views are mainly based on the already sufficiently proven factual circumstance, that Georgian and Mathematical languages, in general, are languages of a one and same types. This very important factual circumstance is strengthened by the fact that almost any token of the Georgian language i.e. any word, any morpheme, and any punctuation symbol of the Georgian language are describable as a some kind symbol of the $\mathfrak{J}$ sufficiently general mathematical language, which was defined by Shalva Pkhakadze in his notation theory and which are very shortly overviewed below.[5]

Thus, one more time, we underline, that in general, notation theory is the system of the formal rules of the formal extensions of the formal theories and languages. At the same time, we call a formal language and theory without possibility to be formally extended, respectively, with possibility to be formally extended as a formally non developable, respectively, as a formally developable language and theory.

In notation theory, which is formed on the basis of Shalva Pkhakadze's $\mathfrak{J}$ sufficiently general mathematical language, are described different types of formal extension rules, which are called as contracted rules. With the help of these contracted rules one can extend any $\mathfrak{J}$ sufficiently general mathematical language in almost any case of extension needed. This means that notation theory gives us scientifically founded understanding of $\mathfrak{J}$ formally developable mathematical languages.

At the same time, based on the $\mathfrak{J}$ sufficiently general mathematical language there is defined $\mathfrak{J}$ sufficiently general mathematical theory and any $\mathfrak{J}$ sufficiently general mathematical theory together with the above mentioned contracted rules gives us scientifically founded understanding of $\mathfrak{J}$ formally developable mathematical theories.

In addition, if $\mathfrak{J}^*$ is any extension of any semantically completely under-

[5]The views, which will be presented below, are a result of the researches, which permanently goes on with leadership of Konstantine Pkhakadze. The researches have begun in 1999; since 2002 until 2010, researches were going in confine of the State Priority Program "Free and Complete Inclusion of a Computer in the Georgian Natural Language System" and, from 2010 till todays it goes on in confine of the long-term project "The Technological Alphabet of the Georgian language" of the center for the Georgian language technology of the Georgian technical university. As a result, more than 200 papers were published by K. Pkhakadze in diferent years together with diferent co-authors. They are: M. Ivanishvili, E. Soselia, L. Lekiashvili, G. Chankvetadze, L. Tibua, I. Beriashvili, R. Skhirtladze, K. Gabunia, N. Buadze, G. Kandelaki, V. Pkhakadze, T. Esitashvili, B. Tskhadadze, G. Chichua, A. Maskharashivili, L. Abzianidze, N. Pkhakadze, N. Vakhania, N. Labadze, B. Chikvinidze, L. Gurasashvili, M. Beriasvili, M. Chikvinidze, D. Kurtskhalia, S. Shinjikashvili, Sh. Malidze, C. Demurchev and N. Okroshiashvili.

stood, i.e. interpreted i.e. consistent $\mathfrak{J}$ formally developable mathematical theory and if this extension was made with the help of above overviewed contracted rules Pkhakadze, then $\mathfrak{J}^*$ is also a semantically completely understood, i.e. interpreted i.e. consistent theory.

Above, non-formally and very shortly we have described the general semantic program of foundation of mathematics, which is elaborated by Shalva Pkhakadze and which is the main result of his notation theory.

Thus, as a conclusion of above-mentioned, we underline the next: since Frege's mathematical language is Shalva Pkhakadze's $\mathfrak{J}$ sufficiently general mathematical language and Hilbert's mathematical theory is Pkhakadze's $\mathfrak{J}$ sufficiently general mathematical theory, we can conclude that Shalva Pkhakadze's notation theory gives formally developable abilities to the formally non developable Frege's mathematical language and Hilbret's mathematical theory.

It is clear, that above shortly described $\mathfrak{J}$ formally developable mathematical languages and $\mathfrak{J}$ formally developable mathematical theories of Shalva Pkhakadze give very fruitful new possibilities to construct non-simple intelligence systems. Also, they give us scientifically founded understanding on the human's lingual nature and lead us to formulate a particular and a general thesis of the Georgian Language:[6]

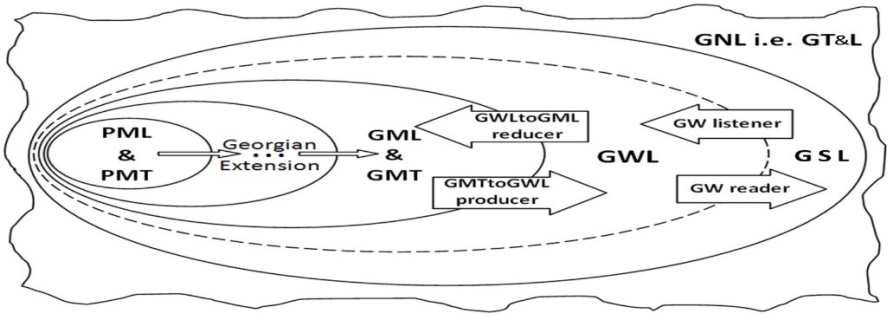


Figure 1: Gradual extension of PML to GT&L via GERs

A particular thesis of the Georgian language: Natural Georgian language is a result of that formal extension of the Primary Mathematical Language (PML), whose extension is made with the help of Contracted Rules (CRs) Shalva Pkhakadze's type, which we have called Georgian Extension Rules (GER).

Natural Georgian Thinking and Language System (GT&L), shortly

[6]The fact that we applied the Shalva Pkhakadze's notation theory to the Georgian language is based on the above underlined and already sufficiently confirmed fact, that Georgian language is a language of the mathematical type.

Georgian Natural Language System (GNL) is a result of that formal extension of the PMT, which is made with the help GERs (see, above, figure 1, where GML, respectively, GMT abbreviates the Georgian Mathematical Language, respectively, Georgian Mathematical Theory, which together with PML, respectively, PMT is a some $\mathfrak{J}$ formally developable mathematical language, respectively, $\mathfrak{J}$ formally developable mathematical theory of Shalva Pkhakadze's type, which exists in the Georgian-speaking peoples by nature in an innate way).

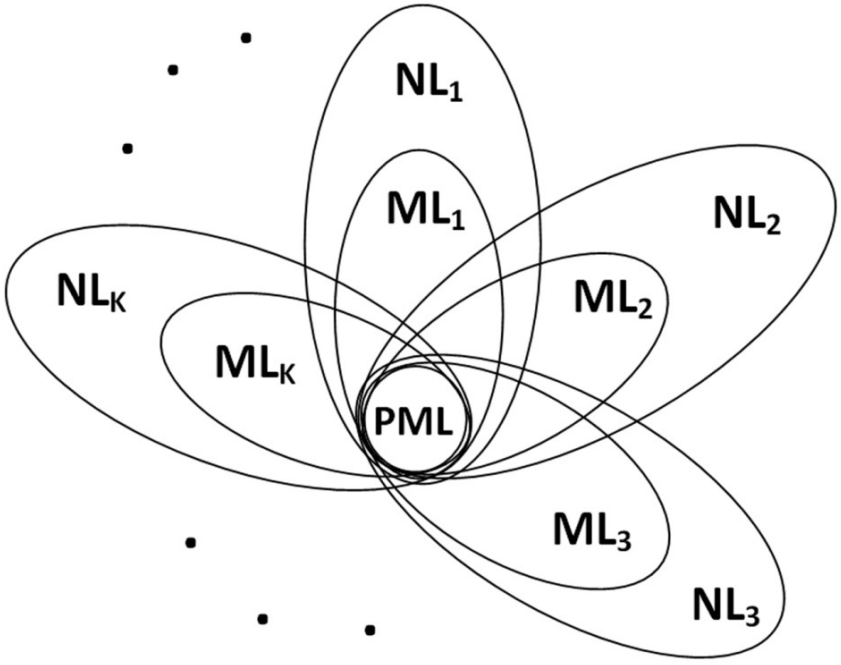


Figure 2: Graphical representation of the relation between PML and NLs

A general thesis of the Georgian language: According to us any Natural Language (NL) is a result of step by step extensions of the Primary Mathematical Theory (PMT), alphabet of which is called Primary Mathematical Language (PML) and axioms, inference and extension rules of which are called Primary Mathematical Concepts (PMC) (see, above, figure 2, where by $\mathbf{NL_K}$ is denoted **K**-th **NL** and by the $\mathbf{ML_K}$ the Mathematical Language, which stands between the PML and $\mathbf{NL_K}$). At the same time, we declare that the PML and PMCs are naturally and universally inborn in all humans and in confines of the PMT they act automatically, i.e. instinctually.[7]

[7]Thus, it can say, that we share Pascal's insight on existence of selfunderstandable

To summarize, it must be underlined, that the above presented new lingual views, which together with notation theory and general semantic program of Shalva Pkhakadze are mainly based on the very clear mathematical specifics of the Georgian language, differ with other today existing lingual views and gives us a new understanding of genesis of different natural languages. This new understanding is clearly shown on the above presented diagram (see figure 2), which we call "lingual flower with mathematical heart". Thus, on the basis of all above presented it becomes clear, that for us:

1. The primary mathematical language, which is naturally inborn in all humans, is the parent of all natural languages i.e. parent of all native languages, that means that this primary mathematical language is a native language for all humans. In other words, Shalva Pkhakadze's theory of formal notations dictates that mathematics together with linguistics is a part of natural science;

2. Shalva Pkhakadze's theory of formal notations changes a paradigm in formal mathematical and linguistic researches, because this theory dictates us, that for complete formal and technological foundation of the mathematics and natural languages, firstly, it is necessary to recover the deep subconscious part of natural languages, in other words, of conscious natural languages. In other words, Shalva Pkhakadze's theory of formal notations dictates that for construction of artificial intelligence system almost completely knowing a natural language and general mathematical skills it is necessary complete recovering of the primary mathematical language and primary mathematical concepts on the basis of which is constructed the primary mathematical theory, which, in turn, as primary and universally lingual knowledge is naturally inborn in all humans and gives birth to all different natural languages, in other words, to all different human languages.

References

1. Pkhakadze, Sh. On repeated integrals. *Bulletin of Academy of Sciences of GSSR*, Vol.14, 3-10, 1953.

words, because of – according to his insight – in the case of nonexistence of such words, it would be impossible to understand as them, as well as all other words and phrases. Moreover we share completely also next old bible view that "in the beginning was the word" i.e. language and "all things were made by him" i.e. by language "and without him was not any thing made that was made", and, accordingly, we know, that "in him" i.e. "in language" is and "was life", because of for us life is a whole of divisible alike units (cells) that are related with each other informatically, in other words, lingually.

2. Pkhakadze, Sh. On repeated integrals. *Proceedings of the TMl*, Vol 20, 167-208, 1954.

3. Pkhakadze, Sh. Absolutely null set. *Bulletin of Academy of Sciences of GSSR*, Vol 15, 201-205, 1954.

4. Pkhakadze, Sh. On various definitions of the concept of absolutely null sets. *Bulletin of Academy of Sciences of GSSR*, Vol. XV (8), 489-496, 1954.

5. Pkhakadze, Sh. Immeasurable absolutely null sets, their countable sums and proper almost invariant sets. *Bulletin of Academy of Sciences of GSSR*, Vol. 16, 343-350, 1955.

6. Pkhakadze, Sh. Extensibility of resolvable classes. *Bulletin of Academy of Sciences of GSSR*, Vol. 16, 761-768, 1955.

7. Pkhakadze, Sh. On the continuability of a measure. *Bulletin of Academy of Sciences of GSSR*, Vol. 17, 769-776, 1956.

8. Pkhakadze, Sh. Some sentences equivalent to the continuum hypothesis. *Proceedings of USSR Academy of Sciences*, 111, 299-300, 1956.

9. Pkhakadze, Sh. The theory of the Lebesgue measure. *Proceedings of TMI*, Vol. 25, 3-272, 1958.

10. Pkhakadze, Sh. One general method for constructing an effective example of a continual subgroup of an abelian group of real numbers. *Proceedings of TMI*, Vol. 29, 103-119, 1963.

11. Pkhakadze, Sh. Measure Decomposition. *Bulletin of Academy of Sciences of GSSR*, Vol. 31, 15-22, 1963.

12. Pkhakadze, Sh. Decomposition of various types measures. *Bulletin of Academy of Sciences of GSSR*, Vol. 31, 521-527, 1963.

13. Pkhakadze, Sh. Measure Decomposition. *Proceedings of TMI*, Vol. 29, 121-145, 1963.

14. Pkhakadze, Sh. On the properties of continuity and discontinuity of measures. *Bulletin of GPI*, Vol. 1(97), 1-4, 1964.

15. Pkhakadze, Sh. The relationship between questions about the existence of some flat sets and the canonical decomposition of $\mathfrak{J}^n$-measures. *Bulletin of GPI* 1(99), 1-2, 1965.

16. Pkhakadze, Sh. Canonical decomposition and properties of continuity and discontinuity of measures. *Proceedings of TMI*, Vol. 34, 123-140, 1968.

17. Pkhakadze, Sh. Difficulties associated with the problem of the existence of a canonical decomposition of a given $\mathfrak{J}^n$-measure. *Proceedings of TMI*, Vol. 34, 141-154, 1968.

18. Pkhakadze, Sh. A general method of construction. *Proceedings of TMI*, 1969, Vol 29, 103-119

19. Pkhakadze, Sh. Application of the mathematical logic in the automata theory. *VIAM of TSU*, 1971.

20. Pkhakadze, Sh. On a class of contracted symbols. *VIAM of TSU*, 1-75, 1975.

21. Pkhakadze, Sh. Some Problems of the Notation Theory. *TSU*, 1-196. 1977.

22. Pkhakadze, Sh. Elements of the set theory and mathematical logic. *TSU*, 1-311. 1978.

23. Pkhakadze, Sh. Some Problems of the Notation Theory, *Proceedings of VIAM of TSU*, Vol. 11, 28-42, 1982.

24. Pkhakadze, Sh. One Example of the Intuitively Computable Everywhere Defined Function and Church's Thesis. *VIAM of TSU*, 1-70, 1984.

25. Pkhakadze, Sh. On mathematical theories. *VIAM of TSU*, Vol. 22, 24-32, 1993.

26. Pkhakadze, Sh. Quantifiers in mathematical analysis. *VIAM of TSU*, Vol. 22, 32-40, 1993.

27. Pkhakadze, Sh. A N. Bourbaki Type General Theory and the Properties of Contracting Symbols and Corresponding Contracted Forms. *Georgian Mathematical Journal*, Vol.26, Num.2, Kluwer academic publisher, 179-290, 1999.

28. Pkhakadze, Sh. Mathematical Logic – Foundations. *TSU* /Part 1(1-276), 1996/Part 2(1-268) 1999/Part 3(1-198) 2009.

A LOGIC WITH MEASURABLE SPACES FOR NATURAL LANGUAGE SEMANTICS *

Jean-Philippe Bernardy Rasmus Blanck
Aleksandre Maskharashvili

University of Gothenburg, Department of Philosophy, Linguistics and Theory of Science, Centre for Linguistics and Studies in Probability.
name.surname@gu.se

Abstract

We present a Logic with Measurable Spaces (LMS) and argue that it is suitable to represent the semantics of a number of natural language phenomena. LMS draws inspiration from several sources. It is decidable (like descriptive logics). It features Sigma spaces (like Martin-Löf type-theory). It internalises the notion of the cardinality (in fact, here, measures) of spaces (see [6]) and ratios thereof, allowing to capture the notion of event probability. In addition to being a powerful system, it is also concise and has a precise semantics in terms of integrals. Thanks to all these qualities, we hope that LMS can play a role in the foundations of natural language semantics.

1 Introduction

The ability of humans to reason under uncertainty has reflections within natural language where we find various lexico-syntactic constructions which allow us to express uncertain information. Moreover, we are able draw conclusions - make inferences under uncertainty. To give an adequate account to this crucial aspect of natural language, it has been long argued for employing probabilistic tools in defining semantics of natural language.

The question remains of which tool is best suited for the purpose. [7, 10] have proposed to use probabilistic programming languages.

In this paper, we propose to use instead a special-purpose language which aims at providing the most convenient interface for compositional semantics, while giving a fully precise semantics.

We call this intermediate language Logic with Measurable Spaces (LMS). We argue that LMS is suitable to represent the semantics of a number of natural language phenomena. LMS draws inspiration from several sources. It is decidable (like descriptive logics). It features Sigma spaces (like

*Supported by Swedish Research Council, Grant number 2014-39

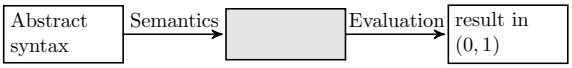


Figure 1: Overview of the parts of a complete probabilistic (bayesian) inference system

Martin-Löf type-theory). It internalises the notion of the cardinality (in fact, here, measures) of spaces (see [6]) and ratios thereof, allowing to capture the notion of event probability.

A fully-fledged probabilistic semantics will be comprised of several parts, shown in 1. In this paper, we focus on the interface between compositional semantics and evaluation *only*. Yet, to get a sense of how such a system is articulated, we present an example inference — remaining at a suitably abstract level:

$$\frac{\text{Most birds fly}}{\text{A few birds fly}}$$

We wish to compute the probability

$$P(\text{A few birds fly} \mid \text{Most birds fly})$$

and test if its value is closer to 1 than 0.

In order to do so, the natural sentences are first parsed, yielding abstract syntax trees. For example one can use the GF tool [11], but any tool which produces a syntax compatible with Montague-style categories would be suitable. The abstract syntax that we obtain for the premiss and the hypothesis could be written as follows.

$$P = many\ bird\ fly$$
$$H = aFew\ bird\ fly$$

The abstract syntax is then translated to LMS. This makes explicit their spaces, and the measures thereof. Using these features, the probabilities of all propositions of interest can be expressed precisely. To convert to this intermediate representation, we first must express our (lack of) prior knowledge about the common nouns, verbs, etc. present in the problem. To do so we gather the lexical items and introduce them as as random variables in the appropriate space. The premiss(es) are added as extra conditions, using a compositional semantics [2]. These conditions effectively update the distributions of the representations of lexical items, yielding a global space of situations Ω and assuming a proportion Θ_m corresponding to the

meaning of "most".

$$\begin{aligned}\Omega = [&bird : Pred\\ &fly : Pred\\ &p : \mathsf{measure}([x : Ind; b : bird(x); f : fly(x)])\\ &> \Theta_m \mathsf{measure}([x : Ind; b : bird(x)])]\end{aligned}$$

Note that to make the language more concise, we unify the language of spaces and the language of propositions — effectively we sample p over the space of proofs of the propositions. We leave here the space of predicates $Pred$ abstract: possible choices are spelled out by [3] and [4].

The truth value of the conclusion is then expressed as a probability measure of a proposition over the whole space Ω that we just defined, with a suitable proportion Θ_f for "few".

$$\begin{aligned}X = P_{\omega:\Omega}([x : Ind; b : \omega.bird(x); f : \omega.fly(x)])\\ > \Theta_f \mathsf{measure}([x : Ind; b : \omega.bird(x)])\end{aligned}$$

The convenient expression above can be turned into a mathematical expression using the semantics for spaces and probabilities (1). In our running example, the expression begins with integration over the spaces of predicates:

$$\sum_{bird:Pred} \sum_{fly:Pred} \frac{\mathbf{1}(P \wedge Q)}{\mathbf{1}(P)}$$

where the conditions P and Q are given by

$$\begin{aligned}P &= \mathsf{measure}([x : Ind; b : bird(x); f : fly(x)])\\ &\quad > \Theta_m \cdot \mathsf{measure}([x : Ind; b : bird(x)])\\ Q &= \mathsf{measure}([x : Ind; b : bird(x); f : fly(x)])\\ &\quad > \Theta_f \cdot \mathsf{measure}([x : Ind; b : bird(x)])\end{aligned}$$

The integrations and measures get further expanded if $Pred$ is made concrete. But, we can already see that $P \wedge Q = P$ if $\Theta_m > \Theta_f$, and in this simple case the integral therefore evaluates to 1, meaning that the inference is (stochastically) certain.

However, in the vast majority of cases, integrals are not computable symbolically. This would happen for example if we do not choose a fixed value Θ_m or Θ_f, but rather used random variables. In this kind of situation, one typically resorts to simulated sampling — using Monte Carlo methods (see 2.3).

$A, B, \ldots ::= \mathsf{IsTrue}(\phi)$	types of proofs
$\mid \Sigma(x : A)B$	sigma type, generalised pair
$\mid \mathsf{Distr}(d)$	Real-valued base distribution with finite support
$\mid \{e \mid x : A\}$	image of A under $\lambda x.e$
$\phi, \psi, e ::= x$	variable
$\mid \phi \wedge \psi$	
$\mid e_1 \leq e_2$	
$\mid \pi_1(e) \mid \pi_2(e)$	projections
$\mid op(e_i)$	arithmetic operators
$\mid \Diamond$	uninformative object
$\mid \mathsf{measure}(A)$	internalisation of measure
$\tau, \sigma ::= Unit \mid Bool \mid \mathbb{R} \mid \tau \to \sigma \mid \tau \times \sigma$	

Figure 2: Syntax of LMS

2 Logic with Measurable Spaces

In this section we describe a Logic with Measurable Spaces (LMS). LMS is the representation language connecting parse structures to mathematical expressions of probabilities. We use it to describe the meaning of inferences. As a first approximation, one can see LMS as a precise formalisation of informal notations used when manipulating logical expressions involving random variables. Readers familiar with these concepts can skip this section on first reading. But it will be helpful for understanding subsequent definitions.

The syntax of LMS is comprised of two categories: spaces (A, B, C, etc.), and expressions (e or ϕ, ψ for boolean-valued expressions.)

The main objects of interest are *spaces*. Every space has two aspects: an underlying *type* and a probability distribution over it. The types are formed by the unit type, booleans, reals, functions, and products.

In LMS, types are used as in a programming language, to verify that nonsensical expressions are disallowed. We do not follow the tradition of intuitionistic logic in that we ignore the inhabitants of types. Specifically, we do not consider types as propositions, via the Curry-Howard isomorphism. LMS does not include quantification over all types, nor over all spaces. Instead, the densities of spaces are their logical content. Before turning to

$$\frac{\Gamma \vdash \phi : Bool}{\Gamma \vdash \mathsf{IsTrue}(\phi) : \mathsf{Space}\, Unit} \qquad \frac{\Gamma \vdash A : \mathsf{Space}\,\tau \qquad \Gamma, x : \tau \vdash B : \mathsf{Space}\,\sigma}{\Gamma \vdash \Sigma(x : A)B : \mathsf{Space}\,(\Sigma(x : \tau)\sigma)}$$

$$\frac{\Gamma \vdash e_i : \mathbb{R}}{\Gamma \vdash \mathsf{Distr}(d_1[e_i]) : \mathsf{Space}\,\mathbb{R}} \qquad \frac{\Gamma, x : \tau \vdash e : \sigma \qquad \Gamma \vdash A : \mathsf{Space}\,\tau}{\Gamma \vdash \{e \mid x : A\} : \mathsf{Space}\,\sigma}$$

$$\frac{\Gamma \vdash e_1 : \tau \qquad \Gamma \vdash e_2 : \sigma[e_1/x]}{\Gamma \vdash (e_1, e_2) : \tau \times \sigma} \qquad \frac{\Gamma \vdash e : \tau \times \sigma}{\Gamma \vdash \pi_1(e) : \tau} \qquad \frac{\Gamma \vdash e : \tau \times \sigma}{\Gamma \vdash \pi_2(e) : \sigma}$$

$$\Gamma \vdash \Diamond : Unit \qquad \frac{\Gamma \vdash \phi : Bool \qquad \Gamma \vdash \psi : Bool}{\Gamma \vdash \phi \wedge \psi : Bool} \qquad \frac{\Gamma, x : \tau \vdash e : \sigma}{\Gamma \vdash \lambda x.e : \tau \to \sigma}$$

$$\frac{\Gamma \vdash e_0 : \tau \to \sigma \qquad \Gamma \vdash e_1 : \tau}{\Gamma \vdash e_0(e_1) : \sigma} \qquad \Gamma \vdash true : Bool \qquad \Gamma \vdash false : Bool$$

$$\frac{\Gamma \vdash e : Bool}{\Gamma \vdash \mathbf{1}(e) : \mathbb{R}} \qquad \Gamma \vdash k : \mathbb{R} \qquad \frac{\Gamma \vdash e_i : \mathbb{R}}{\Gamma \vdash op(e_i) : \mathbb{R}}$$

Figure 3: Typing rules for LMS. In the above *op* stands for an arbitrary arithmetic operator or arbitrary arity, with e_i being its operands. Similarly, we list only only one logical connective ($\wedge$); others follow the same pattern.

density we give a brief overview of LMS typing and its consequences. We use two judgments. First, the judgment $\Gamma \vdash e : \tau$, which is the standard typing judgment for terms in the simply typed lambda calculus. We call Boolean-valued expressions *propositions*, so if $\Gamma \vdash \phi : Bool$ holds, it means that ϕ is a proposition in context Γ. Second, the judgment $\Gamma \vdash A : \mathsf{Space}\,\sigma$ expresses that A is a space over the ground type σ, in a context Γ.

Because expressions are simply typed, they inherit the usual normalisation properties of typed lamdba terms [1]. Any closed term of type $\mathbb{R}$ is a real number.

We now focus on spaces and distributions over them. We have four basic space constructions:

1. Given a distribution with n parameters $d(x_1, \ldots, x_n)$, we have the space $\mathsf{Distr}(d_1(e_1, \ldots, e_n))$ (each of the parameters can be assigned any real-valued expression).

2. We can construct a space whose density is 1 when a proposition ϕ is

true and 0 otherwise. It is written $\mathsf{IsTrue}(\phi)$.

3. We can construct sigma spaces. Given a space A and a space $B[x]$, we can write $\Sigma(x : A)B[x]$ for the the sigma space.

4. We can take the image of a space A under a function f. This space is written $\{f(x) \mid x : A\}$. (In fact we generalise to and allow any expression dependent on x instead of just $f(x)$.)

These constructions are listed in 3.

Formally, we do not manipulate densities directly, thus avoiding theoretical difficulties, in particular for $\{f(x) \mid x : A\}$. Instead, we generalise the notion of integration so that it does not just apply to distributions, but to arbitrary spaces. For this purpose we use the symbol $\sum$, as it is a natural extension of the summation operator.

Definition 1. If $\Gamma \vdash A : \mathsf{Space}\,\alpha$ and $\Gamma, x : \alpha \vdash e : \mathbb{R}$, we define $\sum_{x:A} e$ (which can be read as the integral of e for x ranging over A), by induction on A:

$$\sum_{x:\mathsf{Distr}(d)} e = \int_{\mathbb{R}} \mathrm{PDF}(d, x) \cdot [\![e]\!] dx$$

$$\sum_{x:\mathsf{IsTrue}(\phi)} e = \mathbf{1}([\![\phi]\!]) \cdot [\![e[\Diamond/x]]\!]$$

$$\sum_{z:\Sigma(x:A)B} e = \sum_{x:A} \sum_{y:B} e[(x, y)/z]$$

$$\sum_{y:\{e \mid x:A\}} e_2 = \sum_{x:A} e_2[e/y]$$

Definition 2. (Evaluation of expressions) The value of an expression e is written $[\![e]\!]$ and defined by induction on the structure of expressions, as is standard in the lambda calculus. We know that evaluation terminates because of our type-system. The only case that merits attention is the evaluation of $\mathsf{measure}(A)$, which is specified by the following equation:

$$[\![\mathsf{measure}(A)]\!] = \sum_{x:A} 1$$

The expression $\sum_{x:A} 1$ integrates over the whole space of the constant value 1, thus "counting" the elements of that space. Therefore it is the *measure* of the space A. Overloading the notation, we also write $\mathsf{measure}(A)$ for the measure of the space A as a meta-theoretical expression (not an LMS expression), with the same definition.

Definition 3. (Expected value) We define the *expected value* of e for a random variable x distributed in A as follows:

$$E_{x:A}(e) = \frac{\sum_{x:A} e}{\mathsf{measure}(A)}$$

Remark:

$$E_{z:(\Sigma(x:A)B)}(e) = E_{x:A}(E_{y:B}(e[(x,y)/z]))$$

Notation:

$$E_{x_0:A_0,\ldots,x_n:A_n}(e) = E_{x_0:A_0}(\ldots E_{x_n:A_n}(e))$$

Finally, we can define the *probability* of a proposition ϕ over a random variable x ranging in A as the proportion of (the measure of) the space A where ϕ holds.

Definition 4.

$$P_{x:A}(\phi) = E_{x:A}(\mathbf{1}(\phi))$$

An equivalent definition is the following:

$$P_{x:A}(\phi) = \frac{\mathsf{measure}(\Sigma(x:A)\mathsf{IsTrue}(\phi))}{\mathsf{measure}(A)}$$

In general, for probabilistic inference, we define a space of possible situations Ω, and evaluate the expected truth value of some proposition ϕ over this space. The space Ω typically has a complex structure.

We now verify that $P_{x:A}(\phi)$ satisfies the expected properties of probabilities, starting with the following lemma:

Lemma 1. $\sum_{x:A}$ *is a linear operator*

(i) $\sum_{x:A}(k \cdot t) = k \cdot \sum_{x:A} t$ *if k does not depend on x*

(ii) $\sum_{x:A}(t+u) = \sum_{x:A} t + \sum_{x:A} u$

Proof. By induction on the structure of A. □

When a space A has zero measure, the probabilities over it are undefined. Otherwise, the Kolmogorov laws of probability are respected. It is easy to verify that any probability is positive, and that the probability of *true* is 1. The last law (in its finite variant) needs a bit more work, and its proof follows.

Theorem 1. *If $\phi \wedge \psi = false$, then*

$$P_{x:A}(\phi \vee \psi) = P_{x:A}(\phi) + P_{x:A}(\psi)$$

Proof.

$$\begin{aligned} E_{x:A}(\phi \vee \psi) &= \sum_{x:A} \mathbf{1}(\phi \vee \psi) && \text{by def.} \\ &= \sum_{x:A} (\mathbf{1}(\phi) + \mathbf{1}(\psi)) && \text{because } \phi \wedge \psi = false \\ &= \sum_{x:A} \mathbf{1}(\phi) + \sum_{x:A} \mathbf{1}(\psi) && \text{by linearity of } \sum_{x:A} \\ &= E_{x:A}(\phi) + E_{x:A}(\psi) && \text{by def.} \end{aligned}$$

The result is obtained by dividing by $\mathsf{measure}(A)$. □

The property that probabilities are positive can be checked in a similar way. The assumption of unit measure ($P_{x:A}(true) = 1$) is a simple consequence of the definition.

2.1 Dealing with equality

In some situations it is useful to use equality of real-valued expressions (for example "john is as tall as mary"). Perhaps the most obvious way to encode equality between x and y is by using $\mathsf{IsTrue}(x = y)$. Assume that x and y are both taken in a space A of strictly positive measure, we can naively write the space B of equal x and y as follows.

$$B = \Sigma(x : A)\Sigma(y : A)\mathsf{IsTrue}(x = y)$$

Unfortunately, the above definition is problematic, because $x = y$ is stochastically impossible for real-valued x and y. [1] Consequently $\mathsf{measure}(B) = 0$. In turn, when evaluating probabilities involving B, one gets division by zero and the probabilities are undefined using the definitions given above.

2.1.1 A theoretical approach

What we would like is to replace $\mathsf{IsTrue}(x = y)$ by another space $x \equiv y$, such that the density of $x \equiv y$ is zero when $x \neq y$, but whose total measure is 1 (instead of 0). This can be done conceptually by increasing the density at the points where $x = y$. To do this, we must first introduce the $\mathsf{Factor}(e)$ space, which acts like $\mathsf{IsTrue}(\phi)$, but e gives directly the factor to be used

[1] Readers who are not familiar with this property can convince themselves informally by seeing that getting x and y to be equal requires an impossible alignment of infinite precision. Formally this can be seen by carrying out the computation of integrals as defined above.

in the integration (which can thus be greater than 1). Hence, its integrator is as follows:

$$\sum_{x:\mathsf{Factor}(e_1)} e_2 = [\![e_1]\!] \cdot [\![e_2[\Diamond/x]]\!].$$

Second, we need to pick a sufficiently great factor, so when integrating it over a 0-measure area, the result end up being 1. This can only be done with an infinitely large factor.

One may believe that no such space exists, but, fortunately, such a space has already been extensively studied, and it is known as the *Dirac δ function*. Classically, δ has a single parameter, and its density is 0 when this parameter is nonzero, and its defining property is:

$$\int_{-\infty}^{\infty} f(x)\delta(x)\,dx = f(0)$$

In terms of spaces, the same property becomes:

$$\sum_{x:\mathsf{Distr}(\delta)} t = t[0/x]$$

Hence we can define $x \equiv y$ to be for $\mathsf{Factor}(\delta(x-y))$.

We can now compute the measure of our motivating example B:

$$\begin{aligned}
\mathsf{measure}(\Sigma(x:A)\Sigma(y:A)x \equiv y) &= \sum_{x:A}\sum_{y:A}\sum_{p:\mathsf{Factor}(\delta(y-x))} 1 \\
&= \sum_{x:A}\sum_{z:\{y-x\,|\,x:A\}}\sum_{p:\mathsf{Factor}(\delta(z))} 1 \quad \text{by substitution} \\
&= \sum_{x:A}\sum_{z:\{y-x\,|\,x:A\}}\sum_{z:\mathsf{Distr}(\delta)} 1 \\
&= \sum_{x:A}\sum_{z:\{y-x\,|\,x:A\}} 1 \quad \text{by } \delta \text{ property} \\
&= \sum_{x:A}\sum_{y:A} 1 \\
&= \mathsf{measure}(A)^2
\end{aligned}$$

We see that involving $x \equiv y$ does not make the measure of spaces 0, and hence probabilites remain well-defined. Computing symbolic integration involving δ is not possible in every case, but we refer the reader to [12] for a generic method.

2.1.2 A numerical approach

Perhaps more disturbing that δ not being always computable, it does not lend itself well to Monte Carlo methods, which we describe in 2.3. We essentially are faced with the same problem as originially. If we sample random any x and y, and their numerical representations have a high resolution then, it will be extremely rare that $x = y$, and the Monte-Carlo approximation will not converge.

A possible solution is to increase the density in a non-zero region around the points such that $x = y$, in a smooth fashion. One way to do that is to take the density of the space $x \equiv y$ to be a Gaussian curve of a suitably small standard deviation σ^2 and which has its maximum at $x = y$:

$$\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{(x-y)}{\sigma})^2}$$

Like all probability density functions, the gaussian has density 1, and we thereby avoid spaces of zero measure.

While this approach is pleasing, choosing a suitable value for σ is not necessarily obvious. If it is too small, then we fall into the original pitfall: most of the time the density will be too small to contribute significantly to the integral. Conversely, if σ is too large, then one gets an excessively imprecise result. Unless otherwise stated, we have run our models with $\sigma = 1$.

2.2 Record notation

When dealing with complex structures involving nested Σ spaces, the expressions for projections become quickly inscrutable. For this reason we use the record notation for such spaces and the corresponding projections.

Definition 5. (Record spaces and projections) Formally, record spaces are defined by translation to Σ spaces, as follows:

$$[x_1 : A_1; \ldots; x_n : A_n] = \Sigma(x_1 : A_1)\Sigma(x_2 : A_2) \ldots A_n$$

Additionally if $e : [x_1 : A_1; \ldots; x_n : A_n]$, then $e.x_i$ is a shorthand for $\pi_1(\pi_2(\pi_2(\ldots e)))$ (the number of repetitions of π_2 is the index of the field in the record).

For similar reasons, we use a shorthand notation for the expected value over several variables, defined as follows:

$$E_{x_0:A_0,\ldots,x_n:A_n}(e) = E_{x_0:A_0}(\ldots(E_{x_n:A_n}(e)))$$

[2] In fact, if f_σ is a gaussian function with mean 0 and standard deviation σ, then $\delta(x) = \lim_{\sigma\to 0} f_\sigma(x)$

2.3 Approximation via sampling

Unfortunately, in the majority of cases the mathematical expressions produced by the semantics given above contain integrals which cannot be evaluated symbolically.

Hence, we are forced to resort to a numerical approximation algorithm to evaluate them. We use a variant of Gibbs sampling, which is itself an instance of a Markov Chain Monte Carlo (MCMC) method. The algorithm that we use closely follows the one described in [9].

All Monte Carlo methods are based on the same principle, which can be outlined as follows.To evaluate $P_{x:A}(\phi[x])$: 1. Sample a random x in A; 2. Check if $\phi[x]$ holds for a chosen value of x; 3. Repeat this process a large number of times.The ratio of the number successes to the number of tries converges to $P_{x:A}(\phi[x])$ as the number of tries tends to infinity.

In certain cases it is very hard to find any sample $x : A$. If (say) A contains an $\mathsf{IsTrue}(\psi)$ space where ψ is satisfied one time in a million, then it will be necessary to try a million samples until one try can be counted. In our application, these kind of situations will happen whenever 1. sets with many hypotheses are considered, 2. very strong hypotheses are tested. For example, "99.9 percent of men walk" requires such a precise arrangement of parameters that most samples will end up being discarded when this condition is checked.

To mitigate this problem, MCMC methods do not sample elements independently. Rather, each new sample x is based on a previous sample. Typically, only a single parameter is changed at every step. On average, the next sample is chosen to be as probable as the previous one, or more so. This way, the system is able to find many (probable) samples.

But samples can form (probably) disconnected regions in the chain space, and thus certain configurations may end up being explored more thoroughly than other, equally (or more) probable ones.

Ultimately, it is up to the designer of the underlying problem to avoid the pitfalls of the approximation methods. Because the phrasing of the hypotheses are infinitely variable for any natural language, we cannot avoid these pitfalls entirely. However, certain semantic designs will be more prone to problems than others.

3 Quantifiers

Even though it has very few constructions, LMS is sufficently expressive to encode the usual logical quantifiers: every x in A satisfies ϕ iff the subspace

of A where ϕ holds is (at least) as big as A itself.[3] The definition of the existential quantifier follows a similar pattern.

$$\forall x : A.\phi \stackrel{def}{=} \mathsf{measure}(A) \leq \mathsf{measure}(\Sigma(x : A).\mathsf{IsTrue}(\phi))$$
$$\exists x : A.\phi \stackrel{def}{=} 0 \leq \mathsf{measure}(\Sigma(x : A).\mathsf{IsTrue}(\phi))$$

4 Comparison with probabilistic programming languages

With LMS, we propose a way to describe types and an associated density (spaces). The tradition in the linguistic community is to use instead probabilistic programming languages [8, 5, 9]. Simply put, probabilistic programming languages do not describe spaces as such, but instead functions which generate elements of a certain type. Using LMS presents advantages. First, probabilistic programming languages typically do not natively offer the option to run an inference *within* another inference. In contrast this is done straightforwardly in LMS using the $\mathsf{measure}(e)$ expression. Second, the semantics of LMS is more straightforward than that of a formal probabilistic programming language: this is because LMS does not allow sampling within expressions. (Only spaces can refer to other spaces). We refer the reader to the work of [5] for an example of a probabilistic programming language equipped with formal semantics. Third, constructing spaces is very similar to constructing types and logical formulas. Thus we hope that LMS can readily be used by linguists who are used to interpret natural language into type theories (or similar logical systems).

5 Conclusion

In sum, LMS aims to solve a language design problem. It aims to bridge a semantic gap between abstract syntax for natural languages and the evaluation of probabilistic truth values.

On the one hand, this language is sufficiently powerful to express probabilistic problems, is convenient enough to support probabilistic syllogisms. On the other hand, it has a simple model in terms of integrators, and, for linguistic purposes it compares favourably with usual probabilistic programming languages.

[3] This definition is problematic when ϕ logically false for some x, but still stochastically true. To deal with such cases, one must then use disintegrators, as explained by [12].

References

1. Barendregt, H. P. Lambda calculi with types. *Handbook of logic in computer science 2* (1992), 117–309.

2. Bernardy, J.-P., Blanck, R., Chatzikyriakidis, S., and Lappin, S. A compositional Bayesian semantics for natural language. In *Proceedings of the International Workshop on Language, Cognition and Computational Models, COLING 2018, Santa Fe, New Mexico* (2018), pp. 1–11.

3. Bernardy, J.-P., Blanck, R., Chatzikyriakidis, S., Lappin, S., and Maskharashvili, A. Bayesian inference semantics: A modelling system and a test suite. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM), Minneapolis* (2019), Association for Computational Linguistics, pp. 263–272.

4. Bernardy, J.-P., Blanck, R., Chatzikyriakidis, S., Lappin, S., and Maskharashvili, A. Predicates as boxes in bayesian semantics for natural language. In *Proceedings of the 22nd Nordic Conference on Computational Linguistics* (2019), ACL.

5. Borgström, J., Gordon, A. D., Greenberg, M., Margetson, J., and Van Gael, J. Measure transformer semantics for Bayesian machine learning. *Logical Methods in Computer Science 9* (2013), 1–39.

6. Fox, C., and Lappin, S. *Foundations of Intensional Semantics*. Blackwell, 2005.

7. Goodman, N., and Lassiter, D. Probabilistic semantics and pragmatics: Uncertainty in language and thought. In *The Handbook of Contemporary Semantic Theory, Second Edition*, S. Lappin and C. Fox, Eds. Wiley-Blackwell, Malden, Oxford, 2015, pp. 655–686.

8. Goodman, N., Mansinghka, V. K., Roy, D., Bonawitz, K., and Tenenbaum, J. Church: a language for generative models. In *Proceedings of the 24th Conference Uncertainty in Artificial Intelligence (UAI)*. 2008, pp. 220–229.

9. Goodman, N., and Stuhlmüller, A. The Design and Implementation of Probabilistic Programming Languages. `http://dippl.org`, 2014. Accessed: 2018-4-17.

10. Lassiter, D., and Goodman, N. Adjectival vagueness in a Bayesian model of interpretation. *Synthese 194* (2017), 3801–3836.

11. Ranta, A. Grammatical framework. *Journal of Functional Programming 14*, 2 (2004), 145–189.

12. Shan, C.-c., and Ramsey, N. Exact bayesian inference by symbolic disintegration. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages* (2017), POPL, pp. 130–144.

THE PARALLEL MEANING BANK: A FRAMEWORK FOR SEMANTICALLY ANNOTATING MULTIPLE LANGUAGES

Lasha Abzianidze Rik van Noord Chunliu Wang Johan Bos

CLCG, University of Groningen, the Netherlands
{l.abzianidze, r.i.k.van.noord, chunliu.wang, johan.bos}@rug.nl

Abstract

This paper gives a general description of the ideas behind the Parallel Meaning Bank, a framework with the aim to provide an easy way to annotate compositional semantics for texts written in languages other than English. The annotation procedure is semi-automatic, and comprises seven layers of linguistic information: segmentation, symbolisation, semantic tagging, word sense disambiguation, syntactic structure, thematic role labelling, and co-reference. New languages can be added to the meaning bank as long as the documents are based on translations from English, but also introduce new interesting challenges on the linguistics assumptions underlying the Parallel Meaning Bank.

Keywords and phrases: parallel corpus, semantic annotation, meaning banking, compositional semantics, formal semantics

1 Introduction

The Parallel Meaning Bank (PMB) is a semantically annotated parallel corpus for English, Dutch, German, Italian, Chinese, and Japanese. The key idea behind the PMB is based on the assumption that translations—at least to a large extent—preserve the meaning between the source and target language. Making use of translated texts, annotation for one language can be re-used for the translations, resulting in an economical annotation platform. One of the core ideas is that the human annotations can help improve existing language technology (based on supervised machine learning) in the areas of machine translation, automatic question answering and advanced information retrieval.

The PMB can be viewed as a multilingual version of the Groningen Meaning Bank, GMB [7, 14], an annotation platform designed for the meaning of English texts. Like the GMB, the PMB contains the raw

EN Alfred Nobel invented dynamite in 1866.

DE Alfred Nobel erfand 1866 das Dynamit.

IT Alfred Nobel inventò la dinamite nel 1866.

NL Alfred Nobel vond in 1866 het dynamiet uit.

x_1 x_2 e_1 t_1
male.n.02(x_1)
Name(x_1, alfred_nobel)
invent.v.01(e_1)
Time(e_1, t_1)
Result(e_1, x_2)
Agent(e_1, x_1)
time.n.08(t_1)
YearOfCentury(t_1, 1866)
$t_1 \prec$ now
dynamite.n.01(x_2)

Figure 1: 03/0766 PMB document has four meaning-preserving translations. As a result, each translation is annotated with the same meaning representation.

texts and various layers of linguistic annotation, ultimately resulting in a formal meaning representation based on Discourse Representation Theory (DRT) [25]. The annotations are automatically generated by a pipeline of state-of-the-art natural language processing (NLP) tools and then manually corrected by annotators. Semantic annotation is hard, even for trained linguists. To give an idea what a meaning representation in the PMB looks like, consider Figure 1. These representations are called Discourse Representation Structures (DRSs) in DRT.

This paper gives a general overview of the PMB and describes several aspects of it in more details. First, we describe the seven annotation layers that are used to automatically obtain formal meaning representations (Section 2 and Section 3). Then, we sketch how the semantic annotation can be projected from one language to another (Section 4). This is followed by an overview of applications of the released PMB data (Section 5). Finally, we show how new documents in new languages are added to the PMB and how language technology tools are bootstrapped for new languages (Section 6).

2 The Seven Annotation Layers

There are two main approaches on semantic annotation. The first approach is to go directly from the source text to the target meaning representations, without any layer of analysis in between. An example of this method is the corpus constructed for Abstract Meaning Representations [5]. The second approach, adopted in the PMB, is to view annotation as a sequence of layers of analysis, where each layer builds on the previous layer by adding a piece of (semantic) information to it. In the PMB, seven layers of annotation are distinguished:

1. Tokenisation: detecting sentence boundaries and word tokens;
2. Symbolisation: assigning a non-logical symbol to a word (or multi-word) token. This layer unifies lemmatization and normalization.
3. Word sense disambiguation: assigning concepts to symbols, based on the WordNet [23] sense inventory;
4. Co-reference resolution: marking antecedents for anaphoric expressions;
5. Thematic role labelling: annotate relations between entities using VerbNet roles [11] and comparison operators (e.g., temporal and spatial orders);
6. Syntactic analysis: providing lexical categories for each token and building a syntactic structure for the sentence, based on Combinatory Categorial Grammar [36];
7. Semantic tagging: assigning a semantic type to a word token [3].

These annotation layers are demonstrated in Figure 2. The annotation layers provide all information needed to provide a compositional semantic analysis for a sentence (for additional details about the PMB annotation layers see [2]). This is done by using the lambda calculus, and adopting Discourse Representation Theory as semantic formalism, implemented by the semantic parser Boxer [13]. In a final step, the semantic analysis of single sentences are combined into one meaning representation covering the entire text.

3 Annotation Pipeline

Manually creating the seven annotation layers for a large amount of documents is not a feasible task. For this reason, we use an annotation pipeline to automatically segment raw documents, label tokens with token-based annotations, and produce the final meaning representation. The pipeline consists of a sequence of NLP tools each serving for a specific annotation layer. The pipeline of English-specific tools is highlighted with a green background in Figure 3. Below, we briefly describe each NLP tool:[1]

- Elephant [21] is used for tokenisation. The tool performs sentence boundary and word token detection as a single labelling task: each

[1]Currently, the pipeline lacks specialized NLP tools for word sense disambiguation and co-reference resolution. Therefore, these layers are manually annotated for now.

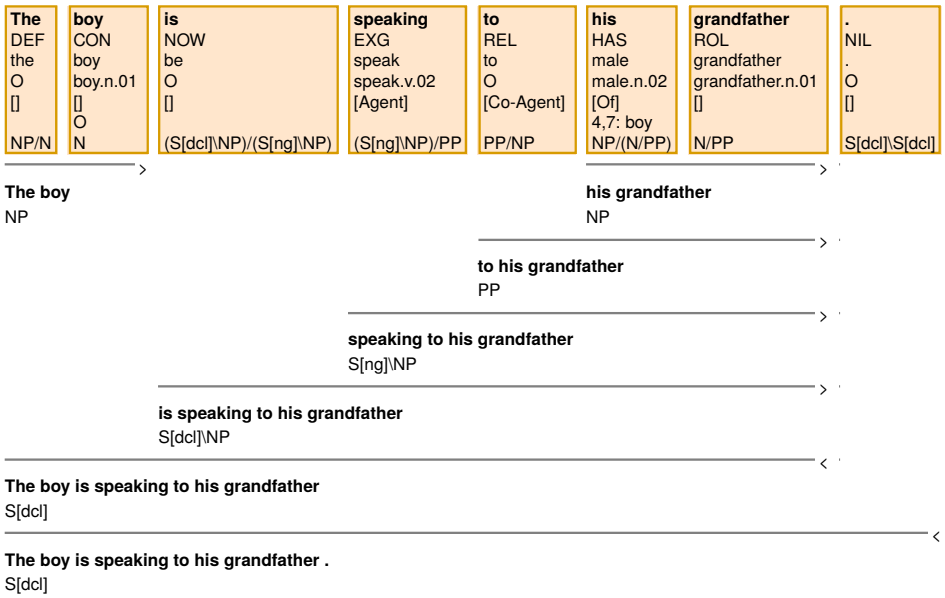


Figure 2: All the seven annotation layers of the English translation of 46/2924 PMB document. The order of layers starting from top: tokenisation, semantic tagging, symbolisation, word sense disambiguation, thematic role labelling, co-reference resolution, and syntactic analysis.

character is labelled with one of the four labels depending on being sentence beginning, token beginning, inside token, and outside token;

- Semantic tagging is carried using the tri-gram based TnT tagger [15];
- The lemmatisation part of symbolisation is done with the help of the lemmatizer Morpha [33]. Currently, we use instance-based learning for the normalisation part. In particular, for every existing combination of lemma and semantic tag in the PMB, the most frequent symbol is memorized which is later reused to tag a token with the corresponding pair of semantic tag and lemma. For example, to get a symbol for a token *eight*, first its lemma `eight` and semantic tags `QUC` is obtained and then the instance-based learning will assign `8` as a symbol to it
- Obtaining syntactic analysis consists of assigning lexical categories to tokens and constructing a derivation tree over these categories. The both subtasks are performed using EasyCCG [28], a CCG-based parser that requires only tokenised input and pre-trained word embeddings.

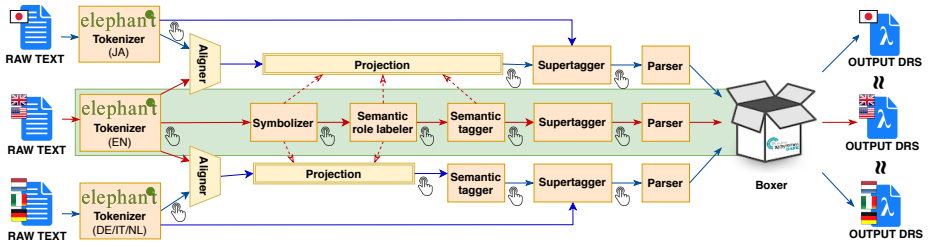


Figure 3: The PMB pipeline: a sequence of NLP tools that processes raw texts and outputs formal meaning representations. The hand icon indicates functionality of overwriting parts of the system outputs with manual annotations.

- Thematic role labelling is done with a tagger based on Conditional Random Fields [26]. The tagger employs semantic tags, symbols and CCG lexical categories as features to predict thematic roles.

The output of each tool can be manually corrected by human annotators.[2] In this way, we use a human-in-the-loop approach to obtain gold standard annotation layers and the final meaning representations. We also apply bootstrapping with the gold standard annotation layers to retrain and further improve the quality of the NLP tools. This aims at reducing human annotation efforts while still retaining high quality system outputs.

4 Annotation Projection

The previous two sections gave a rough overview of what is required to provide a compositional analysis for the meaning of a text for one language. For historical reasons, this language is English, because of the tools developed earlier in the Groningen Meaning Bank [6]. Instead of starting from scratch and implementing a pipeline for other languages, we follow a different approach in the PMB. This approach is called *annotation projection*, and requires that the English text has an adequate translation in the language of your choice. The first languages that we added in the PMB were languages close to English, such as other Germanic languages (Dutch and German) and Italian, a Romance language.

[2]The PMB documents can be manually annotated with the PMB explorer, an online annotation environment, available at: https://pmb.let.rug.nl/explorer. Anybody can register and annotate the documents.

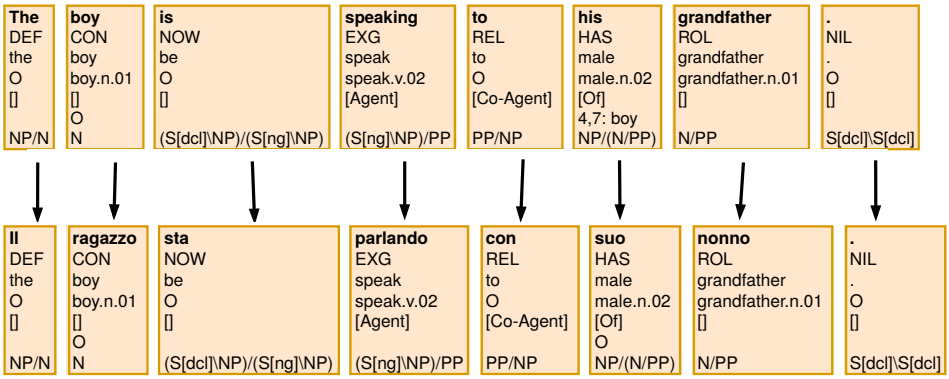


Figure 4: An example of a complete annotation projection: all the seven annotation layers are projected from English to Italian.

The idea of semantic projection is extremely simple, but its implementation is surprisingly challenging even for closely-related languages. The assumption that a translation doesn't change much of the meaning, is the driving force in this approach. But for reasons of scalability, we are not just interested in the final meaning representation, but also in the compositional analysis supporting this final meaning representation. This makes projection more challenging.

In the PMB, annotation projection is implemented using word alignment between English and the target language.[3] The alignments provide clues how to transfer the layers of annotation from English to the other languages [19]. For cases where the syntactic structure of the target language is similar to that of the source language (English), this is often straightforward. Figure 4 shows one of such cases where a literal translation leads to a perfect word alignment and therefore to a complete annotation projection. This leads to the very same meaning representation for the Italian translation that the English translation had.

But translations are not always in a perfect word-to-word and order-preserving correspondence as in the previous example. Even closely-related language show different behaviour with respect to a word order, multi-word expressions, definiteness, use of articles, and noun-noun compounds. So automatic projection requires the help of human annotators to provide corrections.

In the PMB, we go beyond the mere annotation projection as it is brittle for wide-coverage translations. To do so, using the same NLP tools, we (re-)train semantic tagging and syntactic parsing models for non-English

[3] We employ GIZA++ [34] to automatically induce word alignments.

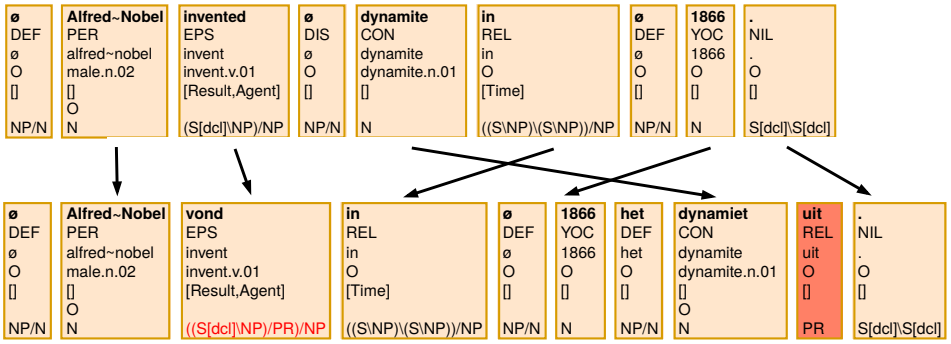


Figure 5: An imperfect annotation projection is compensated by the language-specific syntactic parsing model.

languages. Initially the training data consisted of translations with perfect annotation projections. Gradually the training data increased as a result of reprocessing the rest of the translations with new models and correcting manually where necessary. For example, the annotation projection in Figure 5 fails for the syntactic analysis layer due to the difference in a word order of the Dutch translation. But with the help of the in-house trained Dutch model of the parser, it is possible to automatically recover a correct syntactic analysis of the Dutch translation, which eventually leads to the same meaning representation (see Figure 1).[4]

Figure 3 shows the PMB pipeline of NLP tools that simultaneously processes documents in five languages. While currently only symbols and thematic roles are projected for the Dutch, German, and Italian translations, the Japanese translations also get semantic tags projected from the English translations. In the near future, we plan to retrain Japanese-specific model for the semantic tagging.

Currently we are investigating what consequences semantic annotation projection has on languages that behave significantly different from English from a linguistic perspective. Here we think of languages such as Chinese and Japanese, and perhaps also Kartvelian languages such as Georgian [35]. These languages add pressure on the principles of the PMB, in particular on the extent one can adopt a single framework for each layer in the semantic analysis pipeline. To give a first example, the semantic tags might be subject to extension of the tagset for new languages that show phenomena that cannot be captured with the existing semantic categories. To give a second example, we assume CCG as the theory of syntatic structure suitable for all languages. CCG starts with a base of atomic categories, which work

[4]To verify whether projected annotations yield the same meaning representation as of English, we perform fine-grained matching of meaning representations [40].

well for Germanic languages, but other languages could be hard to adopt in the parameters provided for English. In future work we need to take a closer look at such a wider perspective. As a final example, in Chinese, there are less syntactic constraints for verbs, but there is widespread use of pro-drop, and a larger distribution of ambiguous constructions, such as the relative clause and verbal coordination. In addition, the inherent ambiguities caused by both verbal coordination and relative clauses of Chinese make semantic parsing more difficult than syntactic parsing [48].

5 Applications

The PMB annotations are released periodically, free of charge.[5] It includes gold standard data, which is fully manually corrected, as well as silver (partially manually corrected) and bronze (with no manual corrections) data. The releases so far contain documents for English, German, Italian and Dutch, but for future releases we plan to include Chinese and Japanese. An overview of the releases is shown in Table 1.

Table 1: Number of released documents per language for the five current PMB releases.

Release	Quality	EN	DE	IT	NL
PMB-1.0.0	**Gold**	2,049	641	387	394
PMB-2.0.0	**Gold**	3,925	1,048	568	527
	Silver	66,693	611	266	192
PMB-2.1.0	**Gold**	4,555	1,175	635	586
	Silver	71,308	688	306	207
PMB-2.2.0	**Gold**	5,929	1,419	724	633
	Silver	67,965	4,235	2,515	1,051
	Bronze	120,622	102,998	61,504	20,554
PMB-3.0.0	**Gold**	8,403	1,979	1,062	1,012
	Silver	97.598	5,250	2,772	1,301
	Bronze	146,371	121,111	64,305	21,550

One of the goals of the PMB releases is to aid DRS parsing, a task in which a model has to automatically produce a DRS from raw text. These produced DRSs can then potentially be of benefit in other language related tasks, such as machine translation or question answering. Early approaches used rule-based system for only small fragments of English [24,43], though

[5] https://pmb.let.rug.nl/data.php

wide-coverage semantic parsers that use supervised machine learning were also developed, mainly on the GMB data [12, 27, 13, 30, 31].

The main advantage of the PMB is that it contains gold standard data for evaluating the parsers. This is in contrast to the GMB, which contains partially manually corrected evaluation sets that are not guaranteed to be gold standard. This allowed for the organization of a shared task on English DRS parsing in PMB format [4]. Five systems participated in this shared task, which all used neural networks in some capacity. Three systems used sequence-to-sequence models based on the first PMB-based DRS parser [41], which was extended by including linguistic features [42, 39] and by swapping the bi-LSTM encoder/decoder for a transformer model [29], which was the winning system. The two other systems consisted of a transition-based parser that relied on stack-LSTMs [20] and a neural graph parsing system that converted the DRSs to a more general graph format before parsing [22]. The latter is also the first system that produced results for German, Italian and Dutch DRS parsing.

There are also other applications of the PMB data. For one, semantic tagging can be useful as either an auxiliary task to improve a main task [10, 9, 1], or as a general dataset for evaluating neural architectures [8, 32, 16, 18]. Moreover, PMB data has been used in research on natural language inference [45] and machine translation [17].

6 A Look at the Future: Extending the PMB

The PMB can be extended in terms of introducing new documents or new translations. Translations may belong to languages that are new or already covered in the PMB. In case a translation belongs to a new language, its integration in the PMB requires more work as the new language needs to be processed by the PMB pipeline. In this section, we describe the procedure and conditions for extending the PMB.

The simplest extension procedure is when adding translations to PMB documents in one of the PMB (non-English) languages, let's say $L_{\texttt{PMB}}$. In this case, no new documents are created, and there is no need to develop new NLP tools as the PMB pipeline can already process texts in $L_{\texttt{PMB}}$. If the PMB uses the projection method for $L_{\texttt{PMB}}$, then it is necessary to align the new $L_{\texttt{PMB}}$-translations to the existing English translations. For the best results, the alignment is usually done on all PMB English-$L_{\texttt{PMB}}$ bitexts. This might affect the alignments of old PMB documents and annotations of the projected layers, consequently. Since the alignment is carried out on more bitexts than before, the assumption is that the quality of alignments improves. Whether the change influences alignments negatively, this can

be verified for the translations already having a gold standard annotation for the projected layers. The difference for the projected layers will show up as conflicts with the gold standard.

Adding a new parallel corpus to the PMB involves adding completely new documents. Taking the architecture of the PMB into account, one of the languages of the new corpus must be English. Let's first consider the scenario when all the languages of the corpus are covered by the PMB. All new documents (consisting of translations) get new `part/doc` identifiers and are uniformly distributed over all the 100 parts of the PMB. If the newly added documents belong to a text genre new to the PMB, some NLP tools in the pipeline might require further adaptation. For example, if the documents belong to the social media domain, one might need to correct the tokenization or semantic tagging of slang words and retrain the corresponding tools on the corrected annotations. Additionally, the procedures of inducing new alignments and verifying the changes caused by them are also applicable in this scenario.

The case where newly added parallel corpus contains translations not belonging to the PMB languages is the most laborious. New languages require their own annotation pipelines. Here, we describe our first experiences from adding Japanese [46] and Chinese, using translations from Tatoeba.[6]

To enable the annotation projection from English to Japanese, it is necessary to extract word alignments from the bitext, which itself presupposes tokenisation of the Japanese translations. Since we strive to use the same NLP tools with language-specific models for each annotation layer, we trained a Japanese model of the Elephant tokenizer.[7] After extracting the word alignments, token-based annotations were projected for one-to-one word alignments. Since English and Japanese are languages with radically different typologies, the annotation projection for the syntactic analysis failed for almost all Japanese translations. As syntactic analyses play a key role for obtaining meaning representations in the PMB because they contribute to defining lexical semantics and guiding composition of phrasal semantics, a quick integration required a Japanese CCG parser in the PMB pipeline. Fortunately, there exists a Japanese CCG parser, depCCG [47]. We trained a new Japanese model for EasyCCG on the output of depCCG. We opted for training a new model to keep the PMB pipeline lean rather than integrating an additional tool in it. In the near future, we plan to train a Japanese model for the semantic tagging in order to eliminate "holes" in the semantic tagging layer caused by the annotation projection.

We are currently adding (Mandarin) Chinese translations for the PMB

[6] https://tatoeba.org

[7] The training data was obtained by processing the Japanese translations in the PMB with the UDPipe 1.2.0 [37] and the model `japanese-gsd-ud-2.3-181115`.

documents. While doing so, we are taking a route similar to the one we took for Japanese. To train the Chinese model for Elephant, we used the output from jieba.[8] The EasyCCG model was trained on the CCG derivation trees which were obtained from the Chinese Treebank [44] following [38].

The current undertakings of adding more languages to the framework doesn't mean that all problems are solved. The entire PMB enterprise emits a formal flavour of universality of language analysis. This is reflected in the practical use of our language technology pipeline, with the aim of using the same NLP tools but employing the language-specific models as the only variable element. We have reached a high level of generalization, but there are also many refinements that seek improvement, in particular on the ontological, categorial, and contextual level. The only way to make progress in this area of computational semantics is by considering other languages and getting your hands dirty!

References

1. Abdou, M., Kulmizev, A., Ravishankar, V., Abzianidze, L., and Bos, J. What can we learn from semantic tagging? In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (2018), pp. 4881–4889.

2. Abzianidze, L., Bjerva, J., Evang, K., Haagsma, H., van Noord, R., Ludmann, P., Nguyen, D.-D., and Bos, J. The Parallel Meaning Bank: Towards a multilingual corpus of translations annotated with compositional meaning representations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers* (Valencia, Spain, April 2017), Association for Computational Linguistics, pp. 242–247.

3. Abzianidze, L., and Bos, J. Towards universal semantic tagging. In *Proceedings of the 12th International Conference on Computational Semantics (IWCS 2017) – Short Papers* (Montpellier, France, September 2017), Association for Computational Linguistics.

4. Abzianidze, L., van Noord, R., Haagsma, H., and Bos, J. The first shared task on discourse representation structure parsing. In *Proceedings of the IWCS Shared Task on Semantic Parsing* (Gothenburg, Sweden, May 2019), Association for Computational Linguistics.

[8] https://github.com/fxsjy/jieba

5. Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Koehn, P., Palmer, M., and Schneider, N. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse* (Sofia, Bulgaria, 2013), pp. 178–186.

6. Basile, V., Bos, J., Evang, K., and Venhuizen, N. Developing a large semantically annotated corpus. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC 2012)* (Istanbul, Turkey, 2012), pp. 3196–3200.

7. Basile, V., Bos, J., Evang, K., and Venhuizen, N. A platform for collaborative semantic annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2012)* (Avignon, France, 2012), pp. 92–96.

8. Belinkov, Y., Màrquez, L., Sajjad, H., Durrani, N., Dalvi, F., and Glass, J. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (2017), pp. 1–10.

9. Bjerva, J. Will my auxiliary tagging task help? estimating auxiliary tasks effectivity in multi-task learning. In *Proceedings of the 21st Nordic Conference on Computational Linguistics* (2017), pp. 216–220.

10. Bjerva, J., Plank, B., and Bos, J. Semantic tagging with deep residual networks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers* (Osaka, Japan, 2016), pp. 3531–3541.

11. Bonial, C., Corvey, W. J., Palmer, M., Petukhova, V., and Bunt, H. A hierarchical unification of LIRICS and VerbNet semantic roles. In *Proceedings of the 5th IEEE International Conference on Semantic Computing (ICSC 2011)* (2011), pp. 483–489.

12. Bos, J. Wide-Coverage Semantic Analysis with Boxer. In *Semantics in Text Processing. STEP 2008 Conference Proceedings*, J. Bos and R. Delmonte, Eds., vol. 1 of *Research in Computational Semantics*. College Publications, 2008, pp. 277–286.

13. Bos, J. Open-domain semantic parsing with Boxer. In *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015)* (2015), B. Megyesi, Ed., pp. 301–304.

14. Bos, J., Basile, V., Evang, K., Venhuizen, N., and Bjerva, J. The Groningen Meaning Bank. In *Handbook of Linguistic Annotation*, N. Ide and J. Pustejovsky, Eds. Springer Netherlands, 2017.

15. Brants, T. Tnt: A statistical part-of-speech tagger. In *Proceedings of the Sixth Conference on Applied Natural Language Processing* (Stroudsburg, PA, USA, 2000), ANLC '00, Association for Computational Linguistics, pp. 224–231.

16. Dalvi, F., Sajjad, H., Durrani, N., and Belinkov, Y. Exploiting redundancy in pre-trained language models for efficient transfer learning. *arXiv preprint arXiv:2004.04010* (2020).

17. Durrani, N., Dalvi, F., Sajjad, H., Belinkov, Y., and Nakov, P. One size does not fit all: Comparing NMT representations of different granularities. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (Minneapolis, Minnesota, June 2019), Association for Computational Linguistics, pp. 1504–1516.

18. Ek, A., Bernardy, J.-P., and Lappin, S. Language modeling with syntactic and semantic representation for sentence acceptability predictions. In *NEAL Proceedings of the 22nd Nordic Conference on Computional Linguistics (NoDaLiDa), September 30-October 2, Turku, Finland* (2019), no. 167, Linköping University Electronic Press, pp. 76–85.

19. Evang, K. *Cross-lingual Semantic Parsing with Categorial Grammars.* PhD thesis, University of Groningen, 2016.

20. Evang, K. Transition-based DRS parsing using stack-LSTMs. In *Proceedings of the IWCS 2019 Shared Task on Semantic Parsing* (2019).

21. Evang, K., Basile, V., Chrupała, G., and Bos, J. Elephant: Sequence labeling for word and sentence segmentation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (Seattle, Washington, USA, 2013), pp. 1422–1426.

22. Fancellu, F., Gilroy, S., Lopez, A., and Lapata, M. Semantic graph parsing with recurrent neural network DAG grammars. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (Hong Kong, China, Nov. 2019), Association for Computational Linguistics, pp. 2769–2778.

23. Fellbaum, C., Ed. *WordNet. An Electronic Lexical Database*. The MIT Press, Cambridge, Ma., USA, 1998.

24. Johnson, M., and Klein, E. Discourse, anaphora and parsing. In *11th International Conference on Computational Linguistics. Proceedings of Coling '86* (Bonn, Germany, 1986), pp. 669–675.

25. Kamp, H., and Reyle, U. *From Discourse to Logic; An Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and DRT*. Kluwer, Dordrecht, 1993.

26. Lafferty, J. D., McCallum, A., and Pereira, F. C. N. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning* (San Francisco, CA, USA, 2001), ICML '01, Morgan Kaufmann Publishers Inc., p. 282–289.

27. Le, P., and Zuidema, W. Learning compositional semantics for open domain semantic parsing. In *Proceedings of COLING 2012* (Mumbai, India, 2012), The COLING 2012 Organizing Committee, pp. 1535–1552.

28. Lewis, M., and Steedman, M. A* CCG parsing with a supertag-factored model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (Doha, Qatar, 2014), pp. 990–1000.

29. Liu, J., Cohen, S., and Lapata, M. Discourse representation structure parsing with recurrent neural networks and the transformer model. In *Proceedings of the IWCS 2019 Shared Task on Semantic Parsing* (2019).

30. Liu, J., Cohen, S. B., and Lapata, M. Discourse representation structure parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (Melbourne, Australia, 2018), vol. 1, pp. 429–439.

31. Liu, J., Cohen, S. B., and Lapata, M. Discourse representation parsing for sentences and documents. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (Florence, Italy, July 2019), Association for Computational Linguistics, pp. 6248–6262.

32. Liu, N. F., Gardner, M., Belinkov, Y., Peters, M. E., and Smith, N. A. Linguistic knowledge and transferability of contextual

representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (Minneapolis, Minnesota, June 2019), Association for Computational Linguistics, pp. 1073–1094.

33. Minnen, G., Carroll, J., and Pearce, D. Applied morphological processing of English. *Natural Language Engineering 7*, 3 (2001), 207–223.

34. Och, F. J., and Ney, H. A systematic comparison of various statistical alignment models. *Computational Linguistics 29*, 1 (2003), 19–51.

35. Pkhakadze, K., Chikvinidze, M., Chichua, G., Maskharashvili, A., and Beriashvili, I. An overview of the trial version of the georgian self-developing intellectual corpus necessary for creating georgian text analyzer, speech processing, and automatic translation systems. In *Reports of Enlarged Session of the Seminar of I. Vekua Institute of Applied Mathematics* (2014), vol. 28, I. Vekua Institute of Applied Mathematics.

36. Steedman, M. *The Syntactic Process.* The MIT Press, Cambridge, Ma., USA, 2001.

37. Straka, M., Hajič, J., and Straková, J. UDPipe: Trainable pipeline for processing CoNLL-u files performing tokenization, morphological analysis, POS tagging and parsing. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)* (Portorož, Slovenia, May 2016), European Language Resources Association (ELRA), pp. 4290–4297.

38. Tse, D., and Curran, J. R. Chinese CCGbank: extracting CCG derivations from the Penn Chinese treebank. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)* (Beijing, China, Aug. 2010), Coling 2010 Organizing Committee, pp. 1083–1091.

39. van Noord, R. Neural Boxer at the IWCS shared task on DRS parsing. In *Proceedings of the IWCS 2019 Shared Task on Semantic Parsing* (2019).

40. van Noord, R., Abzianidze, L., Haagsma, H., and Bos, J. Evaluating scoped meaning representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)* (Miyazaki, Japan, 2018).

41. van Noord, R., Abzianidze, L., Toral, A., and Bos, J. Exploring neural methods for parsing discourse representation structures. *Transactions of the Association for Computational Linguistics 6* (2018), 619–633.

42. van Noord, R., Toral, A., and Bos, J. Linguistic information in neural semantic parsing with multiple encoders. In *IWCS 2019-13th International Conference on Computational Semantics-Short papers* (2019).

43. Wada, H., and Asher, N. BUILDRS: An implementation of DR theory and LFG. In *11th International Conference on Computational Linguistics. Proceedings of Coling '86* (Bonn, Germany, 1986), pp. 540–545.

44. Xue, N., Zhang, X., Jiang, Z., Palmer, M., Xia, F., Chiou, F.-D., and Chang, M. Chinese treebank 9.0 ldc2016t13, 2016.

45. Yanaka, H., Mineshima, K., Bekki, D., Inui, K., Sekine, S., Abzianidze, L., and Bos, J. Can neural networks understand monotonicity reasoning? In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP* (2019), pp. 31–40.

46. Yanaka, H., Mineshima, K., Yamada, A., Yamaguchi, Y., Kubota, Y., Abzianidze, L., and Bos, J. Building a japanese version of parallel meaning bank. In *26th Annual Meeting of the Association for Natural Language Processing* (2020), pp. 1145–1158.

47. Yoshikawa, M., Noji, H., and Matsumoto, Y. A* CCG parsing with a supertag and dependency factored model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (Vancouver, Canada, July 2017), Association for Computational Linguistics, pp. 277–287.

48. Yu, K., Miyao, Y., Matsuzaki, T., Wang, X., and Tsujii, J. Analysis of the difficulties in Chinese deep parsing. In *Proceedings of the 12th International Conference on Parsing Technologies* (Dublin, Ireland, Oct. 2011), Association for Computational Linguistics, pp. 48–57.

A SHORT PROOF OF THE DECIDABILITY OF NORMALIZATION IN RECURSIVE PROGRAM SCHEMES

Zurab Khasidashvili

Intel Corporation, Haifa, Israel
zurab.khasidashvili@intel.com

Dedicated to work of Shalva Pkhakadze on the centenary of his birth

Abstract

We give a short and simple proof of the decidability of normalization in Recursive Program Schemes (RPSs). As a side result, we obtain an algorithm that effectively transforms any RPS into an irreducible one, in which shortest normalizing reductions are easy to construct.

1 Introduction

It is shown in [6] that normalization is decidable in *Recursive Program Schemes (RPSs)*. The proof there is quite complex as it employs the concept of *essential chains* of rules: An essential chain of rules is a sequence of rules $r_1, r_2, \ldots$ such that an r_{i+1}-redex has an *essential occurrence* in the right-hand side of r_i, for all $i = 1, 2, \ldots$. Here a subterm (in particular, a redex) is called *essential* if it has a descendant under any reduction of the term (where the concept of *descendant* is a refinement of that of residual; it allows to trace subterms along reductions).[1] It is shown in [6] that a term t in an RPS R is normalizable iff all essential chains of the rules corresponding to essential redexes in t are finite. Showing the decidability of normalization thus required showing the decidability of essentiality in RPSs.

In this paper we give a much shorter and simpler (thus less informative) proof. We assume that the reader is familiar with the basic concepts of Term Rewriting: All needed information can be found in [9, 4, 13]. We use t and s to denote terms, u to denote redexes, and r to denote rules.

[1] For the reader familiar with the concept of *neededness of redexes* [4], we remark that essentiality is a refinement of neededness in that it makes sense for all subterms, and not redexes only.

We write $t \to s$ or $t \overset{u}{\to} s$ if s is obtained from t by reducing a redex u, and $\twoheadrightarrow$ denotes the transitive reflexive closure of $\to$; we write $\overset{\emptyset}{\twoheadrightarrow}$ when the number of steps is 0.

2 The proof

We start by introducing RPSs [2, 9].[2] RPSs have been studied in [11, 12] under the name of *contracting symbols of type I*.

Definition 1. *An RPS R is a Term Rewriting System (TRS) [9, 13] whose alphabet consists of a finite set $\mathcal{F}$ of* unknown *function symbols, a finite set $\mathcal{G}$ of* basic *function symbols, and variables. The rules of R have the form*

$$r : f(x_1, \ldots, x_n) \to s,$$

where f is an unknown function symbol in $\mathcal{F}$, x_i are pairwise distinct variables, and s is an arbitrary term built from function symbols (basic or unknown) and variables. There is exactly one rule in R for every unknown function symbol in $\mathcal{F}$.

A rule r as above is called *irreducible* if s is an R-normal form, and is called *reducible* otherwise. We call an RPS *irreducible* if every rule in it with normalizable right-hand side is irreducible. That is, an irreducible RPS may contain rules whose right-hand sides are not in normal form, but these right-hand sides are *not* normalizable. Clearly, if a rule $r \in R$ is irreducible, for any term t in R, the normal form of t w.r.t. $\{r\}$ can be computed in (at most) as many steps as the number of r-redexes in t. This is why irreducible rules are attractive.

Lemma 1. If all rules of an RPS R are reducible, then no reducible term t in R has a normal form.

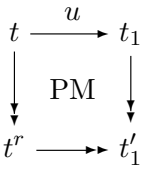
Proof. Suppose on the contrary that t has a normalizing reduction $t \twoheadrightarrow t' \overset{u}{\to} t^*$. Then the right-hand side of the rule for the redex u must be a normal-form – a contradiction. □

Lemma 2. Let R be an RPS containing an irreducible rule r. Further, for any term s in R, let s^r denote its $\{r\}$-normal form. And finally, let R^r be the RPS obtained from R by $\{r\}$-normalizing the right-hand sides of rules in R (i.e., by replacing all rules $t_1 \to t_2$ in R with $t_1 \to t_2^r$, respectively), and then by removing r. Then a term t in R is normalizable in R iff t^r is normalizable in R^r.

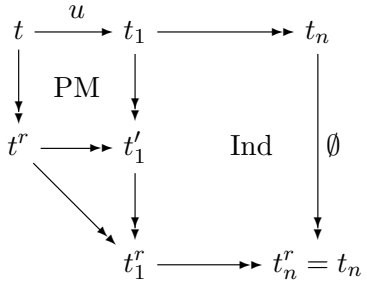
[2] RPSs are called *Recursive Applicative Program Schemes* in [2].

Proof. ($\Leftarrow$) Any step in R^r can be decomposed into an $R\backslash\{r\}$-step followed by a number of r-steps (contracting all created r-redexes). Hence, t^r is R-normalizable (since it is R^r-normalizable), and thus so is t (since $t \twoheadrightarrow t^r$ in R).

($\Rightarrow$) By induction on the length of a shortest R-normalizing reduction $t \overset{u}{\to} t_1 \to \ldots \to t_n$ starting from t. Using the Parallel Moves Lemma (PM) [9, 4], we can construct the following diagram in R, where $t^r \twoheadrightarrow t'_1$ is an R-reduction that contracts all residuals of u in t^r, which are disjoint, if any.



Let $t'_1 \twoheadrightarrow t^r_1$ be a reduction that contracts all r-redexes created by contracting the disjoint residuals of u along $t^r \twoheadrightarrow t'_1$ (in fact, there are no other r-redexes in t'_1). By the decomposition property of R^r-steps mentioned above, $t^r \twoheadrightarrow t^r_1$ in R^r if u is not an r-redex, and $t^r = t^r_1$ otherwise. By the induction assumption, t^r_1 is normalizable in R^r, hence so is t^r.



□

Theorem 1. Normalization is decidable in any RPS R.

Proof. By induction on the number of rules in R. By Lemma 1, we can assume that R contains an irreducible rule, r. By Lemma 2, a term t has a normal form in R iff t^r has a normal form in R^r, and we conclude (since R^r has fewer rules than R). □

3 Concluding remarks

Decidability of normalization in RPSs can also be derived from an advanced result of Nagaya and Toyama [10, page 264], stating that for a left-linear growing TRS R and a regular tree language L, the set of ground terms s

such that $s \twoheadrightarrow_R t$ for some $t \in L$ is regular. Here R is growing if for any rule $r \in R$ and any variable x that occurs both in left- and right-hand sides of r, x occurs in the left-hand side of r at depth 1 (i.e., just below the root symbol). RPSs are clearly growing.

Unlike the proof in [10], our proof gives an algorithm for transformation of RPSs into simpler and more efficient ones: Given an RPS R and a term t in R, we can construct (using Lemma 2) an irreducible RPS R' such that any term in R is normalizable in R iff it is normalizable in R', and the normal forms coincide. Note that $R' = R'_{irr} \cup R'_{red}$, where all rules in R'_{irr} are irreducible, and the right-hand sides of rules in R'_{red} are not normalizable. (Clearly, it does not make sense to compute R'_{red}-redexes, since such redexes do not have normal forms.) Then, if t contains an R'_{red}-redex that is not in an erased argument of an R'_{irr}-redex, then t has no normal form in R or R'. Otherwise, we normalize t w.r.t. R'_{irr} (e.g., using the innermost essential strategy, which is optimal in orthogonal TRSs in general [6]; a subterm of t is essential w.r.t. R_{irr} iff it is not in an erased argument of an R'_{irr}-redex in t). The obtained R_{irr}-normal form is also the normal form of t in R. (Cf. [1], where family-reductions are designed to achieve optimal evaluation of RPSs.)

We note that the proof presented in this work is based on the fact that the only redexes that can be created by reducing a redex are present explicitly already in the right-hand side of the applied rewrite rule. We therefore expect that the proof can be generalized to *Higher Order Recursive Program Schemes* [8] and *Persistent TRSs* [7] and *ERSs* [5, 8, 3]. Higher Order RPSs correspond to *contracting symbols of types IV and IV′* studied in [11, 12].

References

1. Berry G., Lévy J.-J. Minimal and optimal computations of recursive programs. *JACM 26*, 1979, p. 148-175.

2. Courcelle B. Recursive Applicative Program Schemes. In: J. van Leeuwen, ed. *Handbook of Theoretical Computer Science*, Chapter 9, vol. B, 1990, p. 459-492.

3. Glauert J., Kesner D., Khasidashvili Z. Expression Reduction Systems and Extensions: An Overview. *Processes, Terms and Cycles: Steps on the Road to Infinity, Essays Dedicated to Jan Willem Klop on the Occasion of His 60th Birthday*, A. Middeldorp, V. van Oostrom, F. van Raamsdonk, R. de Vrijer, eds. Springer LNCS 3838, 2005, p. 496-553.

4. Huet G. and Lévy J.-J. Computations in orthogonal rewriting systems. *Computational Logic, Essays in Honor of Alan Robinson*, J. - L. Lassez and G. Plotkin, eds. MIT Press, 1991, p. 394-443.

5. Khasidashvili Z. Expression reduction systems. *Proceedings of I. Vekua Institute of Applied Mathematics of Tbilisi State University*, v. 36, 1990, p. 200-220.

6. Khasidashvili Z. Optimal normalization in orthogonal term rewriting systems. *Proceedings of the* 5^{th} *International Conference on Rewriting Techniques and Applications*, RTA'93, C. Kirchner, ed., Springer LNCS, vol. 690, 1993, p. 243-258.

7. Khasidashvili Z. On the equivalence of persistent term rewriting systems and recursive program schemes *Proceedings of the* 2^{nd} *Israel Symposium on Theory of Computing and Systems*, ISTCS'93, IEEE Computer Society Press, 1993, p. 240-249.

8. Khasidashvili Z. On higher order recursive program schemes. *Proceedings of the* 19^{th} *International Colloquium on Trees in Algebra and Programming*, CAAP'94, S. Tison, ed., Springer LNCS, vol. 787, 1994, p. 172-186.

9. Klop J.W. Term rewriting systems. *Handbook of Logic in Computer Science*, vol. 2, S. Abramsky, D. Gabbay, and T. Maibaum eds., Oxford University Press, 1992, p. 1-116.

10. Nagaya T. and Toyama Y. Decidability for left-linear growing term rewriting systems. *Proceedings of the* 10^{th} *International Conference on Rewriting Techniques and Applications*, RTA'99, P. Narendran and M. Rusinowich, eds., Springer LNCS, vol. 1631, 1999, p. 256-270.

11. Pkhakadze Sh. Some problems of the notation theory. (In Russian) *Tbilisi University Press*, Tbilisi, 1977.

12. Pkhakadze Sh. A N. Bourbaki type general theory and the properties of contracting symbols and corresponding contracted forms. *Georgian Mathematical Journal*, Vol. 6, No. 2, 1999, 179-190.

13. Terese. Term Rewriting Systems. *Cambridge Tracts in Theoretical Computer Science*, Volume 55, 2003.

UNIFICATION MODULO α-EQUIVALENCE IN A MATHEMATICAL ASSISTANT SYSTEM

Temur Kutsia

RISC, Johannes Kepler University, Linz, Austria
kutsia@risc.jku.at

Abstract

We study unification modulo α-equivalence in a language that combines permissive nominal terms and sequence unknowns. Such unification problems originate from reasoning tasks in the mathematical assistant system Theorema. We propose an algorithm that combines a version of permissive nominal unification with length-bounded sequence unification. It is terminating, sound, minimal, and satisfies a restricted version of completeness. We also consider two special cases when the boundedness restriction can be lifted: (1) matching fragment and (2) the fragment where sequence unknowns appear in the last argument positions in subterms. They permit minimal and complete algorithms. All three algorithms are implemented and included in the unification package of the Theorema system.

Keywords and phrases: Permissive nominal unification, α-equivalence, mathematical assistant systems, Theorema, sequence variables.

AMS subject classification (2010): 03B70, 68Q42, 68W30, 68T15.

1 Introduction

Unification is a procedure for symbolic equation solving, used as the main computational mechanism in many automated deduction methods. Given two logical expressions, unification algorithms try to find instantiations of variables to make the expressions identical (syntactic unification) or equal modulo an equational theory (equational unification). Unification is a key ingredient in theorem provers, proof assistants, and declarative programming systems.

In this paper we consider a particular variant: unification modulo α-equivalence. Two logical expressions are α-equivalent, if they are the same modulo renaming of bound variables. The algorithm described here corresponds to what is implemented in the mathematical assistant system Theorema [10].

The specific features of Theorema influenced the design of the algorithm. Theorema provides a pretty liberal higher-order syntax. Its expression may

be a constant, a variable, an application of an expression to a sequence of expressions, or a quantified expression. Variables are of two kinds: for individual expressions (individual variables, here called term variables) and for sequences of expressions (sequence variables). Arities of function constants, in general, are not fixed.

As an example of α-unification, consider an equation between two statements about sets from [10]: $X + 1 \in \{x \mid_x x > X\} \approx^?_\alpha Y \in \{y \mid_y y > a\}$, where X and Y are individual variables to be instantiated, and x and y are variables bound by the set quantifier. The algorithm computes the unifier $\sigma = \{Y \mapsto a + 1, X \mapsto a\}$, which maps Y to $a + 1$ and X to a. Applying σ to the given problem, we get $a + 1 \in \{x \mid_x x > a\}$ in the left and $a+1 \in \{y \mid_y y > a\}$ in the right, which are not identical, but equal modulo renaming the bound variable y into x.

Sequence variables are very handy in knowledge representation and rule-based programming. They play an important role both in Theorema and in the programming language this system is implemented in: the Wolfram language of the symbolic computation system Mathematica [65]. However, the expressive power of sequence variables makes unification with them pretty hard. There are problems which may have infinitely many independent unifiers even for the syntactic case. For instance, the equation $f(\overline{x}, a) =^? f(a, \overline{x})$ has infinitely many solutions, mapping $\overline{x}$ to finite (including empty) sequences of a's: $\{\overline{x} \mapsto ()\}, \{\overline{x} \mapsto (a)\}, \{\overline{x} \mapsto (a, a)\}, \ldots$.

As it was pointed out in [10], despite the fact that Theorema provides higher-order syntax, there is no hidden default higher-order logic behind it. In the process of developing an α-unification algorithm for this language, we chose a pragmatic, minimalistic approach, since α-equivalence is the fundamental property of languages with binders. The idea was to provide the basic algorithm that deals with language constructs such as quantifiers/binders, applicative expressions, and sequence variables. In specific reasoners, the algorithm either can be used as provided, or it may be extended/modified to meet the needs of that particular reasoner. For instance, for a special prover for higher-order logic, one may wish to extend the algorithm to deal with equalities modulo β and η rules, while, e.g., for first-order reasoning, the provided algorithm would suffice.

To illustrate the mentioned features of this approach, we recall examples from [10]: For instance, in our language, the equation $X(a) \approx^?_\alpha f(a, a)$ does not have a solution, because unification is not done modulo β (in contrast to four unifiers when the β-rule is permitted). Note also that $f(a)(a)$ and $f(a, a)$ are not seen as equal. The problem $X(a) \approx^?_\alpha f(a)(a)$ can be solved by $\{X \mapsto f(a)\}$.

To distinguish between the variables that are bound in (the context of) an expression, and the variables that are free and can be instantiated by

unification, we call the former *atoms* (as in nominal unification [61]) and keep the word 'variable' only for the latter.

An important feature in the algorithm described in this paper is the use of so called permission sets, like in permissive nominal unification [17]. The permission set of a variable explicitly indicates which atoms may appear in the instantiation of that variable during unification. For instance, $x^{\{a,b\}}$ means that in the instantiations of x, only a and b are permitted from the atoms: $\{a, b\}$ is the permission set for x. Hence, $x^{\{a,b\}}$ may be unified, e.g., with the terms $f(a, g(a))$ or $f(y^{\{b\}}, a)$, but not with $f(c)$, where c is an atom, because c does not belong to the permission set.

We call pairs consisting of a variable and a permission set *unknowns*. In Theorema, they arise in the context of proving. For instance, an attempt to prove $\forall x.\exists y.\ f(x) = y$ gives a unification problem $f(a) =^? y^{\{a\}}$, where a is an atom (an arbitrary but fixed constant obtained after removing the universal quantifier) and $y^{\{a\}}$ is the unknown, whose instantiation is to be computed. The atom a is permitted in the instantiation. The unification problem can be solved by the substitution $\{y^{\{a\}} \mapsto f(a)\}$, which leads to the proof of the statement. On the other hand, proving $\exists y.\forall x.\ f(x) = y$ fails, because it gives the unification problem $f(a) =^? y^{\emptyset}$, which does not have a solution: $f(a)$ is not permitted in $y^{\emptyset}$, since the empty permission set forbids the atom a to appear in the instantiation.

In this work, we describe an algorithm Unif-Alg for solving such unification problems. They may contain unknowns for terms and sequences, atoms, variadic function symbols, applications, and binders. To guarantee the termination of the algorithm, the length of instantiations of sequence unknowns is limited. Termination, soundness, and restricted completeness of the algorithm are shown. The set of unifiers it computes is minimal. We also identify two fragments, for which termination and completeness can be obtained without limiting the length of sequence unknown instantiations: matching fragment (equations where one side is unknown-free) and the fragment, where sequence unknowns occupy the last argument positions in subterms they occur.

The plan of the paper is following: after a brief overview of related work, we introduce the language, define terms, substitutions, unification problems and related notions in Section 2. In Section 3, we describe the unification algorithm Unif-Alg and two other algorithms for special fragments: Match-Alg and Unif-Alg-Last. Properties of these three algorithms are investigated in Section 4, where theorems about termination, soundness, completeness and restricted completeness are proved.

The algorithms are implemented in Mathematica and are a part of the Theorema system.

Related work

α-Unification

Unification modulo α-equivalence has been studied in [61] in the context of nominal terms. Nominal techniques, introduced in [24, 25], extend first-order syntax by names and bindings, where binders quantify names in their arguments. The syntax still remains first-order. Functional abstraction λ, logic quantifiers $\forall, \exists$, integral $\int$ are some well-known examples of binders.[1] The motivation for introducing nominal techniques was to formally represent and study systems with binding. These techniques syntactically distinguish between atoms (object level variables), which can be bound, and unknowns (meta-variables), which can be substituted. Substitutions may cause atom capture by binders. Renaming of atoms is made explicit by their name swapping (which avoids capture). Informal 'fresh variable conditions' is made a part of the language under freshness constraints.

Nominal unification has good algorithmic properties: it is decidable, unitary, and can be solved in polynomial time. Unification, matching, and related problems in nominal setting are quite actively investigated nowadays. Various kinds of equation solving methods between nominal terms, and their relations to similar problems have been studied by several authors, see, e.g., [2, 3, 6, 11–13, 22, 23, 46, 47, 58]. Permissive nominal unification, introduced in [17], differs from nominal unification in that it changes the idea of 'specifying which atoms are forbidden in instantiations' into 'specifying which atoms are permitted in instantiations'. It has several advantages, outlined in [17], including the possibility to always choose a fresh atom and the substitution-only based notion of unifier. A nice survey on permissive-nominal logic can be found in [26].

Permission sets, in general, may be infinite, but in the context of their application in proof-search, finite ones suffice [27]. This applies to our case as well, because our unification problems originate from tasks in Theorema reasoners. Note that our unification problems avoid binding atoms from permissive sets. Hence, substitutions do not cause atom capture. Also, two distinct unknowns do not share the same variable. Another difference from [17] is the way how atoms are renamed. In nominal techniques, this is done by permutations, which has many advantages [27]. However, we stick to a more familiar way of atom replacements such as $[a := b]$ and rely on the capabilities of the meta-language to generate fresh names.

[1] See [55] for rules about introducing new binders in the language.

Unification with sequence variables

Sequence variables come hand-in-hand with variadic symbols (i.e., those without the fixed arity). Such symbols are pretty common. They can be, e.g., names in Common Logic [33] and KIF [28], XML tags, symbols originated from different knowledge bases after their integration, functions and constructors implemented in symbolic computation systems (e.g., Mathematica), arithmetic operations written in variadic form, flexary symbols in OpenMath [32], etc. Unification with sequence variables is infinitary (the minimal complete set of unifiers for some problems can be infinite).

Incomplete unification algorithms, motivated by applications, have been proposed in [29, 56]. A complete procedure was introduced in [39, 41]. Decidability was proved in [39, 43] and various terminating fragments have been studied in [43, 45]. Matching with sequence variables modulo equational theories and its relation with the built-in pattern matching mechanism of Mathematica was investigated in [21]. Among various applications of equation solving with sequence variables one can mention knowledge representation [50], rule-based and constraint (logic) programming [**?**,**?**,19,59], rewriting [20], theorem proving [40], XML processing [14, 15, 44], etc. The main variant we consider in this paper corresponds to bounded-length sequence unification. The idea of imposing such a length bound has been used earlier for dealing with sequence equations and constraints in constraint programming solver [59], program synthesis [56], ontology reasoning [54], etc. To the best of our knowledge, sequence unification and permissive nominal unification have not been combined before.

Theorema

The Theorema project has been initiated by Bruno Buchberger in the mid-1990s [8]. The goal was to develop a software system that aids all the phases of mathematical theory exploration. It includes invention of mathematical concepts; formulation and proof of propositions; formulation of problems; formulation, verification, and execution of algorithms for solving problems; maintenance of knowledge bases developed and verified in this process, and retrieval of mathematical knowledge. Many proof assistant systems and dedicated tools support various aspects of theory exploration, see, e.g., [1,5,7,16,30,31,34–36,49,51–53,60]. Theorema has been used, for instance, for the development and implementation of a new method for solving linear boundary value problems [57], for the automated synthesis of Buchberger's algorithm for computing Gröbner bases [9], for formalizing pillage games in theoretical economics [37], for theory exploration in reduction rings [48], for synthesis of sorting algorithms for binary trees [18], just to name a few.

The object language of Theorema is a version of a higher-order language with sequence variables. Its meta-language for implementing reasoners (provers, solvers, simplifiers) is Mathematica. Theorema provides a modern GUI [64] and an infrastructure for developing special reasoners and for combining them into more general tools. Examples of special reasoners implemented in Theorema are provers for first-order predicate logic [38], set theory [63], equational logic [42], elementary analysis [62], a package for Green's algebra [57], etc. They all rely in a way or another on the unification package of Theorema. This package has been modified and improved several times since its initial implementation in the first version of the system at the end of 1990s. The algorithms we describe in this paper are a part of the new unification package in Theorema 2.0 [10].

2 Preliminaries

We consider an alphabet A consisting of the following pairwise disjoint countable sets of symbols:

- $\mathcal{V}_{\mathrm{T}}$: the set of term variables,
- $\mathcal{V}_{\mathrm{S}}$: the set of sequence variables,
- $\mathcal{A}$: the set of atoms,
- $\mathcal{F}$: the set of variadic function symbols,
- $\mathcal{Q}$: the set of quantifiers.

Definition 1 (Terms, s-terms)**.** A *term* t and an *s-term* s over the alphabet A are defined by the grammar:

$$t ::= x^P \mid a \mid f \mid t(s_1, \ldots, s_n) \mid Qa.t$$
$$s ::= t \mid \overline{x}^P$$

where $x \in \mathcal{V}_{\mathrm{T}}$, $P \subset \mathcal{A}$, $a \in \mathcal{A}$, $f \in \mathcal{F}$, $Q \in \mathcal{Q}$, $\overline{x} \in \mathcal{V}_{\mathrm{S}}$, and $n \geq 0$. The set P is assumed to be finite. It is called a *permission set*.

The expressions x^P and $\overline{x}^P$ are called term and sequence unknown, respectively. Note that variadic function symbols may apply to arbitrary number of arguments, the terms f and $f()$ are not assumed to be the same, and quantifiers operate on atoms. Note also a restricted use of sequence unknowns. Namely, a sequence unknown can be neither the head of a term nor the body of a quantifier, i.e., expressions such as $\overline{x}^P(a, b)$ and $Qa.\overline{x}^P$ are not terms.

We write sequences of s-terms in parentheses for readability. Below the following meta-variables are used:

- for term variables: x, y, z,
- for sequence variables $\overline{x}, \overline{y}, \overline{z}$,
- for term or sequence variables: v, w,
- for atoms: a, b, c, d,
- for function symbols: f, g, h,
- for quantifiers: Q,
- for terms: t, u,
- for s-terms: s, r,
- for finite sequences of terms: $\tilde{t}, \tilde{u}$,
- for finite sequences of s-terms: $\tilde{s}, \tilde{r}$.

Example 1. Let $a, b, c \in \mathcal{A}$, $f, g, h \in \mathcal{F}$, $\lambda \in \mathcal{Q}$. Then the following expressions are terms: f, $f()$, $f(a, f, x^{\emptyset})$, $f(a, b)(c, \overline{x}^{\{a,c\}})$, $\lambda a.f(a, x^{\{a\}})$, $(\lambda a.f(a))(g)$, $(\lambda a.f(a)(b))(g(a, \overline{x}^{(\{a,b\})})$.

The *head* of a term t, denoted by $\mathsf{head}(t)$, is defined as $\mathsf{head}(x^P) = x^P$, $\mathsf{head}(a) = a$, $\mathsf{head}(f) = f$, $\mathsf{head}(t(\tilde{s})) = t$, and $\mathsf{head}(Qa.t) = Q$.

Definition 2 (Free, bound atoms)**.** The sets of *free and bound atoms* of an s-term s, denoted respectively by $\mathsf{fa}(s)$ and $\mathsf{ba}(s)$, are defined as follows:

$$\mathsf{fa}(x^P) = P, \quad \mathsf{fa}(\overline{x}^P) = P, \quad \mathsf{fa}(f) = \emptyset, \quad \mathsf{fa}(a) = \{a\},$$
$$\mathsf{fa}(t(s_1, \ldots, s_n)) = \mathsf{fa}(t) \cup \cup_{i=1}^{n} \mathsf{fa}(s_i),$$
$$\mathsf{fa}(Qa.t) = \mathsf{fa}(t) \setminus \{a\}.$$

$$\mathsf{ba}(x^P) = \mathsf{ba}(\overline{x}^P) = \mathsf{ba}(f) = \mathsf{ba}(a) = \emptyset,$$
$$\mathsf{ba}(t(s_1, \ldots, s_n)) = \mathsf{ba}(t) \cup \cup_{i=1}^{n} \mathsf{ba}(s_i),$$
$$\mathsf{ba}(Qa.t) = \mathsf{ba}(t) \cup \{a\}.$$

Further:

$$\mathsf{fa}((s_1, \ldots, s_n)) = \mathsf{fa}(s_1) \cup \cdots \cup \mathsf{fa}(s_n).$$
$$\mathsf{ba}((s_1, \ldots, s_n)) = \mathsf{ba}(s_1) \cup \cdots \cup \mathsf{ba}(s_n).$$

$$\mathsf{atoms}(s) = \mathsf{fa}(s) \cup \mathsf{ba}(s). \qquad \mathsf{atoms}(\tilde{s}) = \mathsf{fa}(\tilde{s}) \cup \mathsf{ba}(\tilde{s}).$$

The set of all unknowns of s (resp. of $\tilde{s}$) is denoted by $\mathsf{unkn}(s)$ (resp. $\mathsf{unkn}(\tilde{s})$).

Definition 3 (Substitution)**.** A *substitution* σ is a mapping, which maps unknowns to terms and to sequences of s-terms and is defined as follows:

- for each x^P, $\sigma(x^P)$ is a term,
- for each $\overline{x}^P$, $\sigma(\overline{x}^P)$ is a finite sequence of s-terms

such that

- $\mathsf{fa}(\sigma(v^P)) \subseteq P$ for all $v \in \mathcal{V}_\mathrm{T} \cup \mathcal{V}_\mathrm{S}$,
- $\sigma(x^P) = x^P$ for all but finitely many term unknowns,
- $\sigma(\overline{x}^P) = (\overline{x}^P)$ for all but finitely many sequence unknowns,
- if $\sigma(v^P) \neq v^P$ for some $v \in \mathcal{V}_\mathrm{T} \cup \mathcal{V}_\mathrm{S}$ and P, then $\sigma(v^R) = v^R$ for the same v and all $B \neq R$.

Usually, substitutions are written as finite sets of mapping pairs. For instance, $\{x^{\{a,b\}} \mapsto Qa.f(a)(\overline{y}^{\{b\}}),\ y^{\{a\}} \mapsto g(a,f),\ \overline{x}^{\{a,b\}} \mapsto (f(a),\ Qb.b,\ b),\ \overline{y}^{\{b\}} \mapsto ()\}$ is a substitution, which maps $x^{\{a,b\}}$ to $Qa.f(a)(\overline{y}^{\{b\}})$, $y^{\{a\}}$ to $g(a,f)$, $\overline{x}^{\{a,b\}}$ to the sequence $(f(a),\ Qb.b,\ b)$, and $\overline{y}^{\{b\}}$ to the empty sequence $()$. The other term unknowns are mapped to themselves, and the other sequence unknowns are mapped to themselves as singleton sequences.

We use lower case Greek letters for substitutions. The only exception is the identity substitution, denoted by *Id*. The *domain* and *range* of a substitution σ are defined as

$$dom(\sigma) := \{x^P \mid \sigma(x^P) \neq x^P\} \cup \{\overline{x}^P \mid \sigma(\overline{x}^P) \neq (\overline{x}^P)\}$$
$$ran(\sigma) := \{\sigma(v^P) \mid v^P \in dom(\sigma)\}.$$

Given a set of unknowns S, the *restriction* of a substitution σ on S, denoted by $\sigma|_S$, is a substitution for which $\sigma|_S(v^P) = \sigma(v^P)$ if $v^P \in S$, and $\sigma|_S(v^P) = v^P$ otherwise.

Substitutions can be composed in the usual way, see, e.g., [4]. We write $\sigma\vartheta$ for the composition of substitutions σ and ϑ (the order matters).

A substitution σ is *idempotent*, if $\sigma\sigma = \sigma$. The defining property of idempotent substitutions is $dom(\sigma) \cap \mathsf{unkn}(ran(\sigma)) = \emptyset$.

Definition 4 (Replacement)**.** An *atom replacement* or, shortly, *replacement* is a mapping from an atom to an atom, written as $[a := b]$.

Replacements and substitutions can be applied to terms according to the following definitions:

Definition 5 (Replacement application)**.** Application of a replacement $[a := b]$ to an s-term s, denoted $s[a := b]$, is defined as follows:

$$v^P[a := b] = v^{P[a:=b]}, \quad \text{where}$$
$$\text{if } a \in P, \text{ then } P[a := b] = (P \setminus \{a\}) \cup \{b\}, \text{ else } P[a := b] = P.$$

$$f[a := b] = f, \quad a[a := b] = b, \quad c[a := b] = c \text{ if } c \neq a,$$
$$t(s_1, \ldots, s_n)[a := b] = t[a := b](s_1[a := b], \ldots, s_n[a := b]),$$
$$(Qc.t)[a := b] = Qc[a := b].t[a := b].$$

Note that replacement application allows atom capture: $Qa.f(a, b)[b := a] = Qa.f(a, a)$.

Definition 6 (Substitution application)**.** Application of a substitution σ to an s-term s, denoted $s\sigma$, is defined as follows:

$$v^P\sigma = \sigma(v^P), \quad a\sigma = a, \quad f\sigma = f,$$
$$t(s_1, \ldots, s_n)\sigma = t\sigma(s_1\sigma, \ldots, s_n\sigma),$$
$$(Qa.t)\sigma = Qb.t[a := b]\sigma, \text{ where } b \notin \mathsf{atoms}(t).$$

Substitution application avoids atom capture, unlike replacements. For instance, we have $Qa.f(a, x^{\{a\}})\{x^{\{a\}} \mapsto a\} = Qb.f(b, a)$.

Definition 7 (The relation $\approx_\alpha$)**.** The relation $\approx_\alpha$ on s-terms is the smallest relation that satisfies the following:

$$v^P \approx_\alpha v^P, \quad a \approx_\alpha a, \quad f \approx_\alpha f,$$
$$t^1(s^1_1, \ldots s^1_n) \approx_\alpha t^2(s^2_1, \ldots s^2_n) \text{ if } t^1 \approx_\alpha t^2 \text{ and } s^1_i \approx_\alpha s^2_i \text{ for } i = \overline{1, n},$$
$$Qa.t \approx_\alpha Qb.u,$$
$$\text{if } t[a := c] \approx_\alpha u[b := c] \text{ where } c \notin \mathsf{atoms}(t) \cup \mathsf{atoms}(u).$$

It can be proved that $\approx_\alpha$ is a congruence relation. It is called the α-equivalence. Essentially, two s-terms are α-equivalent if they are equal modulo bound atom renaming.

Definition 8 (Instantiation quasi-ordering)**.** An s-term s is *more general* than r (r is an *instance* of s), written $s \precsim r$, if there exists a substitution σ such that $s\sigma \approx_\alpha r$.

A substitution σ is more general than ϑ, written $\sigma \precsim \vartheta$, if there exists a substitution φ such that $x^P\sigma\varphi \approx_\alpha x^P\vartheta$ for any x^P.

The relation $\precsim$ is quasi-ordering (a reflexive and transitive binary relation). It is called *instantiation quasi-ordering*. It induces an equivalence relation (both on terms and on substitutions), denoted by $\sim$.

Definition 9 (Unification problem, unifier)**.** A *unification problem* Γ is a finite set of unification equations (term pairs):

$$\Gamma = \{t_1 \approx^?_\alpha u_1, \ldots, t_n \approx^?_\alpha u_n\}.$$

Γ does not contain two different unknowns with the same variable: If v^{P_1} and v^{P_2} occur in Γ, then $P_1 = P_2$. For each v^P occurring in Γ, the atoms in P are free in the equation where v^P occurs.

A substitution σ is a *unifier* of Γ if $t_i\sigma \approx_\alpha u_i\sigma$ for all $1 \leq i \leq n$. It is called a most general unifier, if $\sigma \precsim \vartheta$ for any unifier ϑ of Γ.

It is known [41, 43] that when unification problems contains sequence variables, there might be infinitely many unifiers, which are not comparable with each other by $\precsim$. (In other words, the problem is infinitary.) In such cases, one talks about minimal complete sets of unifiers:

Definition 10 (Minimal complete set of unifiers)**.** Let Γ be a unification problem and S be a set of substitutions. Then S is called a *complete set of unifiers* of Γ, if the following two properties are satisfied:

Soundness: Every $\sigma \in S$ is a unifier of Γ.

Completeness: For each unifier ϑ of Γ, there exists $\sigma \in \Gamma$ such that $\sigma \precsim \vartheta$.

S is a *minimal complete set of unifiers* of Γ, if, in addition, the minimality property holds:

Minimality: If there exist $\sigma_1, \sigma_2 \in S$ such that $\sigma_1 \precsim \sigma_2$, then $\sigma_1 = \sigma_2$.

We denote S in this case by $\mathsf{mcsu}(\Gamma)$.

A simple example of a unification problem with infinite minimal complete set of unifiers is $\Gamma = \{f(\overline{x}^{\{a\}}, a) \approx_\alpha f(a, \overline{x}^{\{a\}})\}$. We have $\mathsf{mcsu}(\Gamma) = \{\{\overline{x}^{\{a\}} \mapsto ()\}, \{\overline{x}^{\{a\}} \mapsto (a)\}, \{\overline{x}^{\{a\}} \mapsto (a, a)\}, \{\overline{x} \mapsto (a, a, a)\}, \ldots\}$.

Example 2. Here we show some unification problems Γ and their minimal complete sets of unifiers. $(\forall, \lambda \in \mathcal{Q},\ p, >, \langle\cdot\rangle \in \mathcal{F})$:

$$\Gamma = \{\forall a.\forall b.\langle a > b, p(a, b)\rangle \approx^?_\alpha \forall b.\forall a.\langle b > a, p(b, a)\rangle\},$$
$$\mathsf{mcsu}(\Gamma) = \{Id\}.$$

$$\Gamma = \{\forall a.p(a, x^{\{b\}}) \approx^?_\alpha \forall b.p(b, b)\},$$
$$\mathsf{mcsu}(\Gamma) = \emptyset : \text{not unifiable.}$$

$$\Gamma = \{\forall a.p(a, x^{\{b\}}) \approx^?_\alpha \forall c.p(c, b)\},$$
$$\mathsf{mcsu}(\Gamma) = \{\{x^{\{b\}} \mapsto b\}\}.$$

$$\Gamma = \{\forall a.p(a, x^{\{b\}}) \approx^?_\alpha \forall b.p(b, c)\},$$
$$\mathsf{mcsu}(\Gamma) = \emptyset : \text{not unifiable.}$$

$\Gamma = \{\forall a.p(a, x^{\emptyset}) \approx^{?}_{\alpha} \forall a.p(a, \lambda b.b)\}$,
$\mathsf{mcsu}(\Gamma) = \{\{x^{\emptyset} \mapsto \lambda b.b\}\}$.

$\Gamma = \{\forall a.p(a, x^{\emptyset}) \approx^{?}_{\alpha} \forall a.p(a, b)\}$,
$\mathsf{mcsu}(\Gamma) = \emptyset$: not unifiable.

$\Gamma = \{\forall a.p(a, x^{\{b\}}) \approx^{?}_{\alpha} \forall c.p(c, f(b, g))\}$,
$\mathsf{mcsu}(\Gamma) = \{\{x^{\{b\}} \mapsto f(b, g)\}\}$.

$\Gamma = \{\forall a.p(a, \overline{x}^{\{b\}}, \overline{y}^{\{b,c,d\}}) \approx^{?}_{\alpha} \forall a.p(a, b, f(b), c)\}$,
$\mathsf{mcsu}(\Gamma) = \{\{\overline{x}^{\{b\}} \mapsto (),\ \overline{y}^{\{b,c,d\}} \mapsto (b, f(b), c)\},$
$\{\overline{x}^{\{b\}} \mapsto (b),\ \overline{y}^{\{b,c,d\}} \mapsto (f(b), c)\},$
$\{\overline{x}^{\{b\}} \mapsto (b, f(b)),\ \overline{y}^{\{b,c,d\}} \mapsto (c)\}\}$.

$\Gamma = \{p(\overline{x}^{\{a,b\}}) \approx^{?}_{\alpha} p(\overline{y}^{\{a\}}, \overline{z}^{\{a,b,c\}})\}$,
$\mathsf{mcsu}(\Gamma) = \{\{\overline{x}^{\{a,b\}} \mapsto (\overline{y}^{\{a\}}, \overline{z}'^{\{a,b\}}), \overline{z}^{\{a,b,c\}} \mapsto \overline{z}'^{\{a,b\}}\}\}$.

$\Gamma = \{\forall a.p(a, x^{\{c,c'\}}) \approx^{?}_{\alpha} \forall b.p(b, f(y^{\{c,c''\}}))\}$,
$\mathsf{mcsu}(\Gamma) = \{\{x^{\{c,c'\}} \mapsto f(y'^{\{c\}}), y^{\{c,c''\}} \mapsto y'^{\{c\}}\}\}$.

$\Gamma = \{x^{\{a,b\}}(y^{\{a,c\}}) \approx^{?}_{\alpha} f(y^{\{a,c\}})(g(z^{\{c\}}, a))\}$,
$\mathsf{mcsu}(\Gamma) = \{\{x^{\{a,b\}} \mapsto f(g(z'^{\emptyset}, a)), y^{\{a,c\}} \mapsto g(z'^{\emptyset}, a), z^{\{c\}} \mapsto z^{\emptyset}\}\}$.

$\Gamma = \{p(\overline{x}^{\{a,b\}}, \overline{y}^{\{a,c\}}) \approx^{?}_{\alpha} p(f(\overline{y}^{\{a,c\}}), g(z^{\{b\}}, a))\}$,
$\mathsf{mcsu}(\Gamma) = \{\{\overline{x}^{\{a,b\}} \mapsto (f(g(z'^{\emptyset}, a))), \overline{y}^{\{a,c\}} \mapsto (g(z'^{\emptyset}, a)), z^{\{b\}} \mapsto z^{\emptyset}\},$
$\{\overline{x}^{\{a,b\}} \mapsto (f(), g(z^{\{b\}}, a)),\ \overline{y}^{\{a,c\}} \mapsto ()\}\}$.

3 The algorithm

In this section we formulate our unification algorithm in a rule-based way. Rules operate on states, which is a pair $\Gamma; \sigma$, where Γ is a unification problem and σ is a substitution. Intuitively, a state shows the problem "still to be solved" and the unifier "computed so far".

In the rules we use renaming substitutions, defined as follows: A substitution σ is called a *renaming substitution* if it injectively maps term unknowns to term unknowns and sequence unknowns to sequence unknowns.

The rules are the following (the symbol symb is use as a metavariable for a function symbol, atom, or a term unknown):

T: **Trivial**

$$\{t \approx^?_\alpha t\} \uplus \Gamma;\ \sigma \rightsquigarrow \Gamma;\ \sigma.$$

HD: **Head Decomposition**

$$\{t(\tilde{s}) \approx^?_\alpha u(\tilde{r})\} \uplus \Gamma;\ \sigma \rightsquigarrow \{t \approx^?_\alpha u\} \cup \Gamma' \cup \Gamma;\ \sigma,$$

if $t \notin \mathcal{F} \cup \mathcal{A}$ or $u \notin \mathcal{F} \cup \mathcal{A}$. If $\tilde{s} = \tilde{r} = ()$, then $\Gamma' = \emptyset$, otherwise $\Gamma' = \{f(\tilde{s}) \approx^?_\alpha f(\tilde{r})\}$, where f is an arbitrary function symbol.

TD: **Total Decomposition**

$$\{\mathsf{symb}(t_1, \ldots, t_n) \approx^?_\alpha \mathsf{symb}(u_1, \ldots, u_n)\} \uplus \Gamma;\ \sigma \rightsquigarrow \{t_1 \approx^?_\alpha u_1, \ldots, t_n \approx^?_\alpha u_n\} \cup \Gamma;\ \sigma,$$

where $n > 0$.

PD-L: **Partial Decomposition Left**

$$\{\mathsf{symb}(t_1, \ldots, t_n, \overline{x}^P, \tilde{s}) \approx^?_\alpha \mathsf{symb}(u_1, \ldots, u_n, \tilde{r})\} \uplus \Gamma;\ \sigma \rightsquigarrow \{t_1 \approx^?_\alpha u_1, \ldots, t_n \approx^?_\alpha u_n, \mathsf{symb}(\overline{x}^P, \tilde{s}) \approx^?_\alpha \mathsf{symb}(\tilde{r})\} \cup \Gamma;\ \sigma.$$

where $n > 0$.

Q: **Quantifiers**

$$\{Qa.t \approx^?_\alpha Qb.u\} \uplus \Gamma;\ \sigma \rightsquigarrow \{t[a := c] \approx^?_\alpha u[b := c]\} \cup \Gamma;\ \sigma.$$

where $c \notin \mathsf{atoms}(t) \cup \mathsf{atoms}(u)$.

TUE-L: **Term Unknown Elimination Left**

$$\{x^P \approx^?_\alpha u\} \uplus \Gamma;\ \sigma \rightsquigarrow \Gamma\vartheta\rho;\ \sigma\vartheta\rho,$$

where

- $x^P \notin \mathsf{unkn}(u) = \{v_1^{P_1}, \ldots, v_n^{P_n}\}$,
- $\mathsf{fa}(u) \setminus (P_1 \cup \cdots \cup P_n) \subseteq P$,
- $\rho = \{v_i^{P_i} \mapsto w_i^{P_i \cap P} \mid i \in \{1, \ldots, n\}, P_i \cap P \neq P_i\}$ is a renaming substitution with fresh variables w_i, and
- $\vartheta = \{x^P \mapsto u\rho\}$.

The rule below depends on the global parameter ℓ which specifies the maximum length of instantiations of sequence unknowns.[2] It affects completeness, but is necessary for termination.

FIXED-SUE-L: **Fixed-Size Sequence Unknown Elimination Left**

$$\{\mathsf{symb}(\overline{x}^P, \tilde{s}) \approx^?_\alpha \mathsf{symb}(\tilde{r})\} \uplus \Gamma;\ \sigma \rightsquigarrow$$
$$(\{\mathsf{symb}(\overline{x}^P, \tilde{s}) \approx^?_\alpha \mathsf{symb}(\tilde{r})\} \cup \Gamma)\vartheta;\ \sigma\vartheta,$$

where $\vartheta = \{\overline{x}^P \mapsto (x_1^P, \ldots, x_k^P)\}$, where the x's are fresh variables and $k \leq \ell$.

We also have the **Right** counterparts of the **Left** rules. We do not explicitly write them here to save space. They are just dual to the corresponding **Left** rules: If a **Left** rule operates on $t \approx^?_\alpha u$, the right rule would apply to an equation of the form $u \approx^?_\alpha t$. The names of **Right** rules have the suffix -R is place of -L.

To unify two terms t and u, we create the initial state $\{t \approx^?_\alpha u\}; Id$ and apply the abovementioned rules exhaustively, generating derivations. When the Trivial rule T applies to the selected equation, the other rules are not used. If an elimination rule and its right counterpart (i.e., TUE-L and TUE-R, FIXED-SUE-L and FIXED-SUE-R) are applicable to the same equation at the same time, we use only one of them (usually the left one).

FIXED-SUE-L (and FIXED-SUE-R) can transform the same equation in finitely many ways, depending on the choice of k. It can cause branching in the derivation tree, leading to computing multiple answers.

The derivations stop in two cases. Either a state of the form $\emptyset; \sigma$ is generated, or no rule can be applied to the last state $\Gamma; \vartheta$ where $\Gamma \neq \emptyset$. In the first case, the derivation is called successful and $\sigma|_{\mathsf{unkn}(\Gamma)}$ is called the *computed answer*. In the second case, the derivation is called failed.

The described algorithm is denoted by Unif-Alg. The set of answers computed by Unif-Alg for a unification problem Γ with a given ℓ is denoted by Unif-Alg(Γ, ℓ).

There are special fragments of terminating sequence unification (see, e.g., [45]). We can accommodate them in our framework as well, replacing FIXED-SUE-L by rules suitable to the particular fragment. Here we consider two such special cases: (1) when no unknown occurs in the right hand side of an unification problem (sequence matching fragment, SEQ-MATCH) and (2) when all sequence unknowns occur in the last argument positions (sequence last fragment, SEQ-LAST).

[2] Instead of the global parameter ℓ, we could impose individual length-bounds for each sequence unknown occurring in the given unification problem. It would not change the algorithm and its properties.

For Seq-Match, the rule that replaces FIXED-SUE-L is MATCH-SUE.

MATCH-SUE: **Sequence Unknown Elimination,** Seq-Match

$$\{\mathsf{symb}(\overline{x}^P, \tilde{s}) \approx^?_\alpha \mathsf{symb}(\tilde{r}_1, \tilde{r}_2)\} \uplus \Gamma;\ \sigma \rightsquigarrow \\ (\{\mathsf{symb}(\tilde{s}) \approx^?_\alpha \mathsf{symb}(\tilde{r}_2)\} \cup \Gamma)\vartheta;\ \sigma\vartheta,$$

where $\mathsf{fa}(\tilde{r}_1) \subseteq P$ and $\vartheta = \{\overline{x}^P \mapsto \tilde{r}_1\}$.

We do not need the **Right** counterpart of this rule and also for the other elimination rules, since in the matching fragment no unknown occurs in the right hand side.

For Seq-Last, FIXED-SUE-L is replaced by LAST-SUE-L.

LAST-SUE-L: **Sequence Unknown Elimination Left,** Seq-Last

$$\{\mathsf{symb}(\overline{x}^P) \approx^?_\alpha \mathsf{symb}(\tilde{r})\} \uplus \Gamma;\ \sigma \rightsquigarrow \Gamma\vartheta\rho;\ \sigma\vartheta\rho,$$

where

- $\overline{x}^P \notin \mathsf{unkn}(\tilde{r}) = \{v_1^{P_1}, \ldots, v_n^{P_n}\}$,
- $\mathsf{fa}(\tilde{r}) \setminus (P_1 \cup \cdots \cup P_n) \subseteq P$,
- $\rho = \{v_i^{P_i} \mapsto w_i^{P_i \cap P} \mid i \in \{1, \ldots, n\}, P_i \cap P \neq P_i\}$ is a renaming substitution with fresh variables w_i, and
- $\vartheta = \{\overline{x} \mapsto \tilde{r}\rho\}$.

We have also the **Right** counterpart of this rule, called LAST-SUE-R. If both LAST-SUE-L and LAST-SUE-R are applicable to the same equation, we apply only LAST-SUE-L.

For Seq-Match and Seq-Last fragments, there is no global parameter ℓ anymore. Derivations are performed as defined above. The MATCH-SUE rule causes branching, depending on the choice of $\tilde{r}_1$, and leads to finitely many answers. LAST-SUE-L and LAST-SUE-R do not introduce branching. We denote by Match-Alg and Unif-Alg-Last the corresponding algorithms, and by Match-Alg(Γ) and Unif-Alg-Last(Γ) the sets of answers computed by them for Γ.

Example 3. Let Γ be the unification problem $\{f(x^{\{a,b\}}, \lambda b.y^{\{a,c\}}(b)) \approx^?_\alpha f(f(y^{\{a,c\}}), \lambda d.g(z^{\{c\}}, a)(d))\}$. Then Unif-Alg generates the following derivation:

$$\{f(x^{\{a,b\}}, \lambda b.y^{\{a,c\}}(b)) \approx^?_\alpha f(f(y^{\{a,c\}}), \lambda d.g(z^{\{c\}}, a)(d)); Id \rightsquigarrow_{\mathsf{TD}}$$
$$\{x^{\{a,b\}} \approx^?_\alpha f(y^{\{a,c\}}),\ \lambda b.y^{\{a,c\}}(b) \approx^?_\alpha \lambda d.g(z^{\{c\}}, a)(d)\}; Id \rightsquigarrow_{\mathsf{TUE\text{-}L}}$$
$$\{\lambda b.y_1^{\{a\}}(b) \approx^?_\alpha \lambda d.g(z^{\{c\}}, a)(d)\};$$

$$\{x^{\{a,b\}} \mapsto f(y_1^{\{a\}}), y^{\{a,c\}} \mapsto y_1^{\{a\}}\} \rightsquigarrow_{\mathsf{Q}}$$
$$\{y_1^{\{a\}}(d') \approx_\alpha^? g(z^{\{c\}}, a)(d')\}; \{x^{\{a,b\}} \mapsto f(y_1^{\{a\}}), y^{\{a,c\}} \mapsto y_1^{\{a\}}\} \rightsquigarrow_{\mathsf{HD}}$$
$$\{y_1^{\{a\}} \approx_\alpha^? g(z^{\{c\}}, a),\ d' \approx_\alpha^? d'\}; \{x^{\{a,b\}} \mapsto f(y_1^{\{a\}}), y^{\{a,c\}} \mapsto y_1^{\{a\}}\} \rightsquigarrow_{\mathsf{T}}$$
$$\{y_1^{\{a\}} \approx_\alpha^? g(z^{\{c\}}, a)\}; \{x^{\{a,b\}} \mapsto f(y_1^{\{a\}}), y^{\{a,c\}} \mapsto y_1^{\{a\}}\} \rightsquigarrow_{\mathsf{TUE\text{-}L}}$$
$$\emptyset; \{x^{\{a,b\}} \mapsto f(g(z_1^\emptyset, a)), y^{\{a,c\}} \mapsto g(z_1^\emptyset, a), y_1^{\{a\}} \mapsto g(z_1^\emptyset, a), z^{\{c\}} \mapsto z_1^\emptyset\}.$$

Hence, the computed answer is $\{x^{\{a,b\}} \mapsto f(g(z_1^\emptyset, a)),\ y^{\{a,c\}} \mapsto g(z_1^\emptyset, a),\ z^{\{c\}} \mapsto z_1^\emptyset\}$.

Example 4. Let $\Gamma = \{f(\overline{x}^{\{a,b\}}, a, b) \approx_\alpha^? f(a, b, \overline{x}^{\{a,b\}})\}$. Let $\ell = 2$. Then Unif-Alg generates the following derivations:

1. $\{f(\overline{x}^{\{a,b\}}, a, b) \approx_\alpha^? f(a, b, \overline{x}^{\{a,b\}})\};\ Id \rightsquigarrow_{\mathsf{FIXED\text{-}SUE\text{-}L,\,k=0}}$
   $\{f(a, b) \approx_\alpha^? f(a, b)\};\ \{\overline{x}^{\{a,b\}} \mapsto ()\} \rightsquigarrow_{\mathsf{T}}$
   $\emptyset;\ \{\overline{x}^{\{a,b\}} \mapsto ()\}$

2. $\{f(\overline{x}^{\{a,b\}}, a, b) \approx_\alpha^? f(a, b, \overline{x}^{\{a,b\}})\};\ Id \rightsquigarrow_{\mathsf{FIXED\text{-}SUE\text{-}L,\,k=1}}$
   $\{f(x_1^{\{a,b\}}, a, b) \approx_\alpha^? f(a, b, x_1^{\{a,b\}})\};\ \{\overline{x}^{\{a,b\}} \mapsto (x_1^{\{a,b\}})\} \rightsquigarrow_{\mathsf{TD}}$
   $\{x_1^{\{a,b\}} \approx_\alpha^? a,\ a \approx_\alpha^? b,\ b \approx_\alpha^? x_1^{\{a,b\}}\};\ \{\overline{x}^{\{a,b\}} \mapsto (a)\} \rightsquigarrow_{\mathsf{TUE\text{-}L}}$
   $\{a \approx_\alpha^? b,\ b \approx_\alpha^? a\};\ \{\overline{x}^{\{a,b\}} \mapsto (a),\ x_1^{\{a,b\}} \approx_\alpha^? a\}$
   FAIL

3. $\{f(\overline{x}^{\{a,b\}}, a, b) \approx_\alpha^? f(a, b, \overline{x}^{\{a,b\}})\};\ Id \rightsquigarrow_{\mathsf{FIXED\text{-}SUE\text{-}L,\,k=2}}$
   $\{f(x_1^{\{a,b\}}, x_2^{\{a,b\}}, a, b) \approx_\alpha^? f(a, b, x_1^{\{a,b\}}, x_2^{\{a,b\}})\};$
   $\quad\{\overline{x}^{\{a,b\}} \mapsto (x_1^{\{a,b\}}, x_2^{\{a,b\}})\} \rightsquigarrow_{\mathsf{TD}}$
   $\{x_1^{\{a,b\}} \approx_\alpha^? a,\ x_2^{\{a,b\}} \approx_\alpha^? b,\ a \approx_\alpha^? x_1^{\{a,b\}},\ b \approx_\alpha^? x_2^{\{a,b\}}\};$
   $\quad\{\overline{x}^{\{a,b\}} \mapsto (x_1^{\{a,b\}}, x_2^{\{a,b\}})\} \rightsquigarrow_{\mathsf{TUE\text{-}L}}$
   $\{x_2^{\{a,b\}} \approx_\alpha^? b,\ a \approx_\alpha^? a,\ b \approx_\alpha^? x_2^{\{a,b\}}\};$
   $\quad\{\overline{x}^{\{a,b\}} \mapsto (a, x_2^{\{a,b\}}),\ x_1^{\{a,b\}} \mapsto a\} \rightsquigarrow_{\mathsf{TUE\text{-}L}}$
   $\{a \approx_\alpha^? a,\ b \approx_\alpha^? b\};$
   $\quad\{\overline{x}^{\{a,b\}} \mapsto (a, b),\ x_1^{\{a,b\}} \mapsto a,\ x_2^{\{a,b\}} \mapsto b\} \rightsquigarrow_{\mathsf{T}}^2$
   $\emptyset;\ \{\overline{x}^{\{a,b\}} \mapsto (a, b),\ x_1^{\{a,b\}} \mapsto a,\ x_2^{\{a,b\}} \mapsto b\}.$

Hence, $\mathsf{Unif\text{-}Alg}(\Gamma, 2) = \{\{\overline{x}^{\{a,b\}} \mapsto ()\}, \{\overline{x}^{\{a,b\}} \mapsto (a, b)\}\}$.

Example 5. Let $\Gamma = \{f(\overline{x}^{\{a\}}, \overline{y}^{\{a,b,c\}}) \approx^?_\alpha f(a,b,c)\}$. It is a matching problem and we can apply Match-Alg, which generates the following derivations:

1. $\{f(\overline{x}^{\{a\}}, \overline{y}^{\{a,b,c\}}) \approx^?_\alpha f(a,b,c)\};\ Id \rightsquigarrow_{\mathsf{MATCH\text{-}SUE}}$
   $\{f(\overline{y}^{\{a,b,c\}}) \approx^?_\alpha f(a,b,c)\};\ \{\overline{x}^{\{a\}} \mapsto ()\} \rightsquigarrow_{\mathsf{MATCH\text{-}SUE}}$
   $\{f() \approx^?_\alpha f()\};\ \{\overline{x}^{\{a\}} \mapsto (), \overline{y}^{\{a,b,c\}} \mapsto (a,b,c)\} \rightsquigarrow_{\mathsf{T}}$
   $\emptyset;\ \{\overline{x}^{\{a\}} \mapsto (), \overline{y}^{\{a,b,c\}} \mapsto (a,b,c)\}$.

2. $\{f(\overline{x}^{\{a\}}, \overline{y}^{\{a,b,c\}}) \approx^?_\alpha f(a,b,c)\};\ Id \rightsquigarrow_{\mathsf{MATCH\text{-}SUE}}$
   $\{f(\overline{y}^{\{a,b,c\}}) \approx^?_\alpha f(b,c)\};\ \{\overline{x}^{\{a\}} \mapsto (a)\} \rightsquigarrow_{\mathsf{MATCH\text{-}SUE}}$
   $\{f() \approx^?_\alpha f()\};\ \{\overline{x}^{\{a\}} \mapsto (a), \overline{y}^{\{a,b,c\}} \mapsto (b,c)\} \rightsquigarrow_{\mathsf{T}}$
   $\emptyset;\ \{\overline{x}^{\{a\}} \mapsto (a), \overline{y}^{\{a,b,c\}} \mapsto (b,c)\}$.

Hence, Match-Alg$(\Gamma) = \{\{\overline{x}^{\{a\}} \mapsto (), \overline{y}^{\{a,b,c\}} \mapsto (a,b,c)\}, \{\overline{x}^{\{a\}} \mapsto (a), \overline{y}^{\{a,b,c\}} \mapsto (b,c)\}\}$.

If we apply Unif-Alg with $\ell = 1$, there will be no answer computed. All the derivation branches will fail. For any $\ell > 1$ we get the same answers as those computed by Match-Alg.

Example 6. Let $\Gamma = \{f(\overline{x}^{\{a\}}) \approx^?_\alpha f(\overline{y}^{\{a,b,c\}})\}$. Application of Unif-Alg with $\ell = 2$ gives three computed answers:

$$\{\overline{x}^{\{a\}} \mapsto (), \overline{y}^{\{a,b,c\}} \mapsto ()\},\ \{\overline{x}^{\{a\}} \mapsto (z^{\{a\}}), \overline{y}^{\{a,b,c\}} \mapsto (z^{\{a\}})\},$$
$$\{\overline{x}^{\{a\}} \mapsto (z_1^{\{a\}}, z_2^{\{a\}}), \overline{y}^{\{a,b,c\}} \mapsto (z_1^{\{a\}}, z_2^{\{a\}})\}.$$

The problem also falls in the Seq-Last fragment. Unif-Alg-Last gives only one computed answer: $\{\overline{x}^{\{a\}} \mapsto (\overline{y}_1^{\{a\}}), \overline{y}^{\{a,b,c\}} \mapsto (\overline{y}_1^{\{a\}})\}$.

4 Properties of the algorithm

First, we define the sizes of an s-term, an equation, and a unification problem:

$$\mathsf{size}(f) = \mathsf{size}(a) = 1$$
$$\mathsf{size}(x^P) = \mathsf{size}(\overline{x}^P) = 2.$$
$$\mathsf{size}(t(s_1, \ldots, s_n)) = \mathsf{size}(t) + \mathsf{size}((s_1, \ldots, s_n)) + 1.$$
$$\mathsf{size}(Qa.t) = size(t) + 1.$$
$$\mathsf{size}(()) = 0.$$
$$\mathsf{size}((s_1, \ldots, s_n)) = \mathsf{size}(s_1) + \cdots + \mathsf{size}(s_n).$$

$\mathsf{size}(t \approx^?_\alpha u) = \mathsf{size}(t) + \mathsf{size}(u)$.

$\mathsf{size}(\Gamma) = \{\!\{\mathsf{size}(t \approx^?_\alpha u) \mid t \approx^?_\alpha u \in \Gamma\}\!\}$, where $\{\!\{\cdot\}\!\}$ stand for multiset.

Theorem 1. Unif-Alg, Match-Alg, *and* Unif-Alg-Last *terminate.*

Proof. With each state $\Gamma;\sigma$, we associate its complexity measure, a triple $\langle n_1, n_2, M\rangle$, where n_1 and n_2 are respectively the numbers of distinct term and sequence unknowns occurring in Γ, and $M = \mathsf{size}(\Gamma)$. The measures are compared lexicographically, where the first two components are compared by the standard ordering on natural numbers, and the third component is compared by the multiset extension of the standard natural number ordering. The obtained ordering on complexity measures is well-founded. The table below shows that each rule of our algorithms strictly reduces this measure (i.e., if a rule transforms $\Gamma_1;\sigma_1$ into $\Gamma_2;\sigma_2$, then the measure of Γ_2 is strictly smaller than the measure of Γ_1), which implies that Unif-Alg, Match-Alg and Unif-Alg-Last terminate.

Rules	n_1	n_2	M
FIXED-SUE-L, FIXED-SUE-R	$>$		
LAST-SUE-L, LAST-SUE-R	$>$		
MATCH-SUE	$>$		
TUE-L, TUE-R	$=$	$>$	
T	$\geq$	$\geq$	$>$
TD	$=$	$\geq$	$>$
HD, PD-L, PD-R, Q	$=$	$=$	$>$

□

For the other properties of our algorithms, we need the following lemma:

Lemma 1. *If* $\Gamma_1;\sigma \rightsquigarrow \Gamma_2;\sigma\psi$ *is a rule application, then* $\Gamma_1\psi$ *and* Γ_2 *have the same sets of unifiers.*

Proof. Assume the derivation step is made by the TUE-L rule. Then $\psi = \rho\vartheta$, $\Gamma_1 = \{x^P \approx^?_\alpha u\} \uplus \Gamma$, and $\Gamma_2 = \Gamma\rho\vartheta$, where ρ and ϑ are as defined by the rule. We have $x^P\rho\vartheta = u\rho$, $u\rho\vartheta = u\rho$ and, hence, $\Gamma_1\rho\vartheta = \{u\rho \approx^?_\alpha u\rho\} \cup \Gamma\rho\vartheta$. Obviously, $\Gamma_1\rho\vartheta$ and $\Gamma\rho\vartheta$ have the same set of unifiers i.e., $\Gamma_1\psi$ and Γ_2 have the same set of unifiers.

The proof is analogous for the other elimination rules. For trivial, decomposition, and quantifier rules the theorem follows directly from the definition of α-equivalence. □

Theorem 2 (Soundness of Unif-Alg)**.** *For a unification problem* Γ *and a length bound* ℓ*, every substitution* $\sigma \in \mathsf{comp}(\text{Unif-Alg}, \Gamma, \ell)$ *is a unifier of* Γ*.*

Proof. Since $\sigma \in \mathsf{comp}(\mathsf{Unif\text{-}Alg}, \Gamma, \ell)$, there exists a derivation in Unif-Alg (with ℓ) of the form $\Gamma; Id \rightsquigarrow^{+} \emptyset; \sigma$. Then the theorem can be proved by using the induction on the length of the derivation and Lemma 1. □

The Match-Alg and Unif-Alg-Last algorithms are sound as well. The corresponding theorems below can be proved similarly to Theorem 2.

Theorem 3 (Soundness of Match-Alg)**.** *If Γ is a matching problem, then every $\sigma \in \mathsf{Match\text{-}Alg}(\Gamma)$ is a matcher of Γ.*

Theorem 4 (Soundness of Unif-Alg-Last)**.** *If Γ is a unification problem where every sequence unknown appears in the last argument position, and $\mathsf{Unif\text{-}Alg\text{-}Last}(\Gamma) = \{\sigma\}$, then σ is a unifier of Γ.*

Unif-Alg is not complete, in general. It is obvious, since the length restriction on the instantiation of sequence unknowns, imposed by the parameter ℓ, prevents to compute unifiers in which the lengths of sequence unknown instances are larger than ℓ. For example, when $\ell = 2$, Unif-Alg can not compute the unifier $\{\overline{x}^{\{a\}} \mapsto (a, a, a)\}$ of the unification problem $f(\overline{x}^{\{a\}}, a) \approx^{?}_{\alpha} f(a, \overline{x}^{\{a\}})$.

Interestingly, there is another reason of incompleteness of Unif-Alg, which is caused by the fact that a sequence unknown is always replaced by a sequence of term unknowns. Because of this, Unif-Alg can not compute a most general solution $\{\overline{x}^P \mapsto (\overline{y}^P)\}$ of $\Gamma = \{f(\overline{x}^P) \approx^{?}_{\alpha} f(\overline{y}^P)\}$. Instead, it returns ℓ solutions $\{\overline{x}^P \mapsto (), \overline{x}^P \mapsto ()\}$, $\{\overline{x}^P \mapsto (x^P), \overline{y}^P \mapsto (x^P)\}$, ..., $\{\overline{x}^P \mapsto (x_1^P, \ldots, x_\ell^P), \overline{y}^P \mapsto (x_1^P, \ldots, x_\ell^P)\}$.

However, the following restricted version of completeness holds:

Theorem 5 (Restricted completeness of Unif-Alg)**.** *Let Γ be a unification problem and φ be its unifier such that $ran(\varphi)$ does not contain sequence unknowns. Then there exist ℓ and $\sigma \in \mathsf{comp}(\mathsf{Unif\text{-}Alg}, \ell)$ such that $\sigma|_{\mathsf{unkn}(\Gamma)} \precsim \varphi$.*

Proof. First, consider the case when Γ does not contain sequence unknowns. Assume without loss of generality that φ is idempotent and $dom(\varphi) \subseteq \mathsf{unkn}(\Gamma)$.

We will construct a derivation $\Gamma_1; \sigma_1 \rightsquigarrow^{+} \Gamma_n; \sigma_n$, where $\Gamma_1 = \Gamma$, $\sigma_1 = Id$, $\Gamma_n = \emptyset$, and for each $1 \leq i \leq n$, there exists a substitution ψ_i such that

- $\varphi\psi_i$ is an idempotent unifier of Γ_i,
- $dom(\psi_i|_{\mathsf{unkn}(\Gamma)}) \subseteq dom(\varphi)$,
- $\sigma_i \precsim \varphi\psi_i$.

(For $i = 1$ such a ψ_i obviously exists: it is Id.)

If we build such a derivation, we get $\sigma_n \precsim \varphi\psi_n$, which implies that $\sigma_n|_{\mathsf{unkn}(\Gamma)} \precsim (\varphi\psi_n)|_{\mathsf{unkn}(\Gamma)} = \varphi$ and we can take $\sigma = \sigma_n$.

Assume we have constructed $\Gamma_1; \sigma_1 \rightsquigarrow^* \Gamma_i; \sigma_i$ in this derivation and show how to make the step $\Gamma_i; \sigma_i \rightsquigarrow \Gamma_{i+1}; \sigma_{i+1}$.

We pick up an equation arbitrarily from Γ_i, represent the unification problem as $\Gamma_i = \{t \approx^?_\alpha u\} \uplus \Gamma'_i$, and proceed by case distinction on the form of $t \approx^?_\alpha u$.

If $t = u$, then the step is made by the T rule and $\Gamma_{i+1}; \sigma_{i+1}$ obviously satisfies all the desired properties. Assume $t \neq u$. We distinguish the following cases:

$\mathsf{head}(t) = \mathsf{head}(u)$. The applicable rules are TD or Q. In each case, it is easy to see that the obtained state is what we need.

$\mathsf{head}(t) \neq \mathsf{head}(u)$ and none of these terms is an unknown. Then the only possible case is $\mathsf{head}(t) \notin \mathcal{F} \cup \mathcal{A}$, or $\mathsf{head}(u) \notin \mathcal{F} \cup \mathcal{A}$. Otherwise Γ_i would not be unifiable. We apply the HD rule. Again, the obtained state satisfies the desired properties.

$\mathsf{head}(t) \neq \mathsf{head}(u)$ and at least one of them is an unknown x^P. Assume without loss of generality that it is t. hence, we have an equation $x^P \approx^?_\alpha u$. If $x^P \in \mathsf{unkn}(u)$, then for any substitution ϑ we will have $\mathsf{size}(x^P\vartheta) < \mathsf{size}(u\vartheta)$ and Γ_i would not be unifiable.

Assume $x^P \notin \mathsf{unkn}(u)$. Then we apply $\mathsf{TUE\text{-}L}$ rule and get $\Gamma_{i+1} = \Gamma'\vartheta_{i+1}\rho_{i+1}$, $\sigma_{i+1} = \sigma_i\vartheta_{i+1}\rho_{i+1}$, where

$$\rho_{i+1} = \{v^R \mapsto w^{P\cap R} \mid v^R \in \mathsf{unkn}(u), P \cap R \neq R, w \text{ is fresh}\},$$
$$\vartheta_{i+1} = \{x^P \mapsto u\rho_{i+1}\}.$$

We need to find ψ_{i+1} such that

- $\sigma_{i+1} = \sigma_i\vartheta_{i+1}\rho_{i+1} \precsim \varphi\psi_{i+1}$,
- $dom(\psi_{i+1}|_{\mathsf{unkn}(\Gamma)}) \subseteq dom(\varphi)$, and
- $\varphi\psi_{i+1}$ is an idempotent unifier of Γ_{i+1}.

Since $\sigma_i \precsim \varphi\psi_i$, there exists a substitution ν such that $v^R\sigma_i\nu \approx_\alpha v^R\varphi\psi_i$ for all v^R. On the other hand, $\varphi\psi_i$ is a unifier of $x^P \approx^?_\alpha u$. This gives $x^P\nu = x^P\sigma_i\nu \approx_\alpha u\sigma_i\nu = u\nu$. (The σ's are idempotent, therefore, x^P and unknowns from u are not in the domain of σ_i.)

Besides, we have

$$v^R\vartheta_{i+1}\rho_{i+1}\nu\mu \approx_\alpha v^R\nu\mu \text{ for any } v^R. \tag{1}$$

Define μ as

$$\mu = \{v^R\rho_{i+1} \mapsto v^R\nu \mid v^R \in \mathsf{unkn}(u)\}.$$

It is a well-defined substitution: $v^R \in \mathsf{unkn}(u)$ and we have $\mathsf{fa}(v^R\nu) \subseteq R$, since ν is a unifier of $x^P \approx^?_\alpha u$.

Let $\psi_{i+1} = \psi_i\mu$. Then we have $dom(\psi_{i+1}|_{\mathsf{unkn}(\Gamma)}) \subseteq dom(\varphi)$. Since $\varphi\psi_i$ is an idempotent unifier of Γ_i, we get that $\varphi\psi_{i+1}$ is an idempotent unifier of Γ_{i+1}.

Note that ρ_{i+1}, ϑ_{i+1}, μ and ψ_{i+1} are idempotent. Besides, $v^R\rho_{i+1}\mu = v^R\varphi\psi_i\mu$ for every $v^R \in dom(\rho_{i+1}) \cup ran(\rho_{i+1})$. Therefore, we get

- $v^R\rho_{i+1}\nu\mu = v^R\nu\nu\mu = v^R\nu\mu = v^R\varphi\psi_i\mu = v^R\varphi\psi_{i+1}$ for every $v^R \in dom(\rho_{i+1}) \cup ran(\rho_{i+1})$,
- $v^R\rho_{i+1}\nu\mu = v^R\nu\mu = v^R\varphi\psi_i\mu = v^R\varphi\psi_{i+1}$ for every other v^R.

In order to show $\sigma_{i+1} \precsim \varphi\psi_{i+1}$, we will prove $w^T\sigma_{i+1}\nu\mu \approx_\alpha w^T\varphi\psi_{i+1}$ for all w^T.

Let w^T be x^P. Since $\varphi\psi_i$ solves $x^P \approx^?_\alpha u$, we have

$$\begin{aligned} x^P\sigma_{i+1}\nu\mu &= x^P\sigma_i\vartheta_{i+1}\rho_{i+1}\nu\mu = x^P\vartheta_{i+1}\rho_{i+1}\nu\mu \\ &\approx_\alpha u\rho_{i+1}\rho_{i+1}\nu\mu = u\rho_{i+1}\nu\mu = u\nu\nu\mu = u\nu\mu \\ &\approx_\alpha x^P\nu\mu = x^P\varphi\psi_i\mu \\ &= x^P\varphi\psi_{i+1}. \end{aligned} \tag{2}$$

Now let $w^T \neq x^P$. For every unknown v^R from $w^T\sigma_i\vartheta_{i+1}$ we have $v^R\rho_{i+1}\varphi\psi_i\mu = v^R\varphi\psi_i\mu$. Therefore using (1), we get

$$w^T\sigma_{i+1}\nu\mu = w^T\sigma_i\vartheta_{i+1}\rho_{i+1}\nu\mu = w^T\sigma_i\nu\mu = w^T\varphi\psi_i\mu w^T\varphi\psi_{i+1}. \tag{3}$$

Hence, $\sigma_{i+1} \precsim \varphi\psi_{i+1}$. It finishes the proof that for any unifier φ of Γ, the algorithm Unif-Alg computes σ with the property $\sigma|_{\mathsf{unkn}(\Gamma)} \precsim \varphi$, when Γ does not contain sequence unknowns.

Now assume Γ contains sequence unknowns. Let $l := \max\{|\overline{x}^R\varphi| \mid \overline{x} \in dom(\varphi)\}$, i.e., l is the length of the longest sequence to which a sequence unknown from $dom(\varphi)$ is mapped. By fixing $\ell = l$, we can compute $\sigma \in \mathsf{Unif\text{-}Alg}(\Gamma, \ell)$ with the property $\sigma|_{\mathsf{unkn}(\Gamma)} \precsim \varphi$. □

The completeness theorems for Match-Alg and Unif-Alg-Last are easier to prove. We just state them here:

Theorem 6 (Completeness of Match-Alg)**.** *Let φ be a matcher of a matching problem Γ. Then* Match-Alg *computes a σ such that $\sigma = \varphi|_{\mathsf{unkn}(\Gamma)}$.*

Theorem 7 (Completeness of Unif-Alg-Last)**.** *Let Γ be a unification problem where every sequence unknown appears in the last argument position, and φ be its unifier. Then there exists $\sigma \in$ Unif-Alg-Last(Γ) such that $\sigma \precsim \varphi$.*

The sets Unif-Alg(Γ, ℓ) and Match-Alg(Γ) are minimal. This follows from the fact that if there are two distinct σ_1 and σ_2 in such a set, then there exists $\overline{x}^P \in dom(\sigma_1) \cap dom(\sigma_2)$ such that the length of their instantiations are different: $|\overline{x}^P\sigma_1| \neq |\overline{x}^P\sigma_2|$. Such a difference can not be repaired by a substitution composition, because the ranges of σ's do not contain sequence markers by construction. Hence, we have neither $\sigma_1 \precsim \sigma_2$ nor $\sigma_2 \precsim \sigma_1$, which implies minimality.

The set Unif-Alg-Last(Γ) is singleton, since there is no branching in the derivation tree. The computed unifier is most general.

5 Conclusion

We described three algorithms for solving unification problems and their fragments for terms containing unknowns with permission sets, variadic function constants, atoms, applications, and binders that bind atoms. The design is guided by the syntax of Theorema system, where higher-order expressions are permitted. Unification and matching equations are solved modulo α-equivalence. Termination, soundness, and (restricted) completeness of algorithms are proved. They are implemented as a part of the Theorema system.

Acknowledgment

This work is partially supported by the Austrian Science Fund (FWF) under the project P 28789-N32.

References

1. Asperti, A., Ricciotti, W., and Sacerdoti Coen, C. Matita tutorial. *J. Formalized Reasoning 7*, 2 (2014), 91–199.

2. Ayala-Rincón, M., de Carvalho Segundo, W., Fernández, M., and Nantes-Sobrinho, D. Nominal C-unification. In *Logic-Based Program Synthesis and Transformation - 27th International Symposium, LOPSTR 2017, Namur, Belgium, October 10-12, 2017, Revised Selected Papers* (2017), F. Fioravanti and J. P. Gallagher, Eds., vol. 10855 of *Lecture Notes in Computer Science*, Springer, pp. 235–251.

3. Ayala-Rincón, M., Fernández, M., and Nantes-Sobrinho, D. Fixed-point constraints for nominal equational unification. In *3rd International Conference on Formal Structures for Computation and De-*

duction, FSCD 2018, July 9-12, 2018, Oxford, UK (2018), H. Kirchner, Ed., vol. 108 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, pp. 7:1–7:16.

4. Baader, F., and Snyder, W. Unification theory. In *Handbook of Automated Reasoning (in 2 volumes)*, J. A. Robinson and A. Voronkov, Eds. Elsevier and MIT Press, 2001, pp. 445–532.

5. Bancerek, G., Bylinski, C., Grabowski, A., Kornilowicz, A., Matuszewski, R., Naumowicz, A., Pak, K., and Urban, J. Mizar: State-of-the-art and beyond. In *Intelligent Computer Mathematics - International Conference, CICM 2015, Washington, DC, USA, July 13-17, 2015, Proceedings* (2015), M. Kerber, J. Carette, C. Kaliszyk, F. Rabe, and V. Sorge, Eds., vol. 9150 of *Lecture Notes in Computer Science*, Springer, pp. 261–279.

6. Baumgartner, A., Kutsia, T., Levy, J., and Villaret, M. Nominal anti-unification. In *26th International Conference on Rewriting Techniques and Applications, RTA 2015, June 29 to July 1, 2015, Warsaw, Poland* (2015), M. Fernández, Ed., vol. 36 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, pp. 57–73.

7. Bertot, Y., and Castéran, P. *Interactive Theorem Proving and Program Development - Coq'Art: The Calculus of Inductive Constructions*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004.

8. Buchberger, B. Mathematica: Doing mathematics by computer? In *Advances in the Design of Symbolic Computation Systems*, A. Miola and M. Temperini, Eds., RISC Book Series on Symbolic Computation. Springer Vienna, 1997, pp. 2–20.

9. Buchberger, B., and Craciun, A. Algorithm synthesis by Lazy Thinking: Examples and implementation in Theorema. *Electr. Notes Theor. Comput. Sci. 93* (2004), 24–59.

10. Buchberger, B., Jebelean, T., Kutsia, T., Maletzky, A., and Windsteiger, W. Theorema 2.0: Computer-assisted natural-style mathematics. *J. Formalized Reasoning 9*, 1 (2016), 149–185.

11. Calvès, C., and Fernández, M. A polynomial nominal unification algorithm. *Theor. Comput. Sci. 403*, 2-3 (2008), 285–306.

12. Calvès, C., and Fernández, M. Matching and alpha-equivalence check for nominal terms. *J. Comput. Syst. Sci. 76*, 5 (2010), 283–301.

13. Cheney, J. Equivariant unification. *J. Autom. Reasoning 45*, 3 (2010), 267–300.

14. Coelho, J., Dundua, B., Florido, M., and Kutsia, T. A rule-based approach to XML processing and Web reasoning. In *Web Reasoning and Rule Systems - Fourth International Conference, RR 2010, Bressanone/Brixen, Italy, September 22-24, 2010. Proceedings* (2010), P. Hitzler and T. Lukasiewicz, Eds., vol. 6333 of *Lecture Notes in Computer Science*, Springer, pp. 164–172.

15. Coelho, J., and Florido, M. CLP(Flex): Constraint logic programming applied to XML processing. In *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE, OTM Confederated International Conferences, Agia Napa, Cyprus, October 25-29, 2004, Proceedings, Part II* (2004), R. Meersman and Z. Tari, Eds., vol. 3291 of *Lecture Notes in Computer Science*, Springer, pp. 1098–1112.

16. Colton, S. *Automated Theory Formation in Pure Mathematics*. Distinguished dissertations. Springer, 2002.

17. Dowek, G., Gabbay, M. J., and Mulligan, D. P. Permissive nominal terms and their unification: an infinite, co-infinite approach to nominal techniques. *Logic Journal of the IGPL 18*, 6 (2010), 769–822.

18. Dramnesc, I., Jebelean, T., and Stratulat, S. Mechanical synthesis of sorting algorithms for binary trees by logic and combinatorial techniques. *J. Symb. Comput. 90* (2019), 3–41.

19. Dundua, B. *Programming with Sequence and Context Variables: Foundations and Applications*. PhD thesis, University of Porto, 2014.

20. Dundua, B., Kutsia, T., and Marin, M. Strategies in Pρlog. In *Proceedings Ninth International Workshop on Reduction Strategies in Rewriting and Programming, WRS 2009, Brasilia, Brazil, 28th June 2009* (2009), M. Fernández, Ed., vol. 15 of *EPTCS*, pp. 32–43.

21. Dundua, B., Kutsia, T., and Marin, M. Variadic equational matching. In *Intelligent Computer Mathematics - 12th International Conference, CICM 2019, Prague, Czech Republic, July 8-12, 2019, Proceedings* (2019), C. Kaliszyk, E. Brady, A. Kohlhase, and C. Sacerdoti Coen, Eds., vol. 11617 of *Lecture Notes in Computer Science*, Springer, pp. 77–92.

22. Fernández, M., and Gabbay, M. Nominal rewriting. *Inf. Comput. 205*, 6 (2007), 917–965.

23. Gabbay, M., and Pitts, A. M. A new approach to abstract syntax involving binders. In *14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, July 2-5, 1999* (1999), IEEE Computer Society, pp. 214–224.

24. Gabbay, M., and Pitts, A. M. A new approach to abstract syntax with variable binding. *Formal Asp. Comput. 13*, 3-5 (2002), 341–363.

25. Gabbay, M. J. *A Theory of Inductive Definitions with alpha-Equivalence*. PhD thesis, University of Cambridge, UK, 2001.

26. Gabbay, M. J. Nominal terms and nominal logics: from foundations to meta-mathematics. In *Handbook of Philosophical Logic*, vol. 17. Kluwer, 2013, pp. 79–178.

27. Gabbay, M. J., and Wirth, C. Quantifiers in logic and proof-search using permissive-nominal terms and sets. *J. Log. Comput. 25*, 2 (2015), 473–523.

28. Genesereth, M. R., and Fikes, R. E. Knowledge Interchange Format, Version 3.0 Reference Manual. Tech. Rep. Logic-92-1, Stanford University, Stanford, CA, USA, 1992.

29. Ginsberg, M. L. The MVL theorem proving system. *SIGART Bulletin 2*, 3 (1991), 57–60.

30. Gordon, M. J. C., and Melham, T. F., Eds. *Introduction to HOL: a theorem proving environment for higher order logic*. Cambridge University Press, 1993.

31. Harrison, J. HOL light: An overview. In *Theorem Proving in Higher Order Logics, 22nd International Conference, TPHOLs 2009, Munich, Germany, August 17-20, 2009. Proceedings* (2009), S. Berghofer, T. Nipkow, C. Urban, and M. Wenzel, Eds., vol. 5674 of *Lecture Notes in Computer Science*, Springer, pp. 60–66.

32. Horozal, F., Rabe, F., and Kohlhase, M. Flexary operators for formalized mathematics. In *Intelligent Computer Mathematics - International Conference, CICM 2014, Coimbra, Portugal, July 7-11, 2014. Proceedings* (2014), S. M. Watt, J. H. Davenport, A. P. Sexton, P. Sojka, and J. Urban, Eds., vol. 8543 of *Lecture Notes in Computer Science*, Springer, pp. 312–327.

33. ISO/IEC. Information technology—Common Logic (CL): A framework for a family of logic-based languages. International Standard ISO/IEC 24707:2018, 2018. `https://www.iso.org/standard/66249.html`.

34. Johansson, M. Automated theory exploration for interactive theorem proving: - an introduction to the Hipster system. In *Interactive Theorem Proving - 8th International Conference, ITP 2017, Brasília, Brazil, September 26-29, 2017, Proceedings* (2017), M. Ayala-Rincón and C. A. Muñoz, Eds., vol. 10499 of *Lecture Notes in Computer Science*, Springer, pp. 1–11.

35. Johansson, M., Dixon, L., and Bundy, A. Conjecture synthesis for inductive theories. *J. Autom. Reasoning 47*, 3 (2011), 251–289.

36. Kaufmann, M., Moore, J. S., and Manolios, P. *Computer-Aided Reasoning: An Approach.* Kluwer Academic Publishers, Norwell, MA, USA, 2000.

37. Kerber, M., Rowat, C., and Windsteiger, W. Using Theorema in the formalization of theoretical economics. In *Intelligent Computer Mathematics - 18th Symposium, Calculemus 2011, and 10th International Conference, MKM 2011, Bertinoro, Italy, July 18-23, 2011. Proceedings* (2011), J. H. Davenport, W. M. Farmer, J. Urban, and F. Rabe, Eds., vol. 6824 of *Lecture Notes in Computer Science*, Springer, pp. 58–73.

38. Konev, B., and Jebelean, T. Combining level-saturation strategies and meta-variables for predicate logic proving in Theorema. RISC Report Series 00-40, Research Institute for Symbolic Computation (RISC), Johannes Kepler University Linz, Austria, 2000.

39. Kutsia, T. Solving and proving in equational theories with sequence variables and flexible arity symbols. RISC Report Series 02-09, Research Institute for Symbolic Computation (RISC), Johannes Kepler University Linz, Austria, 2002. PhD Thesis.

40. Kutsia, T. Theorem proving with sequence variables and flexible arity symbols. In *Logic for Programming, Artificial Intelligence, and Reasoning, 9th International Conference, LPAR 2002, Tbilisi, Georgia, October 14-18, 2002, Proceedings* (2002), M. Baaz and A. Voronkov, Eds., vol. 2514 of *Lecture Notes in Computer Science*, Springer, pp. 278–291.

41. Kutsia, T. Unification with sequence variables and flexible arity symbols and its extension with pattern-terms. In *Artificial Intelligence,*

Automated Reasoning, and Symbolic Computation, Joint International Conferences, AISC 2002 and Calculemus 2002, Marseille, France, July 1-5, 2002, Proceedings (2002), J. Calmet, B. Benhamou, O. Caprotti, L. Henocque, and V. Sorge, Eds., vol. 2385 of *Lecture Notes in Computer Science*, Springer, pp. 290–304.

42. Kutsia, T. Equational prover of Theorema. In *Rewriting Techniques and Applications, 14th International Conference, RTA 2003, Valencia, Spain, June 9-11, 2003, Proceedings* (2003), R. Nieuwenhuis, Ed., vol. 2706 of *Lecture Notes in Computer Science*, Springer, pp. 367–379.

43. Kutsia, T. Solving equations with sequence variables and sequence functions. *J. Symb. Comput. 42*, 3 (2007), 352–388.

44. Kutsia, T., and Marin, M. Can context sequence matching be used for querying XML? In *Proceedings of the 19th International Workshop on Unification (UNIF'05)* (Nara, Japan, 22 Apr. 2005), L. Vigneron, Ed., pp. 77–92.

45. Kutsia, T., and Marin, M. Solving, reasoning, and programming in common logic. In *14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2012, Timisoara, Romania, September 26-29, 2012* (2012), A. Voronkov, V. Negru, T. Ida, T. Jebelean, D. Petcu, S. M. Watt, and D. Zaharie, Eds., IEEE Computer Society, pp. 119–126.

46. Levy, J., and Villaret, M. An efficient nominal unification algorithm. In *Proceedings of the 21st International Conference on Rewriting Techniques and Applications, RTA 2010, July 11-13, 2010, Edinburgh, Scottland, UK* (2010), C. Lynch, Ed., vol. 6 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, pp. 209–226.

47. Levy, J., and Villaret, M. Nominal unification from a higher-order perspective. *ACM Trans. Comput. Log. 13*, 2 (2012), 10:1–10:31.

48. Maletzky, A. Mathematical theory exploration in Theorema: Reduction rings. In *Intelligent Computer Mathematics - 9th International Conference, CICM 2016, Bialystok, Poland, July 25-29, 2016, Proceedings* (2016), M. Kohlhase, M. Johansson, B. R. Miller, L. de Moura, and F. W. Tompa, Eds., vol. 9791 of *Lecture Notes in Computer Science*, Springer, pp. 3–17.

49. McCasland, R. L., Bundy, A., and Smith, P. F. MATHsAiD: Automated mathematical theory exploration. *Appl. Intell. 47*, 3 (2017), 585–606.

50. Menzel, C. Knowledge representation, the World Wide Web, and the evolution of logic. *Synthese 182*, 2 (2011), 269–295.

51. Montano-Rivas, O., McCasland, R. L., Dixon, L., and Bundy, A. Scheme-based theorem discovery and concept invention. *Expert Syst. Appl. 39*, 2 (2012), 1637–1646.

52. Owre, S., Rushby, J. M., and Shankar, N. PVS: A prototype verification system. In *Automated Deduction - CADE-11, 11th International Conference on Automated Deduction, Saratoga Springs, NY, USA, June 15-18, 1992, Proceedings* (1992), D. Kapur, Ed., vol. 607 of *Lecture Notes in Computer Science*, Springer, pp. 748–752.

53. Paulson, L. C., Nipkow, T., and Wenzel, M. From LCF to Isabelle/HOL. *Formal Asp. Comput. 31*, 6 (2019), 675–698.

54. Pease, A., and Sutcliffe, G. First order reasoning on a large ontology. In *Proceedings of the CADE-21 Workshop on Empirically Successful Automated Reasoning in Large Theories* (2007), G. Sutcliffe and S. Schulz, Eds., no. 257 in CEUR Workshop Proceedings, pp. 59–69.

55. Pkhakadze, S. *Some problems of the notation theory.* Tbilisi University Press, Tbilisi, Georgia, 1977. In Russian.

56. Richardson, J., and Fuchs, N. E. Development of correct transformation schemata for prolog programs. In *LOPSTR* (1997), N. E. Fuchs, Ed., vol. 1463 of *Lecture Notes in Computer Science*, Springer, pp. 263–281.

57. Rosenkranz, M. A new symbolic method for solving linear two-point boundary value problems on the level of operators. *J. Symb. Comput. 39*, 2 (2005), 171–199.

58. Schmidt-Schauss, M., Kutsia, T., Levy, J., and Villaret, M. Nominal unification of higher order expressions with recursive let. In *Logic-Based Program Synthesis and Transformation - 26th International Symposium, LOPSTR 2016, Edinburgh, UK, September 6-8, 2016, Revised Selected Papers* (2016), M. V. Hermenegildo and P. López-García, Eds., vol. 10184 of *Lecture Notes in Computer Science*, Springer, pp. 328–344.

59. Scott, J. D., Flener, P., Pearson, J., and Schulte, C. Design and implementation of bounded-length sequence variables. In *Integration of AI and OR Techniques in Constraint Programming - 14th International Conference, CPAIOR 2017, Padua, Italy, June 5-8, 2017,*

Proceedings (2017), D. Salvagnin and M. Lombardi, Eds., vol. 10335 of *Lecture Notes in Computer Science*, Springer, pp. 51–67.

60. Steen, A., and Benzmüller, C. The higher-order prover Leo-III (extended abstract). In *KI 2019: Advances in Artificial Intelligence - 42nd German Conference on AI, Kassel, Germany, September 23-26, 2019, Proceedings* (2019), C. Benzmüller and H. Stuckenschmidt, Eds., vol. 11793 of *Lecture Notes in Computer Science*, Springer, pp. 333–337.

61. Urban, C., Pitts, A. M., and Gabbay, M. Nominal unification. *Theor. Comput. Sci. 323*, 1-3 (2004), 473–497.

62. Vasaru Dupre, D. Automated Theorem Proving by Integrating Proving, Solving and Computing. RISC Report Series 00-19, Research Institute for Symbolic Computation (RISC), Johannes Kepler University Linz, Austria, 2000.

63. Windsteiger, W. An automated prover for Zermelo-Fraenkel set theory in Theorema. *J. Symb. Comput. 41*, 3-4 (2006), 435–470.

64. Windsteiger, W. Theorema 2.0: A graphical user interface for a mathematical assistant system. In *Proceedings 10th International Workshop On User Interfaces for Theorem Provers, UITP 2012, Bremen, Germany, July 11th, 2012* (2012), C. Kaliszyk and C. Lüth, Eds., vol. 118 of *EPTCS*, pp. 72–82.

65. Wolfram, S. *The Mathematica book, 5th Edition.* Wolfram-Media, 2003.

THE SYNTAX-SEMANTICS INTERFACE: FROM MONTAGUE GRAMMAR TO TODAY

Aleksandre Maskharashvili

Ohio State University, Ohio, USA
maskharashvili.1@osu.edu

Abstract

In this paper, we overview logical studies of the syntax-semantics interface, with the focus on Montague's seminal works in the field, called Montague Grammar. We also discuss more recent developments of Montague Grammar on the example of Abstract Categorial Grammars. We briefly illustrate how recent developments in Montague's line of research contribute to the studies in classical natural language problems such as parsing and generation. We also sketch an approach that makes use of Abstract Categorial Grammars to establish interrelations between the surface form and the meaning of a discourse.

1 Introduction

Studying regularities of forms across natural language expressions was a subject of study already in the first known grammatical system, created by Pāṇini for Sanskrit. One of the first interest in mathematical study of natural language syntax emerged in the works of Ajdukiewicz [1] who proposed a formal system of rules, called Categorial Grammar, which was supposed to represent the ways natural language sentences are built. One of the main insights in Ajdukiewicz's [1] categorial grammar is to use *functor categories* (*Funktoren-Kategorien*) and *basic categories* (*Grundkategorie*), where a functor category carries functional characteristics and a basic category is anything that is not a functor category. Bar-Hillel [2] further elaborated ideas of Ajdukiewicz [1].

Advances in proof theory motivated Lambek [16] to make use of the ideas of Ajdukiewicz and Bar-Hillel to provide a logical, proof-theoretic model of natural language syntax.

Chomsky [5] proposed Context-Free Grammars (CFGs) to study natural language syntax. His approach was based on Bloomfield's [4] principle of immediate constituents.

Shaumyan [30, 31] developed another line of study in natural language syntax in a setting based on Curry's theory of types and combinators.

In concordance with that, he called his framework Applicative Universal Grammar (AUG).

As studies in syntax had been developing in various directions, Montague [21, 19, 20] proposed a program of studying meaning in natural language. He designed a way to translate syntactic expressions to their semantic forms (meanings). Montague made use of Ajdukiewicz's syntactic calculus as a model of syntax. To encode semantic representations, Montague has developed his language, which from today's perspective can be seen as a version of higher-order logic (HOL). Below, we may refer to Montague's works as Montague Grammar (MG).

2 AB-Calculus

We follow Moot and Retoré [22] to define Ajdukiewicz's [1] (and Bar-Hillel's [2]) version of categorial grammar, called AB-calculus or AB-grammar. In an AB-calculus, an expression is a category (type) if and only if it is derivable from a set of basic categories, denoted by P, in following way:

$$L := P \mid (L\backslash L) \mid (L/L) \tag{1}$$

If v and u are categories, then $u\backslash v$ and v/u are such *functor* categories which take an expression of type u as its argument and produce an expression of type v. The difference between $u\backslash v$ and v/u is *directional*, that is, $u\backslash v$ receives an argument from *right*, whereas v/u receives it from *left*. These possibilities of receiving arguments (from right and from left) are encoded with the following rules (called the right and left elimination rules, respectively):

$$\begin{array}{ll} u\ (u\backslash v) \to v & (\backslash_e) \\ (v/u)\ u \to v & (/_e) \end{array} \tag{2}$$

Formally, an AB-grammar is a function $\mathscr{L}$ which maps each word (a terminal symbol) a finite set of types (in a case of an ambiguous word, several but finite number of types may correspond to it).

Definition 1. We say that a string $w_1 \dots w_n$ is *derivable* in an AB-grammar, and its category is u if and only if for each w_i there exist t_i, $i = 1, 2 \dots n$ such that t_i is a category of w_i (i.e. $t_i \in \mathscr{L}(w_i)$) and $t_1 \dots t_n \to u$. The set of strings which are derivable by a given Ab-grammar is called the language produced by that AB-grammar.

CAT	Description	Expressions
S	Sentences	John walks, A woman smiles, etc.
IV	Intransitive Verbs	walk, sleep, smile, etc.
CN	Common Nouns	woman, book, tie, man, etc.
T = S/IV	Noun Phrases	a woman, John, every man, he_0, he_1, etc.
TV = IV/T	Transitive Verbs	love, see, meet, etc.

Table 1: Categories and their use in syntax and semantics

3 Glance at Montague Grammar

In order to provide systematic translations of syntactic expression into logical ones, Montague [21] assumes the following:

- Every lexical entry in the grammar has a syntactic category that has a corresponding semantic category.
- Compositionality Principle: The meaning of a compound expression is a function of the meanings of its parts and of the syntactic rule that combines these parts.

3.1 Syntax

Montague [21] employs an AB-calculus at the level of syntax. He defines rules that enable one to produce new categories out of the given ones as follows:[1]

Definition 2. CAT, the set of categories, is the smallest set such that:

- *S, IV, CN* belong to CAT.
- If A and B are elements CAT, then A/B belongs to CAT as well.[2]

S, stands for sentences, IV for intransitive verbs, and CN for common nouns. Table 1 describes how Montague defines other categories in terms of these three categories.

The syntactic derivation process in Montague's approach is the same as in AB-calculus. B_A denotes the set of lexical items of category A. For instance, B_{T} contains *a man*, *all dogs*, together with he_i for every natural number i.

[1] We do not provide the exact version of Montague's original grammar, but follow a more recent one, namely, Dowty [11].

[2] Montague [21] introduces another syntactic type, denoted as A//B, that has the same semantic meaning (i.e. translation) as A/B, but they model different syntactic categories. That is why, we will only use A/B categories as it is done, for instance, by Gamut [13].

Montague [21] proposes seventeen syntactic rules, which are classified as follows: basic rules, rules of functional application, rules of conjunction and disjunction, rules of quantification, rules of tense and sign. Each of these syntactic rules has the corresponding semantic rule. Let us consider some of Montague's syntactic rules and in the next section we discuss their semantic translations.

S1 is the first rule in MG, which is one of basic rules:

S1 B_A being the set of lexical elements of category A is contained in the set of all expressions of category A denoted as P_A, that is $B_A \subset P_A$

The another basic rule is the following:

S2 If $\zeta \in P_{\mathrm{CN}}$, then $F_0(\zeta), F_1(\zeta), F_2(\zeta) \in P_{\mathrm{T}}$ where:

- $F_0(\zeta) = \textit{every}\ \zeta$
- $F_1(\zeta) = \textit{the}\ \zeta$
- $F_2(\zeta)$ is $a\ \zeta$ or $an\ \zeta$ according as the first word in ζ takes a or an

Some of rules of functional application are as follows:

S4 If $\alpha \in P_{\mathrm{S/IV}}$ and $\delta \in P_{\mathrm{IV}}$, then $F_4(\alpha, \delta) \in P_{\mathrm{S}}$, where $F_4(\alpha, \delta) = \alpha\delta'$ and δ' is the result of replacing the first *verb* in δ by its third person singular present

The following rules are for quantification:

S14 If $\alpha \in P_{\mathrm{T}}$ and $\phi \in P_{\mathrm{S}}$, then $F_{10,n}(\alpha, \phi) \in P_{\mathrm{S}}$, where either:

1. α does not have the form he_k, and $F_{10,n}(\alpha, \phi)$ comes from ϕ by replacing the first occurrence of he_n or him_n by α and all other occurrences of he_n or him_n by $\left\{ \begin{array}{l} he \\ she \\ she \end{array} \right\}$ or $\left\{ \begin{array}{l} him \\ her \\ it \end{array} \right\}$ respectively, according as the gender of the first B_{CN} or B_{T} in α is $\left\{ \begin{array}{l} \text{masc.} \\ \text{fem.} \\ \text{neuter} \end{array} \right\}$, or
2. $\alpha = he_k$, and $F_{10,n}(\alpha, \phi)$ comes from ϕ by replacing all occurrences of he_n or him_n by he_k or him_k respectively

S15 If $\alpha \in P_{\mathrm{T}}$ and $\zeta \in P_{\mathrm{T}}$, then $F_{10,n}(\alpha, \zeta) \in P_{\mathrm{CN}}$

(1) Every man loves Mary.

Fig. 1 shows an exemplifying derivation of a sentence (1) using MG. A node of the tree in Fig. 1 shows the rule that is applied to the expressions from its daughter nodes in order to obtain the expression standing as the current node. If one views the derivation from a bottom up perspective, then the first rule used is **S5**; as a result, one produces out of *loves* of category IV/T and *Mary* of category IV/T the expression *loves Mary* of category IV. Using the rule **S4** leads to he_0 *loves Mary* of category P_{S}. On the other hand, by using the rule **S2**, one produces *every man*. We can then employ the rule **S14** in order to produce the expression $F_{10,0}(\textit{every man}, he_0 \textit{ loves Mary})$ of category S.

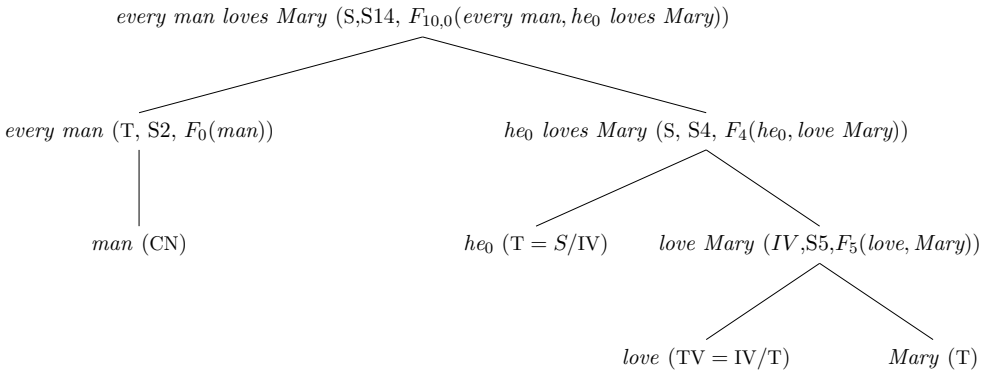


Figure 1: MG Derivation of *Every man loves Mary*

3.2 Translating Syntax to Semantics

Montague [21] shows how to translate syntactic expressions into the corresponding semantic expressions by translating his rules governing syntax into the corresponding semantic ones.[3]

S category for sentences translates to t. Also note that the syntactic categories IV and CN, corresponding to intransitive verbs and common nouns respectively, translate to the same semantic type, $e \to t$.

Montague [21] translates the other syntactic types using the following rule:

$$f \textit{ is a translation function of types such that } f(A/B) = f(B) \to f(A) \tag{3}$$

[3]While Montague's original translation is an *intensional* one, we only present extensional translations as it is sufficient for our current purposes.

Using the rule (3), one can derive Montague's translation[4] of the categories of noun phrases and transitive verbs as follows:

$$\begin{array}{rl} \textit{Noun phrases}: & f(\mathrm{T}) = f(\mathrm{S/IV}) = (e \to t) \to t \\ \textit{Transitive verbs}: & f(\mathrm{TV}) = f(\mathrm{IV/T}) = f(\mathrm{T}) \to f(\mathrm{IV}) = \\ & = f(\mathrm{S/IV}) \to f(\mathrm{IV}) = ((e \to t) \to t) \to e \to t \end{array} \tag{4}$$

In the MG semantic alphabet, one has individual constants and variables of type e, such as **j** for John, **m** for Mary etc.

To encode semantic representations of intransitive verbs, Montague introduces constants such as **sleep** of type $e \to t$, **cry** of type $e \to t$, etc. In words, these constants are one-place predicates whose argument is of type e. The same is true for common nouns (e.g. *woman*, *book*) and adjectives (e.g. *smart*, *interesting*, *tall*), in MG they are also treated as one-place predicates of type $e \to t$.

One of the insights in MG is Montague's [21] higher-order interpretations of noun phrases: all noun-phrases are of type $(e \to t) \to t$. For instance, (a) *John* is translated as $\lambda P.P\, j$, where j is of type e and P is of type $e \to t$; (b) A noun phrase such as *every man* is translated as $\lambda P.\forall x.\mathbf{man}\, x \supset P\, x$. The syntactic variables he_n, for $n = 0, 1, \ldots$, translate to $\lambda P.P\, x_n$, where x_n is a variable of type e. (c) A noun phrase, which is built with the help of an indefinite article, e.g. *a woman* is translated as $\lambda P.\exists x.\mathbf{woman}\, x \wedge P\, x$.

Consider the translation of transitive verbs in MG on the example of the verb *loves*. MG translates *loves* as follows:[5]

$$\lambda T.\lambda x.T(\lambda y.\mathbf{love}\, x\, y) : ((e \to t) \to t) \to e \to t \tag{5}$$

In order to illustrate a way of using the semantic translations of the syntactic rules, let us compute the meaning of the sentence *Every man loves Mary* (1), whose MG derivation is depicted in Fig. 1. For that, we consider Montague's [21] translation of the syntactic rules **S2**, **S4**, **S5**, and **S14**, which we encounter in the derivation of the sentence (1). We read the derivation tree in Fig. 1 from the left to right and bottom to top perspective.

We have already discussed the semantic translation of *every man*, it remains to see how to translate the syntactic rules **S4**, **S5**, and **S14**.

[4]By convention, type constructors such as $\to$ associate to the right, that is $\alpha \to \beta \to \gamma$ means $\alpha \to (\beta \to \gamma)$. In Montague's notations $\alpha \to \beta$ is $\langle \alpha, \beta \rangle$.

[5]One may observe an asymmetry between the object and subject encodings in the translation (5): the variable T standing for the object of *loves* is of a higher order type,$(e \to t) \to t$, whilst the variable modeling a subject, x, is of type e. The asymmetry in the formula $\lambda T \lambda x.T(\lambda y.\mathbf{love}\, x\, y)$ is due to the syntactic analyzes that Montague makes use of and not because of his semantic approach.

As both **S4** and **S5** are rules for *application*, their semantic translations are very similar.

T4 If $\alpha \in P_{\mathrm{T}}$ (i.e. $\alpha \in P_{\mathrm{S/IV}}$) and $\beta \in P_{\mathrm{IV}}$, and their translations are $(f\,\alpha)$ and $(f\,\beta)$ then $F_4(\alpha, \beta)$ is translated as $(f\,\alpha)\,(f\,\beta)$;

T5 If $\alpha \in P_{\mathrm{IV/T}}$ and $\beta \in P_{\mathrm{T}}$, and their translations are $f(\alpha)$ and $f(\beta)$ then $F_5(\alpha, \beta)$ is translated as $f(\alpha)(f(\beta))$.

In words, the *application of a functor to its argument* at the syntax level translates into *functional application* (of lambda calculus)[6] of their corresponding terms at the semantic level.

Hence, one can translate he_0 *loves Mary*. By following the derivation shown in Fig. 1, first we translate *loves Mary*; then, the translation of he_0 will apply to the translation of *loves Mary*. Thus, we translate *loves Mary* as follows:

$$f(\textit{loves Mary}) = (\lambda T.\lambda x.T(\lambda y.\mathbf{love}\,x\,y))\,(\lambda P.P\,\mathbf{m}) \twoheadrightarrow_\beta \lambda x.\mathbf{love}\,x\,\mathbf{m} \quad (6)$$

Now, we apply $\lambda P.P\,x_n$ to $\lambda x.\mathbf{love}\,x\,\mathbf{m}$:

$$\lambda P.P\,x_n(\lambda x.\mathbf{love}\,x\,\mathbf{m}) \to_\beta (\lambda x.\mathbf{love}\,x\,\mathbf{m})\,x_n \;\to_\beta\; \mathbf{love}\,x_n\,\mathbf{m} \quad (7)$$

Let us look up the semantic counterpart of **S14**. It is as follows:

T14 If $\alpha \in P_{\mathrm{T}}$ and $\phi \in P_{\mathrm{S}}$, and their translations are $f(\alpha)$ and $f(\phi)$ then $F_{10,n}(\alpha, \phi) \in P_{\mathrm{S}}$ is translated as $f(\alpha)(\lambda x_n f(\phi))$

Note that in **T14** application $f(\alpha)$ to $(\lambda x_n f(\phi))$ is in concordance with the fact that $f(\alpha)$ is of type $(e \to t) \to t$, which is why it can only take an argument of type $e \to t$; however, since $f(\phi)$ is of type t (as type of ϕ is P_{S}, which translates to t), $f(\phi)$ cannot be an argument of $f(\alpha)$, but $\lambda x_n f(\phi)$ can since its type is $e \to t$ (as x_n is of type e).[7]

Thus, we translate the results of the last step of the derivation given in Fig. 1. The syntactic expression $F_{10,0}(\textit{every man}, he_0\ \textit{loves Mary})$ is translated as:

[6] We refer readers to [3] for more details about lambda calculus and its variants. We may write either $f(x)$ and $(f\ x)$ for denoting application of f to x.

[7] According to S14, $F_{10,n}(\alpha, \phi)$ is the result of substituting by α the first occurrence of he_n in ϕ, whereas the other occurrences of he_n in ϕ are substituted by he/she/it or him/her/it (depending on the gender of α). From a semantic point of view, it means that all the occurrences of he_n should be substituted by α, because those occurrences of he_n that are substituted by he/she/it or him/her/it are antecedents of the first occurrence of he_n, which is substituted by α.

$\lambda P.\forall x.\mathbf{man}(x) \supset P\,x$ applied to $\lambda x_0.\mathbf{love}\,x_0\,\mathbf{m}$, which is computed as follows:

$$(\lambda P.\forall x.\mathbf{man}\,x \supset P\,x)(\lambda x_0.\mathbf{love}\,x_0\,\mathbf{m}) \twoheadrightarrow_\beta \forall x.\mathbf{man}\,x \supset \mathbf{love}\,x\,\mathbf{m} \quad (8)$$

If we had *a woman* instead of *Mary*, the translation would be as follows:

$$\begin{aligned}&(\lambda P.\forall x.\mathbf{man}\,x \supset P\,x)(\lambda x_0.\exists y.\mathbf{woman}\,y \wedge \mathbf{love}\,x_0\,y)\\ \twoheadrightarrow_\beta\quad &\forall x.\mathbf{man}\,x \supset (\exists y.\mathbf{woman}\,y \wedge \mathbf{love}\,x\,y)\end{aligned} \quad (9)$$

The formula (9) in words means that *for every man there is a woman whom he loves*. On the other hand, the initial sentence could also mean that *there is a woman whom every man loves*. However, a formula that would give rise to the second interpretation cannot be obtained using the derivation we have used. In order to obtain the second reading, Montague [21] proposes a technique, known as *Montague's trick*.

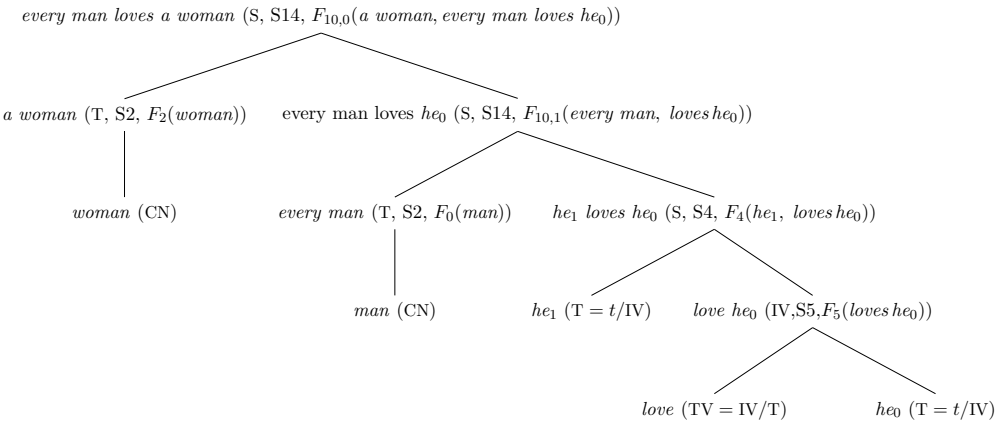


Figure 2: Montague's Trick: Deriving Second Reading

The idea behind of Montague's trick is to analyze a sentence in a way as it is shown in Fig. 2: The desired reading is obtained as a result of *reversing scopes*, i.e., by making *a woman* to scope over *every man loves somebody*.

We compute the formula corresponding to the final step of the derivation in Fig. 2 as follows:

$$\begin{aligned}&(\lambda Q.\exists y.\mathbf{woman}\,y \wedge Q\,y)(\lambda x_0.\lambda P.\forall x.\mathbf{man}\,x \supset \mathbf{love}\,x\,x_0) \twoheadrightarrow_\beta\\ &\twoheadrightarrow_\beta \exists y.\mathbf{woman}\,y \wedge \forall x.\mathbf{man}\,x \supset \mathbf{love}\,x\,y\end{aligned} \quad (10)$$

4 Abstract Categorial Grammars

Abstract Categorial Grammars (ACGs) by de Groote [9] are grammatical framework based on Curry's idea of having two levels of grammar [6],

pheno and tecto. At the pheno-grammatical level, natural language expressions are produced, whereas the tecto-grammatical one is responsible for production of those expressions with the help of rules that do not involve any information about surface forms. Another source of inspiration of ACGs comes from MG, namely the use of a typed lambda calculus and a compositional interpretation of syntax into semantics.

The following preliminary notions help to define Abstract Categorial Grammars (ACGs).

- A higher-order linear signature (HOS) is a triple $\Sigma = \langle A, C, \tau \rangle$ where:
  - A is a finite set of atomic types;
  - C is a finite set of constants;
  - $\tau : C \longrightarrow \mathcal{T}(A)$ is type assignment function mapping each constant from C to a linear implicative type built upon A.
- The order of a type ξ, denoted as $ord(\xi)$ is defined as:

$$ord(\xi) = \begin{cases} 1 \text{ if } \xi \text{ is atomic.} \\ max(1 + ord(\alpha), ord(\beta)) \text{ if } \xi = \alpha \to \beta \end{cases}$$

- Linear λ-terms $\Lambda^l(\Sigma)$over a HOS $\Sigma = \langle A, C, \tau \rangle$ is the set containing all and only elements defined as follows:
  - If $t_1, t_2 \in \Lambda^l(\Sigma)$ the $(t_1 t_2) \in \Lambda^l(\Sigma)$.
  - If $t \in \Lambda^l(\Sigma)$, then $\lambda x.t \in \Lambda^l(\Sigma)$, where x is a variable.
  - For any subterm $\lambda x.p$ of a term $t \in \Lambda^l(\Sigma)$, x is free in p.
  - for any subterm $t_1 t_2$ of $t \in \Lambda^l(\Sigma)$, t_1 and t_2 have no common free variable.

Definition 3. An ACG is a quadruple $G = (\Sigma_a, \Sigma_o, \mathcal{L}, S)$ where:

- $\Sigma_a = \langle A_a, C_a, \tau_a \rangle$ is a higher order signature, called the abstract signature;
- $\Sigma_o = \langle A_o, C_o, \tau_o \rangle$ is a higher order signature, called the object signature;
- $\mathcal{L}$ is a mapping from C_a to $\Lambda^l(\Sigma_o)$, called the lexicon of the grammar G, which is uniquely lifted to a homomorphism from $\Lambda^l(\Sigma_a)$ to $\Lambda^l(\Sigma_o)$ (which we denote again with $\mathcal{L}$) that has the following properties:
  - $\mathcal{L}(x) = x$ where x is a variable;

- $\mathcal{L}(t_1 t_2) = \mathcal{L}(t_1)\mathcal{L}(t_2)$;
- $\mathcal{L}(\lambda x.t) = \lambda x.\mathcal{L}(t)$.

- S is a type of Σ_a, called the distinguished type of G.

With the ACG $G = (\Sigma_a, \Sigma_o, \mathcal{L}, S)$, we associate two languages, defined as follows:

The *abstract language*: $\mathcal{A}(G) = \{u \in \Lambda(\Sigma_a) \mid \vdash_{\Sigma_o} u : s \quad \text{is derivable}\}$
The *object language*: $\mathcal{O}(G) = \{v \in \Lambda(\Sigma_o) \mid \exists u \in \mathcal{A}(G) : v = \mathcal{L}(u)\}$

In words, the object language is the image of the abstract language by the lexicon.

We call an ACG $G = (\Sigma_a, \Sigma_o, \mathcal{L}, S)$ of order n (or n-th order) if n is the maximum of orders of the types of the constants in the abstract signature Σ_a.

5 Montague Grammar as an ACG

We can build a HOS, Σ_1, to model the syntax used by Montague. More specifically, the terms over Σ_1 model the derivation trees. The derivation shown in Fig. 3 is modeled by the term u defined as follows:

$$u = \; C_{every}\, C_{man}\, (C_{loves}\, (C_a\, C_{woman})) : \mathrm{S}$$

We can view the term u as an instantiation of the Montague's rule **S14**, which allows us to produce the expression $F_{10,0}(\textit{every man}, \textit{he}_0 \textit{ loves Mary})$ of category S.

To be able to build (logical) semantic formulas, we construct another HOS, Σ_2, whose constants are shown in Fig. 4.

C_{every}, C_a, C_{some}	$: \mathrm{CN} \to \mathrm{IV} \to \mathrm{S}$
C_{mary}, C_{john}	$: \mathrm{IV} \to \mathrm{S}$
C_{walks}, C_{sleeps}	$: \mathrm{IV}$
C_{woman}, C_{man}	$: \mathrm{CN}$
C_{loves}, C_{meets}	$: (\mathrm{IV} \to \mathrm{S}) \to \mathrm{IV}$

Figure 3: Σ_1: Derivation Trees

$\mathbf{run}, \mathbf{woman}, \mathbf{man}$	$: e \to t$
$\mathbf{love}, \mathbf{meet}, \mathbf{read}$	$: e \to e \to t$
$\wedge, \supset$	$: t \to t \to t$
$\forall, \exists$	$: (e \to t) \to t$

Figure 4: Σ_2: Logical Alphabet

By mapping the term u under the lexicon f, we get:

$$\begin{aligned} f(u) = \; & f(C_{every})\, f(C_{man})\, (f(C_{loves})\, (f(C_a)\, f(C_{woman}))) \twoheadrightarrow_\beta \\ & \forall x.(\mathbf{man}\, x) \supset \exists y.(\mathbf{woman}\, y) \wedge (\mathbf{love}\, x\, y) \end{aligned} \tag{11}$$

$$\begin{array}{lll} \text{S} & := t & C_{every} := \lambda P.\lambda Q.\forall x.Px \supset Qx \\ \text{IV} & := e \to t & C_a, C_{some} := \lambda P.\lambda Q.\forall x.Px \wedge Qx \\ \text{CN} & := e \to t & C_{walks} := \mathbf{walk} \\ & & C_{woman} := \mathbf{woman} \\ & & C_{loves} := \lambda T.\lambda x.T(\lambda y.\mathbf{love}\, x\, y) \end{array}$$

Figure 5: f: mapping From Derivation Trees to Logical Formulas

Montague's trick Note that the current state of the grammar does not allow to obtain the second reading of the sentence (1). We can introduce new constants to build a term that could be mapped to the second reading. To mimic Montague's trick, we first combine the verb and the subject and the resultant term is then combined with the object. So, we extend the abstract signature with the constant $C^2_{loves} : (\text{IV} \to \text{S}) \to \text{IV}$ whose semantic interpretation is the following one:

$$f(C^2_{loves}) = \lambda S.\lambda x.S(\lambda y.\mathbf{love}\, y\, x)$$

The term u_{trick} is defined as follows:

$$u_{trick} = C_a\, C_{woman}\, (C^2_{loves}\, (C_{every}\, C_{man})) : \text{S}$$

The semantic interpretation of u_{trick} is as follows:

$$\begin{aligned} f(u) = f(C_a)\, f(C_{woman})\, (f(C^2_{loves})\, (f(C_{every})\, f(C_{man}))) \twoheadrightarrow_\beta \\ \exists x.(\mathbf{woman}\, x) \wedge \forall y.(\mathbf{man}\, y) \supset (\mathbf{love}\, y\, x) \end{aligned} \tag{12}$$

6 Montague Grammar in Natural Language Processing

Parsing for ACGs of order three is an NP-complete problem, as showed by Salvati [29, 28]. The grammar based on Montague's syntax, which we proposed in the previous section, is of third order.

For second order ACGs, parsing is of polynomial complexity as shown by Salvati [27] and Kanazawa [15]. Second order ACGs can encode a number of formalisms, including CFGs, Tree-Adjoining Grammars (TAGs) by Joshi [14] and Linear Context-Free Rewriting Systems by Weir [33].

6.1 The Syntax-Semantics interface: TAG for Syntax

TAG were found to be useful for modeling natural language phenomena that CFGs cannot, like certain kind of long-distance dependencies [32]. At

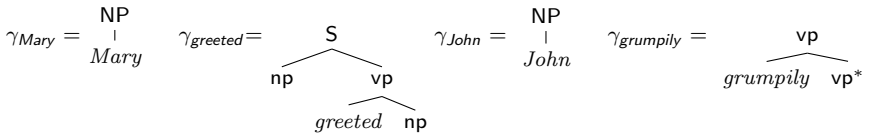
the same time, for TAGs, as it for CFGs, polynomial parsing algorithms are available. Subsequently TAGs found a number of applications across natural language processing problems.

6.2 Tree Adjoining Grammars

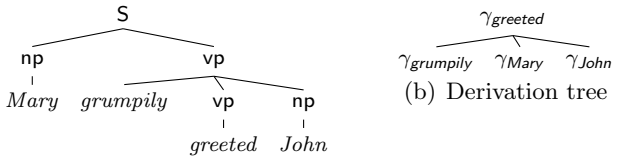
TAG is a tree generating formalism. The TAG tree language is generated by combining *elementary* trees. There are two kinds of elementary trees, *initial* and *auxiliary* ones. There are two ways of combining elementary trees, by *substitution* and by *adjunction*.

Substitution is a replacement of a frontier node of a tree with an initial tree that has the same root label as the given node.

Adjunction is like substitution, but in this case one can also substitute an internal node (i.e. a node which has children) of a tree with an auxiliary tree whose root node has the same label as the substituted node does. Since an internal node, call it n, of a tree (call this tree γ) has children (by definition), those children would be left orphan as a result of adjoining an auxiliary tree (call it β) on n. As a consequence, the tree structure would be lost. To avoid that, a TAG auxiliary tree β has a frontier node, marked with $*$, which has the same label as the root of β (and thus the same label as n). This frontier node, called the *foot* node of β, becomes mother to the children of the node n. For example, $\gamma_{greeted}$, γ_{John} and γ_{Mary} are initial trees, whereas $\gamma_{grumpily}$ is an auxiliary tree. Substituting γ_{John} and γ_{Mary} into $\gamma_{greeted}$ on the frontier labeled with np and adjoining $\gamma_{grumpily}$ into $\gamma_{greeted}$ on the node with label vp produces the derived tree shown in Figure 6(a).



The process of the production of the derived tree 6(a) is recorded by the corresponding *derivation* tree, which is represented as the tree 6(b).



(a) Derived tree (b) Derivation tree

Figure 6: *Mary grumpily greeted John*

6.3 TAG with Semantics as ACGs

While TAGs proved to be successful for encoding a number of syntactic phenomena, it was not clear how one could design a compositional semantic approach with TAGs when there is a mismatch between the derivational scope (parent-child relations in a derivation tree) and the semantic (logical) one.

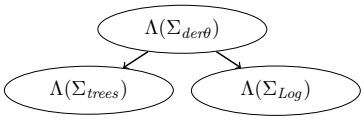


Figure 7: ACG architecture for TAG

In the ACG encoding of TAG of de Groote [10], TAG derivation trees are represented as the abstract language, whereas TAG derived trees as the object one. Pogodalla [25, 24] provided an encoding of TAG with Montagovian semantics. In Pogodalla's approach, TAG derivation trees are realized as an abstract language (like it is in [10]), whereas logical formulas are modeled as another object language. This architecture is pictorially represented in Fig. 7: there are the following signatures and lexicons involved: a signature $\Sigma_{der\theta}$, where we model TAG derivation trees; a signature Σ_{trees} where TAG derived trees are encoded; $\mathscr{L}_{\textit{d-ed trees}} : \Sigma_{der\theta} \longrightarrow \Sigma_{trees}$, it maps derivation trees to derived trees; Σ_{Log} where we define HOL terms encoding Montague semantics; $\mathscr{L}_{Log} : \Sigma_{der\theta} \longrightarrow \Sigma_{Log}$ maps derivation trees to Montague semantics.

$\Sigma_{der\theta}$: Its atomic types include S, vp, np, S_A, $\mathsf{vp}_A \ldots$ where the X types stand for the categories X of the nodes where a substitution can occur while the X_A types stand for the categories X of the nodes where an adjunction can occur. For each elementary tree $\gamma_{\textit{lex. entry}}$ it contains a constant $\mathrm{C}_{\textsf{lex. entry}}$ whose type is based on the adjunction and substitution (see Table 2). It additionally contains constants $I_X : X_A$ that are meant to provide a fake auxiliary tree on adjunction sites labeled with X where no adjunction actually takes place in a TAG derivation.

Σ_{trees}: Its unique atomic type is τ the type of trees. For any X of arity n belonging to the ranked alphabet describing the elementary trees of the TAG, Σ_{trees} has a constant $X_n : \overbrace{\tau \multimap \cdots \multimap \tau}^{n \text{ times}} \multimap \tau$

$\mathscr{L}_{\textit{d-ed trees}}(X_A) = \tau \multimap \tau$ and for any other type X, $\mathscr{L}_{\textit{d-ed trees}}(X_A) = \tau$. Table 2 illustrates the way $\mathscr{L}_{\textit{d-ed trees}}$ interprets constants of $\Sigma_{der\theta}$.

Constants of Σ_{Log} are shown in Table 3. We have two atomic types in Σ_{Log}, e for entities and t for propositions.

The lexicon $\mathscr{L}_{Log} : \Sigma_{der\theta} \longrightarrow \Sigma_{Log}$ is given in Table 4.

(2) Mary grumpily greeted John.

The term v models the TAG derivation tree on Fig. 6. By mapping v with $\mathscr{L}_{\textit{d-ed trees}}$, one obtains the term representation of the derived tree for (2); and by mapping v with $\mathscr{L}_{Log}$, one obtains Montague style HOL semantics of (2).

$$
\begin{aligned}
v &= \mathrm{C}_{\mathsf{greeted}}\ \mathrm{I}_{\mathsf{S}}\ (\mathrm{C}^{\mathsf{V}}_{\mathsf{grumpily}}\ \mathrm{I}_{\mathsf{S}})\ \mathrm{C}_{\mathsf{Mary}}\ \mathrm{C}_{\mathsf{John}} \\
\mathscr{L}_{\textit{d-ed trees}}(v) &= \mathsf{S}_2\ (\mathsf{np}_1\ \textit{Mary})\ (\mathsf{v}_2\ (\mathsf{v}_2\ \textit{grumpily}\ (\mathsf{v}_1\ \textit{greeted}))\ (\mathsf{np}_1\ \textit{John})) \\
\mathscr{L}_{Log}(v) &= \mathbf{grumpily}\,(\mathbf{greet}\,\mathbf{m}\,\mathbf{j})
\end{aligned}
$$

Abstract constants $\Sigma_{der\theta}$	Their images by $\mathscr{L}_{\textit{d-ed trees}}$	The corresponding TAG trees
$\mathrm{C}_{\mathsf{John}} : \mathsf{np}$	$c_{\textit{John}} \begin{array}{l} :\tau \\ = \mathsf{np}_1\ \textit{John} \end{array}$	$\gamma_{\textit{John}} =$ NP | *John*
$\mathrm{C}^{\mathsf{V}}_{\mathsf{grumpily}} : \mathsf{vp}_A \multimap \mathsf{vp}_A$	$c^{\mathsf{vp}}_{\textit{grumpily}} \begin{array}{l} :(\tau \multimap \tau) \multimap (\tau \multimap \tau) \\ = \lambda^{\mathrm{o}} adv_v\, x.adv_v\ (\mathsf{vp}_2\ \textit{grumpily}\ x) \end{array}$	$\gamma_{\textit{grumpily}} =$ vp: *grumpily* vp*
$\mathrm{C}_{\mathsf{greeted}} : \begin{array}{l} \mathsf{S}_A \multimap \mathsf{v}_A \multimap \\ \multimap \mathsf{np} \multimap \mathsf{np} \multimap \mathsf{S} \end{array}$	$c_{\textit{greeted}} = \begin{array}{l} : \begin{array}{l} (\tau \multimap \tau) \multimap (\tau \multimap \tau) \\ \multimap \tau \multimap \tau \multimap \tau \end{array} \\ \lambda^{\mathrm{o}} adv_s\, adv_v\, subj\, obj.\ adv_s \\ (\mathsf{S}_2\ subj\ (adv_v\ (\mathsf{vp}_2\ greeted\ obj))) \end{array}$	$\gamma_{\textit{greeted}} =$ S: np, vp: *greeted*, np
$\mathrm{I}_X : X_A$	$\lambda x.x : \tau \multimap \tau$	

Table 2: TAG with Semantics as ACGs: $\mathscr{L}_{\textit{d-ed trees}}$ lexicon

$\mathbf{j}, \mathbf{m}$	$: e$	**because**	$: t \to t \to t$
woman, **smart**, **work-out**	$: e \to t$	**greet**, **love**	$: e \to e \to t$
grumpily	$: t \to t$	**fast**	$: (e \to t) \to e \to t$
$\wedge$	$: t \to t \to t$	$\vee$	$: t \to t \to t$
$\Rightarrow$	$: t \to t \to t$	$\neg$	$: t \to t$
$\exists$	$: (e \to t) \to t$	$\forall$	$: (e \to t) \to t$

Table 3: Constants in the semantic vocabulary Σ_{Log}

6.4 Discourse Grammars as ACGs

Several authors have proposed grammatical approaches to discourse based on TAGs [12, 7]. Each of these grammars consist of two levels, a discourse-level one and a sentence (clause) level one. An ACG approach to discourse facilitated to overcome some problems that TAG based grammars experience when modeling an important discourse phenomenon of *clause-medial adverbials*. Consider the following discourse:

(3) Mary worked out. She *then* watched TV.

In (3), *then* is a discourse adverbial. At the discourse level, it has two places: two discourse units which it connects rhetorically (with a temporal relation of succession), which are *Mary worked out* and *Mary watched TV*. But at the sentence level, it operates on the verb phrase *watched TV*, which is its only argument. Thus, there is a mismatch between the discourse level and sentence level descriptions of discourse adverbials.

To distinguish a discourse meaning and a surface structure, we build two ACGs, one for studying structural properties of discourse and the other one to study discourse meaning. Let us sketch the constraint we are modeling: in discourse meaning, *then* is a two-place predicate whose arguments are discourse units, but in syntax, it is a verb-phrase modifier.

Let us first model the case when then is fronted: $d_{then} : \mathsf{DU} \rightarrow \mathsf{DU} \rightarrow \mathsf{DU}$, which encodes that a discourse connective takes two arguments that are pieces of discourses (in our modeling, they are terms of type DU); the resulting term also models a discourse (its type is again DU). We can interpret the constant $d_{then} : \mathsf{DU} \rightarrow \mathsf{DU} \rightarrow \mathsf{DU}$ into TAG derived trees as follows:

$$d_{then} := \lambda s_1.\, \lambda\, s_2.\mathsf{S}_3\ s_1\ \mathbf{dot}\ (\mathsf{S}_2\ (\mathit{Then},\ s_2)$$

In the case of a clause-medial connective *then*, we introduce a constant d^m_{then} typed as $d^m_{then} : \mathsf{DU} \rightarrow (\mathsf{S}_\mathsf{A} \rightarrow \mathsf{DU}) \rightarrow \mathsf{DU}$. It indeed models the fact that the one of its arguments needs an adjunction in order to become a discourse unit. Then we can easily interpret it as TAG derived trees. However note that while theoretically it is perfectly possible to have d^m_{then} defined as it is now, it makes the ACG we are building of order three (and thus polynomial parsing cannot be guaranteed).

As it was shown by Danlos et al. [8], it is possible to build a second order ACG that would be still able to express the needed constraint. Let us therefore introduce a new type DU_m, which we use in order to model clause-medial connectives. In this new approach, the type of d^m_{then} is $\mathsf{DU} \rightarrow$

Constants of $\Sigma_{der\theta}$	Their interpretations by $\mathscr{L}_{Log}$
$\mathrm{C}_{\mathsf{woman}} : \mathsf{n}_\mathsf{A} \multimap \mathsf{np}$	$\lambda D.\lambda q\,.D\ \mathbf{woman}\ q$
$\mathrm{C}_{\mathsf{Mary}} : \mathsf{np}$	$\lambda D.D\ \mathbf{m}$
$\mathrm{C}_{\mathsf{grumpily}} : \mathsf{V}_\mathsf{A} \multimap \mathsf{V}_\mathsf{A}$	$\lambda a_v.\lambda r.\ a_v\ (\lambda x.\mathbf{grumpily}(r\,x))$
$\mathrm{C}_{\mathsf{greeted}} : \mathsf{S}_\mathsf{A} \multimap \mathsf{V}_\mathsf{A} \multimap \mathsf{np} \multimap \mathsf{np} \multimap \mathsf{S}$	$\lambda\, s\, a\, S\, O.s(S(a(\lambda x.O(\lambda y.(\mathbf{greet}\ x\ y)))))$
$\mathrm{C}_{\mathsf{every}}, \mathrm{C}_{\mathsf{each}} : \mathsf{n}_\mathsf{A}$	$\lambda\, P.\, \lambda Q\,.\, \forall\, x.\, (P\, x)\ \supset\ (Q\, x)$
$\mathrm{C}_{\mathsf{a}}, \mathrm{C}_{\mathsf{some}} : \mathsf{n}_\mathsf{A}$	$\lambda\, P.\, \lambda Q\,.\, \exists\, x.\, (P\, x)\ \wedge\ (Q\, x)$

Table 4: Interpretations by $\mathscr{L}_{Log}$

$\mathsf{DU}_m \rightarrow \mathsf{DU}$. In order to interpret it in TAG derived trees, we first interpret it in TAG derivation trees. The constraint that the constant $d^m_{then} : \mathsf{DU} \rightarrow \mathsf{DU}_m \rightarrow \mathsf{DU}$ models can be expressed in TAG derivation trees as follows:

$$d^m_{then} := \lambda s_1.\lambda s_2.\ C_+\ s_1\ (s_2\ C^{\mathsf{vp}}_{then})$$

Indeed, we expressed that the second argument of d^m_{then} must receive an adjunction that would place *then* in its VP (a clause-medial position). (The constant C_+ stands for a concatenation: it can be interpreted in TAG derived trees as an initial tree with two substitution sites separated by a dot.) It means that we have to allow adjunction on the second argument; the first argument, however, does not need it. To achieve that, we interpret types DU and DU_m as follows:

$$\begin{array}{ll} \mathsf{DU} & := \mathsf{S} \\ \mathsf{DU}_m & := \mathsf{vp}_A \rightarrow \mathsf{S} \end{array}$$

Since we have already built the lexicon interpreting TAG derivation trees into TAG derived trees, we can compose that lexicon with the above defined interpretations to obtain interpretation of discourse into TAG derived trees.

On the meaning side of the discourse, d^m_{then} and d_{then} behave in the same way: both model the *succession* rhetorical relation. We therefore interpret them as follows: $d^m_{then}, d_{then} := \textsc{succession} : t \rightarrow t \rightarrow t$. As one can see, we interpret types DU and DU_m, both as t.

7 Discussion and Summary

There are a number of insights that Montague's works offer to the studies in semantics and in particular to the studies of the syntax-semantics interface, such as, for instance, use of typed lamda calculus and higher-order interpretations of noun phrases. Another important point in MG is its uniform treatment of intransitive verbs, common nouns and adjectives. As we already mentioned above, all of them are treated as one-place predicates of type $e \rightarrow t$. While this approach to intransitive verbs, common nouns and adjectives has found a number of uses, there are other ways of their modeling which were offered within other theories. Ranta [26] and Luo [17, 18] propose grammars within frameworks of dependent types. In their grammars, common nouns are not treated as predicates but types. That is, *human*, *book*, *tree* etc. are types. The verbs are predicates. For instance, *talk* is of type *human* $\rightarrow t$. This makes sure that one would not have *a tree talks*, because *a tree* is not of type *human*. Pkhakadze [23] argues in favor of distinguishing between nouns, adjectives and verbs. In

his approach, verbs are predicates. A noun is either a set, or a constant ranging on the set defined by that noun.

Meaning in natural language claimed attention of various communities, including linguistics, mathematics (logics, computer science), philosophy, and psychology, and the syntax-semantics interface is only one way of studying it, treating syntax as a pivot to semantics. In the studies focused on the syntax-semantics interface though, the main question is still whether a syntactic analysis is sufficient to obtain a semantic one, or to put it another way, is syntax a pure pivot for semantics, or syntax is also shaped by semantics? After all, we, speakers, use natural language to create and convey meanings, which allows us to think and communicate by means of a language. In MG we already see an emergence of some semantics inspired syntactic rules. (Montague's trick shows this spirit as it offers a new syntactic analysis in order to obtain one of the possible readings.) A later developments of MG, such as ACGs, have been making use of semantics inspired syntax in order to model various phenomena in studying form-meaning relations.

References

1. Ajdukiewicz, K. Die syntaktische Konnexität. *Stud. Philos. 1* (1935), 1–27.

2. Bar-Hillel, Y. A quasi-arithmetical notation for syntactic description. *Language 29*, 1 (1 1953), 47–58.

3. Barendregt, H. P. Lambda calculi with types. In *Handbook of Logic in Computer Science (Vol. 2)*, S. Abramsky, D. M. Gabbay, and S. E. Maibaum, Eds. Oxford University Press, Inc., New York, NY, USA, 1992, pp. 117–309.

4. Bloomfield, L. *Language*. University of Chicago Press, Chicago, 1933.

5. Chomsky, N. On certain formal properties of grammars. *Information and Control 2*, 2 (1959), 137 – 167.

6. Curry, H. B. Some logical aspects of grammatical structure. *Journal of Symbolic Logic 25*, 4 (1960), 341–341.

7. Danlos, L. D-STAG : un formalisme d'analyse automatique de discours basé sur les TAG synchrones. *Revue TAL 50*, 1 (2009), 111–143.

8. Danlos, L., Maskharashvili, A., and Pogodalla, S. Interfacing sentential and discourse TAG-based grammars. In *Proceedings of the 12th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+12)* (Düsseldorf, Germany, June 2016), pp. 27–37.

9. de Groote, P. Towards abstract categorial grammars. In *Association for Computational Linguistics, 39th Annual Meeting and 10th Conference of the European Chapter* (Toulouse, France, July 2001), pp. 148–155. Colloque avec actes et comité de lecture. internationale.

10. de Groote, P. Tree-Adjoining Grammars as Abstract Categorial Grammars. In *Proceedings of the Sixth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+6)* (2002), Università di Venezia, pp. 145–150.

11. Dowty, D. R., Wall, R. E., and Peters, S. *Introduction to Montague Semantics*. Reidel, Dordrecht, 1981.

12. Forbes, K., Miltsakaki, E., Prasad, R., Sarkar, A., Joshi, A. K., and Webber, B. L. D-LTAG system: Discourse parsing with a Lexicalized Tree-Adjoining Grammar. *Journal of Logic, Language and Information 12*, 3 (2003), 261–279. Special Issue: Discourse and Information Structure.

13. Gamut, L. *Logic, Language, and Meaning: Intensional logic and logical grammar*. Logic, Language, and Meaning. University of Chicago Press, 1991.

14. Joshi, A. K., and Schabes, Y. Tree-adjoining grammars. In *Handbook of Formal Languages*, G. Rozenberg and A. Salomaa, Eds. Springer Berlin Heidelberg, 1997, pp. 69–123.

15. Kanazawa, M. Parsing and generation as datalog queries. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL)* (Prague, Czech Republic, June 2007), Association for Computational Linguistics, pp. 176–183.

16. Lambek, J. The mathematics of sentence structure. *Americal Mathematical Monthly 65* (1958), 154–170.

17. Luo, Z. Common nouns as types. In *Logical Aspects of Computational Linguistics* (Berlin, Heidelberg, 2012), D. Béchet and A. Dikovsky, Eds., Springer Berlin Heidelberg, pp. 173–185.

18. Luo, Z. Formal semantics in modern type theories with coercive subtyping. *Linguistics and Philosophy 35*, 6 (Nov 2012), 491–513.

19. Montague, R. English as a formal language. In *Linguaggi Nella Società e Nella Tecnica*, B. Visentini, Ed. Edizioni di Communita, 1970, pp. 188–221.

20. Montague, R. Universal grammar. *Theoria 36*, 3 (1970), 373–398.

21. Montague, R. The proper treatment of quantification in ordinary English. In *Formal Philosophy: Selected Papers of Richard Montague*, R. Thomason, Ed. Yale University Press, New Haven, CT, 1973, pp. 247–270.

22. Moot, R., and Retoré, C. *The Logic of Categorial Grammars: A Deductive Account of Natural Language Syntax and Semantics*. FoLLI-LNCS. Springer, July 2012.

23. Pkhakadze, K. About logical declination and lingual relations in georgian. *Georgian language and logic 1*, 1 (2005), 19–77.

24. Pogodalla, S. Advances in Abstract Categorial Grammars: Language Theory and Linguistic Modeling. ESSLLI 2009 Lecture Notes, Part II. ESSLLI 2009 Lecture Notes, 2009.

25. Pogodalla, S. A syntax-semantics interface for Tree-Adjoining Grammars through Abstract Categorial Grammars. *Journal of Language Modelling 5*, 3 (2017), 527–605.

26. Ranta, A. *Type-theoretical Grammar*. Indices (Claredon). Clarendon Press, 1994.

27. Salvati, S. *Problèmes de filtrage et problèmes d'analyse pour les grammaires catógorielles abstraites*. PhD thesis, Institut National Polytechnique de Lorraine, 2005.

28. Salvati, S. On the complexity of Abstract Categorial Grammars. In *10th conference on Mathematics of Language (MOL 10), Los Angeles, CA* (July 2007), M. Kracht, G. Penn, and E. Stabler, Eds.

29. Salvati, S. A note on the complexity of abstract categorial grammars. In *The Mathematics of Language* (Berlin, Heidelberg, 2010), C. Ebert, G. Jäger, and J. Michaelis, Eds., Springer Berlin Heidelberg, pp. 266–271.

30. Shaumyan, S. *Structural Linguistics*. Nauka, 1965.

31. Shaumyan, S. *A semiotic theory of language*. Advances in semiotics. Indiana University Press, 1987.

32. Shieber, S. M. Evidence against the context-freeness of natural language. *Linguistics and Philosophy 8*, 3 (1985), 333–343.

33. Weir, D. J. *Characterizing Mildly Context-Sensitive Grammar Formalisms*. PhD thesis, University of Pennsylvania, Philadelphia, PA, USA, 1988. Supervisor: Aravind K. Joshi.

SOLVING TEXTUAL ENTAILMENT WITH THE THEOREM PROVER FOR NATURAL LANGUAGE

Lasha Abzianidze

CLCG, University of Groningen, the Netherlands
L.Abzianidze@rug.nl

Abstract

We present a theorem prover for natural language and show how it processes various types of textual entailment problems. The prover itself is based on a tableau system for natural logic that employs logical forms similar to linguistic expressions. With respect to the problems drawn from textual entailment datasets, a wide-range of the judgments of the prover are discussed, including both correct and incorrect ones. The analysis shows that the false proofs, which are extremely rare, are mainly due to the wrong lexical senses or the noisy gold labels of the dataset. Knowledge sparsity is identified as the main reason for the failure in proof search.

Keywords and phrases: Analytic tableau method, theorem proving, natural reasoning, lambda logical form, natural logic, typed lambda calculus, textual entailment, combinatory categorial grammar

1 Introduction

The automatic detection of logical relations between natural language expressions is a fundamental problem of natural language understanding. In order to model the human reasoning over natural language text, textual entailment data is used for training and evaluating the theories and systems. The data represents a collection of text pairs annotated on the entailment, contradiction and neutral relations by humans. The annotated textual entailments are further used as a benchmark in the recognizing textual entailment (RTE) challenges [8]. On the other hand, studies on formal semantics seek formal logics that model linguistic semantics. Such formal logics supported by an automated theorem prover usually leads to an automated reasoner for natural language. A notable example of this research line is the RTE system NutCracker [6] which combines first-order logic and theorem proving to account for the natural reasoning. We follow the latter research line by adopting a version of higher-order logic as a semantic representation and employing an analytic tableau system for theorem proving. The combination results in an implemented theorem prover for natural

language that is applicable to wide-coverage text using a robust syntactic parser as a preprocessor.

While our previous works [2, 1, 3] talks about the theorem prover for natural language, it was never the purpose to show the matters of failure and success of the prover with respect to entailment problems. Taking into account that the prover has almost prefect precision ($\approx 98\%$) with competitive accuracy [1], it is interesting to see the false proofs or to find out what is the reason for relatively small recall (61%). In this paper we explore these cases by considering a plethora of entailment problems that get mixed judgments from the prover. Based on the analysis, we give the rationale for the decisions of the prover and draw the directions towards improvement of the performance.

The rest of the paper is structured as follows. First, the theory behind the prover, the natural tableau system [22], and its extensions are introduced. Then two relevant components of the natural logic theorem prover are described. The natural logic theorem prover is followed by the prover for natural language. We outline its architecture and the details of entailment processing. For each classification type we discuss several entailment problems that illustrate the issues of theorem proving. Based on the examples, the reasons for true, false and failed proofs are explained. The paper ends with final remarks and a description of future work.

2 Natural tableau system

An analytic tableau system for natural logic [22] is a proof procedure over the logical forms of natural language expressions. The logical forms, so-called Lambda Logical Forms (LLFs), are merely typed λ-terms that employ lexical constants and no logical connectives.[1] LLFs resemble the surface forms and can be considered as formulas of some sort of *natural logic* [15, 5, 26], hence we refer the tableau system as the *natural tableau*. On the other hand, a tableau method is a refutation proof procedure [10]. To prove a statement, it attempts to find a situation that serves as a counterexample to the statement. If such a situation is found, the statement is disproved; otherwise it is proved. The search for a situation is done by building a tableau, also called a truth tree. Each branch of a tableau represents a situation. The construction of a tableau is guided by a predefined set of tableau rules.

[1] In contrast to [22], throughout the paper we assume the extensional semantic types built upon the entity e and truth t types while the type s for possible worlds is dropped out. The terms for common lexical elements are typed in an ordinary way: nouns and intransitive verbs as et, transitive verbs as eet, proper names as e, quantifiers as $(et)(et)t$.

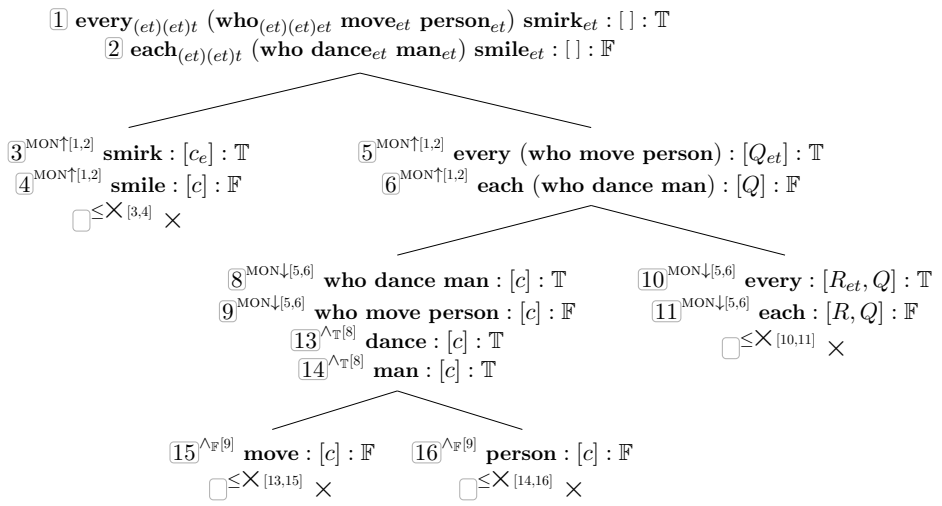


Figure 1. The closed tableau represents a failed attempt to refute that the TEP P:"*every person who moves smirks*", C:"*each man who dances smiles*" does not represent entailment. For the first appearance of a term, its type is written in a subscript and assumed for its later occurrences. Each node is labeled with an ID and a source rule application (i.e. the rule and the IDs of the nodes it applies).

In order to illustrate how the natural tableau works, we give a tableau in Fig. 1 which refutes a given textual entailment problem (TEP) that does not represent entailment. The tableau starts with the counterexample, 1 and 2 nodes, of the argument.[2] It is expanded using the MON↑ rule (see Fig. 2) which takes into account upward monotonicity (mon↑) of function terms. MON↑ is a branching rule and its application yields two situations. The left branch corresponds to the situations where $A \not\leq B$ holds,[3] and the right one to the situations where $G \not\leq H$ holds, including those where $A \leq B$ holds. Since **every** and **each** are mon↑ in the second argument position, (MON↑) applies to them—its antecedents match the nodes and its consequents are introduced in the tableau. From the resulting branches, the left one is closed

[2] A tableau entry (i.e. node) represents a pair of an LLF and its argument list which is additionally marked with a truth sign. The semantics behind an entry is that the term obtained by applying the LLF to its arguments is evaluated according to the sign. The truth signs $\mathbb{T}$ (true) and $\mathbb{F}$ (false) represent the terms of type t. Annotations on the nodes give information about the building process of a tableau. $\vec{C}$ denotes a sequence of the terms.

[3] For any A_α and B_α lexical terms of the same type α, we say B *contains* A and write as $A \leq B$ iff $\forall \vec{X}(A\vec{X} \to B\vec{X})$, where $\mathbb{F} \to \mathbb{T}$. In this way, $\mathbf{dog}_{et} \leq \mathbf{animal}_{et}$ holds since based on the lexical knowledge $\forall x_e(\mathbf{dog}\, x_e \to \mathbf{animal}\, x_e)$ holds. Notice that in case of truth values the containment relation $\leq$ coincides with the material implication $\to$.

$$\frac{\begin{array}{c} GA : [\vec{C}] : \mathbb{T} \\ HB : [\vec{C}] : \mathbb{F} \end{array}}{\begin{array}{ll} A : [\vec{d}] : \mathbb{T} & G : [Q, \vec{C}] : \mathbb{T} \\ B : [\vec{d}] : \mathbb{F} & H : [Q, \vec{C}] : \mathbb{F} \end{array}}\text{MON}\uparrow$$

where G or H is mon$\uparrow$ and $\vec{d}$ and Q are fresh

$$\frac{\begin{array}{c} GA : [\vec{C}] : \mathbb{T} \\ HB : [\vec{C}] : \mathbb{F} \end{array}}{\begin{array}{ll} A : [\vec{d}] : \mathbb{F} & G : [Q, \vec{C}] : \mathbb{T} \\ B : [\vec{d}] : \mathbb{T} & H : [Q, \vec{C}] : \mathbb{F} \end{array}}\text{MON}\downarrow$$

where G or H is mon$\downarrow$ and $\vec{d}$ and Q are fresh

$$\frac{\begin{array}{c} A : [\vec{C}] : \mathbb{T} \\ B : [\vec{C}] : \mathbb{F} \end{array}}{\times}\leq \times$$

where $A \leq B$

Figure 2. The tableau rules for monotonic operators

as it is identified as inconsistent with the help of the closure ($\leq \times$) rule in Fig. 2. The right branch is further developed by applying (MON$\downarrow$) to 5 and 6. The rule application takes into account the downward monotonicity (mon$\downarrow$) of **every** and **each** in the first argument position. From the new branches, the right one is closed due to inconsistency while the left one is grown from 8 and 9 with the rules that treats **who** as a conjunction between terms of type *et*. In the end, each branch of the tableau is closed, i.e. the tableau is closed, which indicates the failure in refuting the textual entailment; therefore the proof for the entailment relation is found.

The previous works [2, 1] extend the natural tableau in several directions in order to make it viable beyond toy examples. First, we incorporate the following syntactic types in the type system: `n` for nouns, `np` for noun phrases, `s` for sentences and `pp` for prepositional phrases. From the perspective of theorem proving, LLFs with syntactic types offer fine-grained matching between tableau entries and antecedents of the rules. For instance, an LLF of type *et* can ambiguously correspond to a noun or an intransitive verb. This ambiguity needs to be resolved before applying a rule; this complicates a proof procedure. On the other hand, an LLF of syntactic type contains no such ambiguity. Interaction between the syntactic and semantics types is established by the subtyping $\sqsubseteq$ relation defined as:

(a) $e \sqsubseteq \texttt{np}$, $\quad \texttt{s} \sqsubseteq t$, $\quad \texttt{n} \sqsubseteq et$, $\quad \texttt{pp} \sqsubseteq et$;

(b) for any $\alpha_1, \alpha_2, \beta_1, \beta_2$ types, $(\alpha_1, \alpha_2) \sqsubseteq (\beta_1, \beta_2)$ iff $\beta_1 \sqsubseteq \alpha_1$ and $\alpha_2 \sqsubseteq \beta_2$

An LLF with syntactic types still has the similar form as its semantic counterpart. For instance, compare the LLF in (1), where `vp` abbreviates $(\texttt{np}, \texttt{s})$, to the one in 1 of Fig. 1.

$$\textbf{every}_{\texttt{n,vp,s}}\ (\textbf{who}_{\texttt{vp,n,n}}\ \textbf{move}_{\texttt{vp}}\ \textbf{person}_{\texttt{n}})\ \textbf{smirk}_{\texttt{vp}} \tag{1}$$

$$\texttt{modifierSet} : \texttt{LLF} : \texttt{argList} : \texttt{truthSign} \tag{2}$$

Moreover, for a single lexical element it is rarely necessary to have two lexical terms with semantic and syntactic types. This is facilitated by the

subtyping relation as $\mathbf{smile}_{\mathtt{vp}}c_e$ and $\mathbf{person}_{\mathtt{n}}c_e$ terms are well-formed and there is no need for the semantic terms $\mathtt{smile}_{et}$ and $\mathtt{person}_{et}$.

The second extension to the system is the introduction of a modifier set in tableau entries, see (2). As argued in [2], the set is used for saving and later retrieving a modifier term that is indirectly applied to its head. This technique is used for the nouns with several adnominals or for the verbs with several adverbs. The trick with the modifier set solves the problem of event modification without introducing an event variable in an LLF and losing the *essence* of natural logic.[4]

The last extension is in terms of tableau rules. While [22] presents the rules for monotonic operators, Boolean connectives, quantifiers, etc., the natural tableau lacks the rules for many phenomena that are often found in open domain text. The inventory of rules is further enriched with the rules for nominal and verbal modifiers, prepositions, copula, passive constrictions, expletives, etc. The procedure of collecting the rules is described in [1]; the rules are presented in [2].

3 Natural logic theorem prover

In order to automatize the natural tableau system, we implemented the theorem prover for natural logic, called NatPro. The prover is written in the Prolog language. It expects a finite set of signed LLFs as an input and, using the inventory of rules and the knowledge base, tries to build a tableau over the input. Below we describe the components of the prover. The organization of the knowledge base is explored in more details as it is relevant for certain judgments discussed in Sec. 5.

3.1 Knowledge base (KB)

Importance of background knowledge for open domain textual entailment is extremely high. Background knowledge is of two kinds. One is *extra-linguistic knowledge*, also called *encyclopedic* or *world knowledge*, which mainly involves information how the current state of the situation or world is organized. For example, correct classification of the TEPs (K1) and (K2) in Table 1 require extra-linguistic knowledge. Another type of knowledge is *linguistic* which encompasses the lexical knowledge and its compositions, where composition is governed by the grammar. In other words, this type of knowledge is encoded in the language itself. For instance, the linguistic

[4]Notice that the extension of the natural tableau is conservative—the tableaux generated with the system of [22], e.g., the one in Figure 2, are still available in the extended version.

Table 1. Examples of textual entailment problems

ID	Premise	Conclusion
K1	Barcelona defeated Real Madrid 4-0	Barcelona thumped Real Madrid
K2	Messi won Ballon d'Or	Lionel Messi won the Ballon d'Or award
K3	Not all birds fly	There is some bird that does not fly
K4	The piano is being played by the boy	The boy is playing a musical instrument
S5	A sad girl is crying	A girl is weeping
S6	The child is crying	The child is screaming and weeping
S7	The child is screaming	The child is weeping

knowledge is required for solving the TEPs (K3) and (K4) in Table 1. Despite this distinction the border between the linguistic and extra-linguistic knowledge is quite vague.

Currently we model only the linguistic knowledge in the tableau system by taking WordNet [12] as a lexical knowledge base. At this moment, for simplicity we employ only the hypernymy and antonymy relations of WordNet. Using these relations, we define the containment ($\leq$) and disjoint ($|$) relations over lexical terms A and B as follows. $A \leq B$ holds if there exist the WordNet synsets S_A and S_B such that S_B is at least as general as S_A and some senses of A and B are in S_A and S_B, respectively. According to this definition, $\mathbf{cry}_{\mathtt{vp}} \leq \mathbf{scream}_{\mathtt{vp}}$ and $\mathbf{scream}_{\mathtt{vp}} \leq \mathbf{cry}_{\mathtt{vp}}$ because there exists senses of $\mathbf{cry}_{\mathtt{vp}}$ and $\mathbf{scream}_{\mathtt{vp}}$ that belong to the same synset with the meaning of "*utter a sudden loud cry*". Similarly, A and B terms are disjoint, written as $A \,|\, B$, if some of their senses belong to the synsets that are in the antonymy relation. In this way, $\mathbf{empty}_{\mathtt{n,n}} \mid \mathbf{full}_{\mathtt{n,n}}$ holds as there exist antonymous singleton synsets with the senses related to the terms. This kind of treatment of lexical semantics assumes that each lexical term has multiple senses independently from the context it occurs in. Due to this feature we call it the *multi-sense* approach.

The proof search with multiple senses can be interpreted as a search over lexical senses too: are there senses for the lexical terms that licenses the given TEP as entailment or contradiction? For example, proving equivalence of the sentences in (S5) literally means that based on certain lexical semantics the sentences are equivalent. The multi-sense approach has a shortcoming as it validates the entailment relations like one in (S6). But it does not lead to any relation between "*weep*" and "*scream*" since their senses are not related in WordNet, hence there is no logical relation between the sentences in (S7).

Compared to the sense disambiguated approach, with multiple senses it is more likely that there is a relation between two terms. Taking into account that rule-based RTE systems often have problems related to knowledge sparsity, the multi-sense approach seems an interesting option for the

tableau prover to overcome the sparsity to some extent. Moreover, the multi-sense approach is simpler and more robust since it does not require to disambiguate word senses, the latter representing an open problem in NLP.[5]

3.2 Inventory of the rules (IR)

The inventory of the rules is the most crucial component of the tableau system. The tableau rules encode simple reasoning steps and instructions how to decompose large LLFs into smaller pieces. According to the phenomena the rules account for, they can be roughly categorized into two groups. The first group of the rules are mainly the ones presented in [22]. These rules unfold the semantics of the LLFs involving the terms with the algebraic properties: Boolean, upward monotone, downward monotone, etc. Hence the rules of this group are called *algebraic*. The rules for determiners and those concerning the format of tableau entries are also part of the algebraic rules. This group also contains so-called *admissible* rules—the rules that are redundant from a completeness point of view but represent a shortcut for several rule applications. Many algebraic rules either introduce a fresh entity, employ an old entity or has a branching structure; these make the rules inefficient from the theorem proving perspective. The admissible rules can be seen as a way of applying some of the inefficient rules in an efficient manner.

The second group counts the rules that essentially deal with LLFs modeling common syntactic constructions; let them be *syntactic* rules. The syntactic rules analyze adjective-head and adverb-head pairs, prepositional phrases, passive constrictions, compound nouns, the constructions with a copula, auxiliary and light verbs, etc.; most of these rules are described in [2].

3.3 A proof engine (PE)

In the prover, a tableau is represented as a list of tableau branches, where a branch itself is a list of signed LLFs. For terminating a tableau building process in a finite time, we set the limit for the number of rule applications (RAL). If there is an open branch after the RAL is reached, the tableau is considered open. In order to make sure that each branch gets its fair share of rule applications, after each rule application on a branch, its next branch

[5]The upper limit of a word sense disambiguation (WSD) system with respect to the WordNet senses (i.e. fine-grained senses) is quite low as the inter-annotator agreement is only 72.5% [24]. Even for course-grained senses the inter-annotator agreement is only 86.5% [23].

is processed; in case of the last branch, the first branch is processed. In average, this strategy guarantees finding an open branch, if it exists, earlier than the strategy where a shift to a new branch happens only if the current one closes.

The *proof engine* of the prover represents an algorithm that decides which of the rule applications is to be carried out next. For the rule application strategy it takes into account *efficiency* of each tableau rule, where efficiency depends on the properties of a rule, e.g, whether it is branching, employs old constants or produces fresh ones. For example, (MON↑) and (MON↓) from Fig. 2 are inefficient rules as they are branching and introduce fresh constants. These rules also have an interesting feature: their antecedents are not equivalent to the disjunction of the consequents. So, discarding a node from a branch after applying such a *non-equivalent* rule to the node might lead to incompleteness. In light of the non-equivalent rules, the proof engine needs to track down a rule application history for each branch in order to avoid performing the same rule application more than once. Recording the history is also important for the admissible rules.

4 LangPro: natural language theorem prover

The section describes the architecture of the theorem prover that reasons over natural language expressions. It is also illustrated how the prover operates on a certain TEP. In the end, the evaluation results of the prover on RTE datasets are presented.

4.1 The architecture

A theorem prover for natural language, called LangPro, is obtained by combining a CCG parser[6], the LLF generator [1] and the natural logic theorem prover (see Fig. 3). In case of the parser component we have at least two choices: the C&C [7] and EasyCCG [16] parsers.[7] It is interesting to employ both parsers as they are based on different approaches. Hereafter, the version of LangPro based on C&C or EasyCCG is referred as ccLangPro or easyLangPro, respectively.

[6]In LLFs, lexical elements are interpreted as functions. Since this interpretation is fundamental for categorial grammars (CGs), the CG-style derivation is a good starting point for obtaining LLFs. To the best of our knowledge, the only wide-coverage CG-based parsers analyze sentences in Combinatory Categorial Grammar (CCG) [25].

[7]In the current settings of LangPro, we employ C&C with the model trained on the improved corpus [13]. While EasyCCG acts as a multi-parser, returning n-best derivations, currently only the first best derivation is employed.

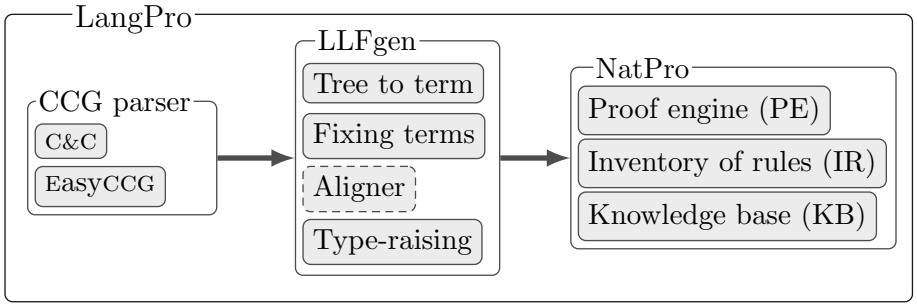


Figure 3. The architecture of LangPro

While describing the architecture of LangPro, in Fig. 4 we also demosntrate how the prover operates on a particlar TEP—SICK-1417 from the SICK dataset [21]. First, the premise and the conclusion are parsed by a parser. The obtained CCG derivation trees are processed by the LLF generator (LLFgen): first, the derivations are converted into CCG terms by removing directionality from the CCG categories, then the terms are *corrected* and further transformed into LLFs by type-raising the quantified NPs.[8] Depending on the choice of the parser, we call an obtained LLF a ccLLF or easyLLF.

Often a premise and a conclusion share several multiword phrases, analysis of which is not relevant for the classification. Due to this reason, many RTE systems adopt alignment techniques that help the systems to concentrate on relevant parts of the text [9]. In LangPro alignment is carried out by the term aligner, a part of LLFgen. The aligner finds the subterms occurring in both CCG terms and replaces them with fresh terms. While doing so, the candidate subterms are checked for monotonicity: they are aligned iff they are not mon$\downarrow$ (for more details see SICK-1207 in Sec. 5). Sometimes the alignment procedure leaves the CCG terms unchanged, like in the current example. Since the alignment procedure may eliminate the chance of finding a proof, the original LLFs are tested if the prover is not able to find a proof with the aligned LLFs. Henceforth, $\overline{\text{cc}}$LLFs and $\overline{\text{easy}}$LLFs will denote the aligned versions of the corresponding LLFs. The aligner is an optional component of the prover which contributes to short proofs, hence to the prefromance too [3].

For each CCG derivation, LLFgen returns a list of LLFs. In the running

[8]The CCG terms are typed with the syntactic types corresponding to the CCG categories. They are not well-formed λ-terms due to the remains of the *type changing* (i.e. *lexical*) combinatory rule of the CCG parsers. The CCG terms are fixed with term-rewriting rules which mainly explain the type changes (see Fig. 4) or modify the terms for better semantic adequacy. More details about the LLFgen procedures can be found in [1, Sec. 3].

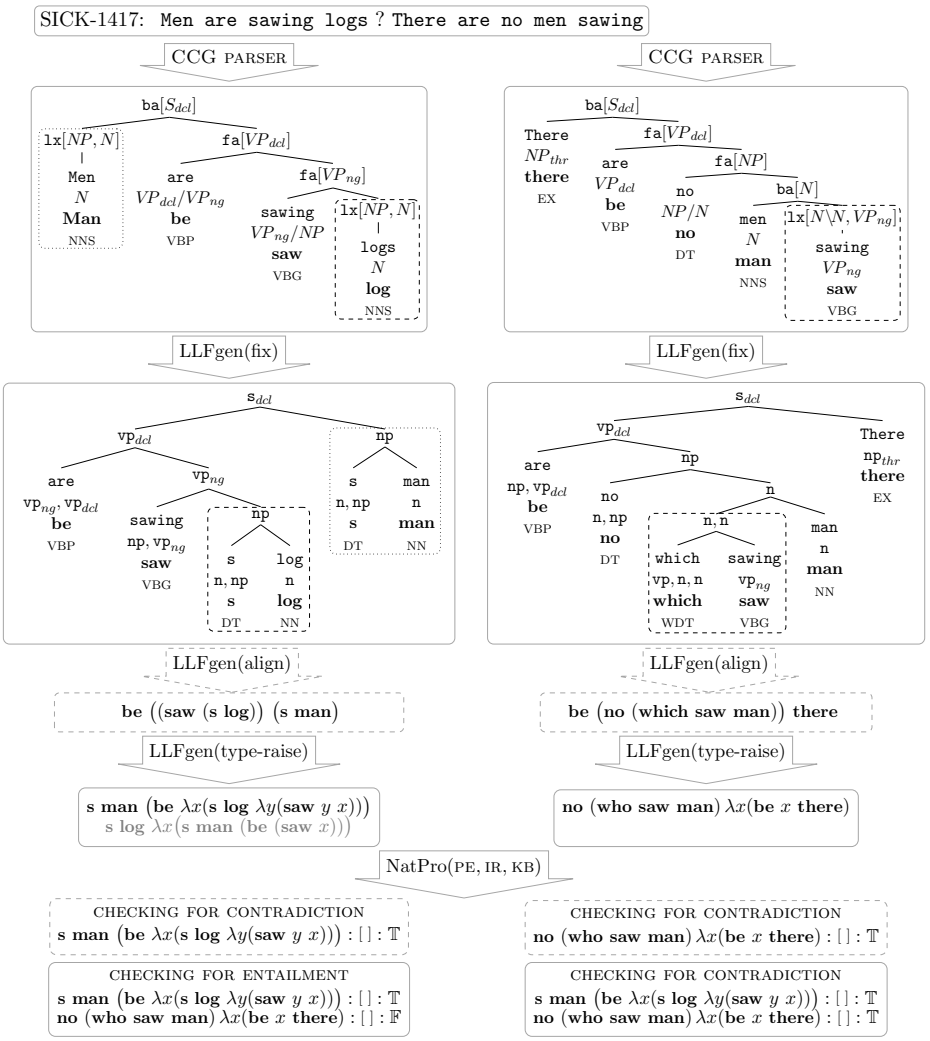


Figure 4. LangPro processes the TEP from SICK[21]. The CCG derivations obtained from a parser, namely C&C, are presented as a tree. The terminal nodes are annotated with a token, a CCG category, a lemma and a part of speech (POS) tag while the non-terminal ones are marked with a CCG category and a combinatory rule that combines the constituent(s). The constituents with a type changing rule and their fixed versions are framed. The term $\mathtt{s}_{\mathtt{n},\mathtt{vp},\mathtt{s}}$ stands for a plural quantifier. VP_i category and $\mathtt{vp}_i$ type abbreviate $S_i\backslash NP$ and $(\mathtt{np}, \mathtt{s}_i)$ respectively, where i is a category feature employed by the CCG parsers.

example, there are two LLFs for the premise due to the possible ambiguity between the quantifier scopes while for the conclusion only one LLF is generated. At this moment, LLFgen does not take the semantics of a quantifier into account during type-raising; that is why despite their semantic equivalence, both LLFs for the premise are generated. In the current settings of LangPro, for each CCG derivation only the first LLF is considered by the prover. Notice that the scope order in the first LLF usually resembles the order in the surface level.

NatPro first checks each LLF for contradiction.[9] If all LLFs are found self-consistent, they are checked for entailment, and if no proof is found then they are checked for contradiction (see the tableau in Fig. 5 which proves the running example as entailment). If the entailment relation is proved, there is no need for checking the LLFs for contradiction. In this way, neutral problems are the most time-consuming; for a neutral single-premised problem, in total 8 tableaux are constructed for the aligned and non-aligned LLFs.

4.2 Performance

For the training and evaluation purposes, we use the SICK [21] and FraCaS [11] RTE datasets for the following reasons.[10] The datasets contain relatively short sentences which is expected to increase correct analysis by CCG parsers. The FraCaS problems require no lexical knowledge while the SICK problems require only linguistic knowledge. The training process is not fully automatized: if LangPro misclassifies a problem, the process is debugged—either a new tableau rule or knowledge fact is introduced or LLFgen is further improved. More details about the training and development phases can be found in [1, Sec. 5].

In Table 2 both versions of the prover, ccLangPro and easyLangPro, are evaluated and compared to state-of-the-art results.[11] ccLangPro achieves

[9] If an LLF is found self-contradictory, i.e. the tableau initiated by it closes, there is a high chance that the source CCG derivation is erroneous [3, Sec. 4]. If one of the LLFs is self-contradictory, the prover aborts and returns the neutral answer. The same decision is made when one of the LLFs has a different type from the rest.

[10] FraCaS is a small set containing semantically challenging multi-premise problems. It is available at: `http://www-nlp.stanford.edu/~wcmac/downloads`. Currently few RTE systems are able to cope with the FraCaS problems. On the other hand, SICK is a large dataset intended for compositional distributional semantic models. It was used as a benchmark at the SemEval RTE challenge [20]: `http://alt.qcri.org/semeval2014/task1`. The TEPs in both datasets are human annotated with three labels: entailment, contradiction and neutral. We adopt the partition of SICK from the RTE challenge [20] and refer to these parts as SICK-trial, SICK-train and SICK-test.

[11] In case of FraCaS, the training and testing data are the same for LangPro; so comparison to the system that did not have a close look at the test problems should be

Table 2. Performance of the versions of LangPro on FraCaS and SICK

(a) Results on FraCaS's section 1 on generalize quantifiers involving both single- (one) and multi- (all) premise problems

Systems' Acc%	One (44)	All (74)
NatLog'07 [18]	84	-
NatLog'08 [19]	**98**	-
NaturalLI [4]	95	-
L&S'13 [17]	70/89	50/80
DCS [27]	79	80
HOL [14]	-	77
ccLangPro	95	**85**
easyLangPro	80	**81**
Baseline (major)	45	50

(b) Results on the test portion of SICK(4927). uniLangPro is a prover unifying judgments of ccLLF- and easyLLF-based provers.

Systems	Prec %	Rec %	Acc %
Illinois-LH	81.56	**81.87**	**84.57**
ECNU	84.37	74.37	83.64
UNAL-NLP	81.99	76.80	83.05
SemantiKLUE	85.40	69.63	82.32
Meaning Factory	93.63	60.64	81.59
ccLangPro	97.53	57.26	80.90
easyLangPro	97.63	57.83	81.15
uniLangPro	**97.67**	61.01	82.50
Baseline (major)	-	-	56.69

higher results on FraCaS than easyLangPro. This is explained by using ccLLFs during the entire training process, which made LLFgen to work better on ccLLFs. Despite fitting LLFgen to ccLLFs, easyLangPro still obtains high results on FraCaS and even beats ccLangPro on SICK-test. uniLangPro is a prover unifying both provers: it finds a proof iff one of the provers finds a proof. On SICK, the versions of the prover is compared to the top systems of the SemEval challenge [20]. Although LangPro shows a low recall on SICK-test, it obtains a competitive accuracy along with an almost prefect precision. Note that most of these errors are due to the noisy gold labels in the dataset. Several problems with noisy gold labels are discussed in the next section. Additional information about the comparison based on the SICK data is given in [1] and [3, Sec. 5].

We briefly compare our prover to two related systems: NatLog [19] and NutCracker [6]. LangPro improves over NatLog in terms of having full-fledged logic and proof system over the logical forms. As a result it can process more complex and multi-premised TEPs. Compared to NutCracker, our prover can reason over higher-order and monotone terms. NatPro is also specially tuned for natural reasoning in contrast to the off-the-shelf provers and model builders incorporated in NutCracker. Further details of the comparison is given in [1, Sec. 6]. The demo version of LangPro is available online.[12]

understood in terms of the expressive power of a system. For SICK the test data, SICK-test, was held out during training, so comparison to related systems can be made in terms of performance too.

[12] `http://lanthanum.uvt.nl/labziani/tableau`

5 Solving textual entailment problems

The section shows in details how LangPro processes various problems drawn from the SICK [21] and FraCaS [11] datasets. Each presented problem is accompanied with its dataset ID, the gold label and two judgments by the prover—one based on C&C and another on EasyCCG derivations. The selected set of examples intend to highlight the issues of natural language theorem proving and give the clues for further improvements.

5.1 True entailments and contradictions

First, we are going to discuss the positive (entailment or contradiction) problems that were correctly proved either by ccLangPro or by easyLangPro.

SICK-1417 GOLD: `cont.` LangPro(C&C/EasyCCG): `cont./neut.`

Men are sawing logs

There are no men sawing

SICK-1417: this is the problem from Fig. 4 in Section 4. Both parsers correctly analyze the premise while in case of the conclusion only EasyCCG makes mistake: "*men sawing*" is identified as a constituent $[\text{men}_{N/N}\ \text{saw}_N]_N$. This mistake is crucial for easyLLFs hence the proof over them is not found. Fortunately, correct C&C derivations result in semantically adequate ccLLFs. LangPro proves them as inconsistent in 10 rule applications; see the proof in Fig. 5.
Conclusion: while the proof is not found with EasyCCG due to a wrong derivation, the C&C derivations salvage the situation.

SICK-8147 GOLD: `ent.` LangPro(C&C/EasyCCG): `ent./ent.`

The girl $[\text{in blue}]_1$ is chasing the base runner $\big[\text{with a number } [\text{on the jersey}]_3\big]_2$

The girl $[\text{in blue}]_1$ is chasing the player $\big[\text{with a number } [\text{on the jersey}]_3\big]_2$

SICK-8147: the problem contains sentences each having three PPs, marked with brackets and indexed; this makes the sentences challenging for the parsers. Apart from the different analyses for each NP[13], the derivation trees from C&C and EasyCCG also differ in PP-attachments. For both sentences, C&C treats PP_3 as an argument of "*number*" while EasyCCG analyzes it as a modifier of "*a number*". In both sentences, EasyCCG wrongly but in a consistent way treats PP_2 as a VP modifier; PP_2 gets mixed analyses

[13] EasyCCG usually analyzes NPs with post-modifiers in the NP-S style, i.e. a determiner is grouped with a head earlier than post-modifiers: $\big[[\text{the girl}]_{NP}\ [\text{in blue}]_{NP\backslash NP}\big]_{NP}$. On the other hand, C&C trained on rebanked CCGbank [13] prefers the Nom-S analysis: $\big[\text{the}_{NP/N}\ [\text{girl}_N\ [\text{in blue}]_{N\backslash N}]_N\big]_{NP}$.

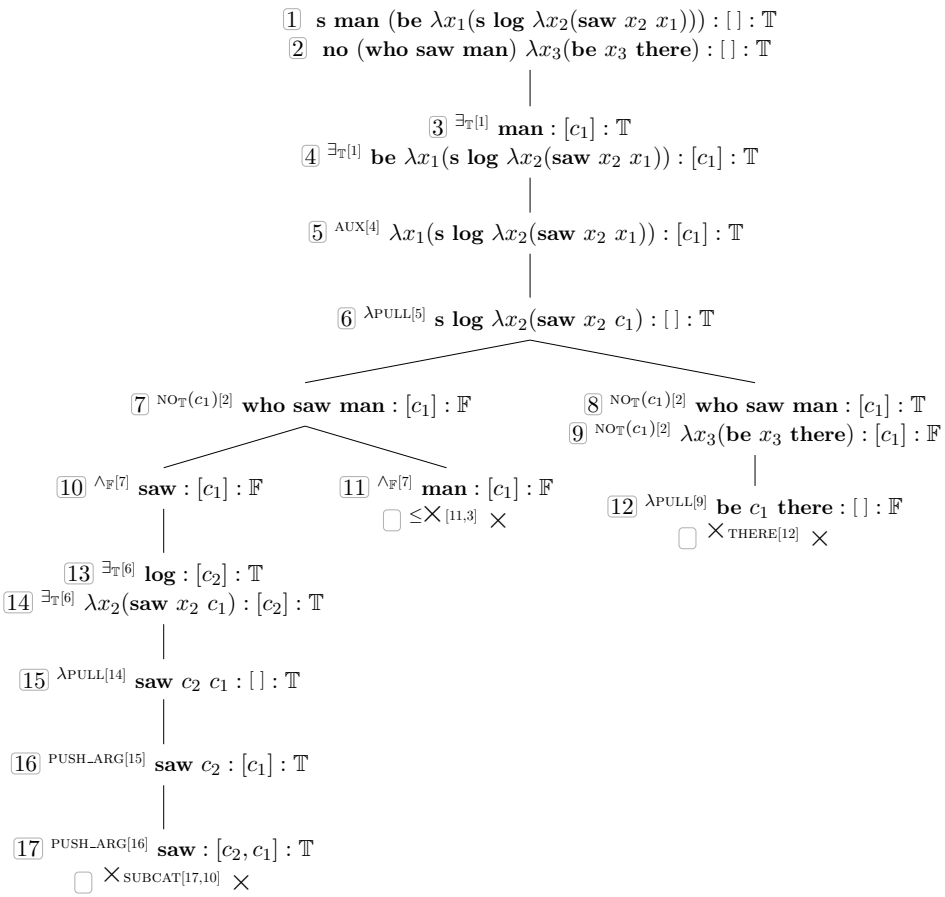


Figure 5. The closed tableau proves SICK-1417 as entailment. The intuition behind the employed rules (e.g., λPULL, AUX, $\text{NO}_{\mathbb{T}}$, etc.) can be read from the tableau.

from C&C: correctly identified as a modifier of "*player*" in the conclusion but analyzed wrongly in the premise, like in case of EasyCCG. Due to these differences, the corresponding ccLLFs and easyLLFs, including their aligned versions, also differ from each other. In particular, the terms for PP_2 are aligned in the $\overline{\text{easy}}$LLFs; but for the $\overline{\text{cc}}$LLFs, the shorter subterms of "*a number on the jersey*" are aligned because C&C analyzes PP_2 in a mixed way and assigns different categories to "*with*" in the sentences.

For both versions of aligned LLFs, LangPro finds proofs for the entailment relation. Due to the poor alignment for the $\overline{\text{cc}}$LLFs, the tableau was closed in 20 rule applications while for the $\overline{\text{easy}}$LLFs, with the better alignment, in 8 applications. Despite the wrong attachments of PP_2 in the

$\overline{\text{easy}}$LLFs, proof search is more efficient because the attachments were *consistent* which contributed to the better alignment. In case of the $\overline{\text{cc}}$LLFs, the attachments of PP_2 were inconsistent (though one of them was correct) which finally costs much in terms of a lengthy proof. Moreover, finding a proof for the $\overline{\text{cc}}$LLFs would be impossible if we did not introduce the $\times_{\text{PP\_ATT}_{\mathbb{T}}}$ rule, which abstracts from inconsistency in PP-attachments. As a result, the rule identifies the nodes in (3) as inconsistent; terms written in CamelCase are the aligned subterms while the individual constant c stands for "*the base runner*".

Capturing the entailment **base runner** $\leq$ **player** is a main part of solving SICK-8147. This piece of information is necessary for the tableau to be close. The multi-sense approach tries to find some senses of the nouns for which the entailment relation holds. Such senses are found in the knowledge base as the synset {"*base runner*", "*runner*"} is indirectly subsumed by {"*player*", "*participant*"} synset in WordNet.[14]

$$\frac{\begin{array}{r}\{\textbf{with aNumberOnTheJersey}\} : \textbf{chase} : [c, \textbf{theGirlInBlue}] : \mathbb{T} \\ \textbf{with} : [\textbf{aNumberOnTheJersey}, c] : \mathbb{F}\end{array}}{\times}\times_{\text{PP\_ATT}^{*}_{\mathbb{T}}} \quad (3)$$

Conclusion: the consistent (possibly wrong) analyses of PP-attachments leads to the better alignment of terms, which itself contributes to the shorter proof. A wrongly attached structurally ambiguous PP can be identified and correctly interpreted with the help of the special rules. The multi-sense approach also works well for this case.

FraCaS-18 GOLD: `ent.` LangPro(C&C/EasyCCG): `neut./ent.`

Every European has the right $\big[$[to live in Europe]$_{VP_{to}}\big]_{N\backslash N}$
Every European is a person
Every person who has the right [to live in Europe] can travel freely within Europe
Every European can travel freely within Europe

FraCaS-18: the textual entailment contains multiple premises but this is not a problem for the prover. The challenge in this example is to obtain decent derivations and to convert them in LLFs. The C&C line fails in the beginning when the parser fails to return the derivations for the first two sentences which contain relevant information for the entailment. Fortunately, EasyCCG gets all the sentences parsed. The produced easyLLFs are

[14] Tthe prover is also able to derive **base runner** $\leq$ **player** relation by first capturing **base runner** $\leq$ **runner** using a rule for subsective adjectives and then combining it with **runner** $\leq$ **player** relation retrieved from WordNet. Based on the information **runner** $\leq$ **player**, unfortunately the multi-sense approach also leads to the proof of "*A runner won*" entails "*A player won*".

not proper λ-terms since LLFgen, at this moment, does not have a remedy for the rule that changes the syntactic type $\mathtt{vp}_{to}$ into $(\mathtt{n},\mathtt{n})$. Despite this shortcoming, LangPro is still able to operate on the easyLLFs and find a proof for entailment in 43 rule applications (actually, only a few of those applications contribute to the proof). In case the proof needed to decompose the LLF corresponding to "*the right ... Europe*", due to the unexplained type-changing rule, the prover would fail to do so and fail to find a proof. Notice that in this example, the aligner is useless as there is no multiword phrase shared by all the sentences.

Conclusion: C&C failed to parse several sentences, however EasyCCG saved the situation. The obtained easyLLFs were not well-formed terms, nevertheless the prover is able to process them as long as the decomposition of the ill-formed subterms is not necessary for the proof.

SICK-1207 GOLD: `cont.` LangPro(C&C/EasyCCG): `cont./cont.`

A woman is not talking on a telephone

A woman is talking on a telephone

SICK-1207: this is a dubious case. One might identify the problem as neutral or contradiction depending whether the indefinite NPs "*a woman*" and "*a telephone*" are co-referencing to the same referents. The human annotations in SICK show a strong tendency towards co-reference of this kind of NPs (the similar problems, SICK-363 and 1989, with the similar judgments are given in Table 3).

In order to support the co-reference, we choose a simple solution that comes for free with the current settings of LangPro. In particular, the aligner is used for this purpose: aligning the identical indefinites make them to refer to the same entity. The aligned LLFs for the problem are given in (4) and (5). Using the aligned LLFs, the prover is able to find a proof in 3 rule applications.

$$\mathbf{not}_{\mathtt{vp},\mathtt{vp}}\ \mathbf{beTalkOnATelephone}_{\mathtt{vp}}\ \mathbf{aWoman}_{\mathtt{np}} \tag{4}$$

$$\mathbf{beTalkOnATelephone}_{\mathtt{vp}}\ \mathbf{aWoman}_{\mathtt{np}} \tag{5}$$

Conclusion: the simple solution with the alignment technique accounts enough well for the co-reference of indefinite NPs. It also makes proofs extremely short.[15]

Above we tried to give the TEPs that were correctly classified in spite of the various shortcomings. Usually this kind of TEPs are rare. For exam-

[15]If in the premise "*a woman*" is replaced by "*a person*", the alignment approach cannot contribute to the proof. More general solution to the co-reference of indefinites is achieved when the negation takes a wide scope. Implementing the latter approach requires further development of LLFgen. This approach is left for future work.

ple, while in our examples the judgments of ccLangPro and easyLangPro diverged in the half of the cases, based on the development set—SICK-train (4500 problems)—the judgments diverge only for 3.3% of the TEPs.

5.2 False entailments and contradictions

Given the almost perfect precision of the prover, the false positive problems represents a special case of interest.

SICK-8461 GOLD: `neut.` LangPro(C&C/EasyCCG): `cont./cont.`

A man with no hat [is sitting on the ground]$_1$
A man with a backwards hat [is sitting on the ground]$_1$

SICK-8461: the ccLLFs and easyLLFs for both sentences are quite similar; the ccLLF and easyLLF for the premise are given in (6) and (7), respectively. The only difference is in the analyses of the verb phrases: whether **be** takes **sit** or the whole verb phrase as an argument. This difference has no influence on proof search since the auxiliaries are currently treated as the identity function.

$$\textbf{no hat}\ \lambda y\big(\textbf{a}\ \big(\textbf{with}\ y\ \textbf{man}\big)\ \lambda x\big(\textbf{the ground}\ \lambda z(\textbf{on}\ z\ (\textbf{be sit})\ x)\big)\big) \quad (6)$$

$$\textbf{no hat}\ \lambda y\big(\textbf{a}\ \big(\textbf{with}\ y\ \textbf{man}\big)\big(\textbf{be}\ \lambda x(\textbf{the ground}\ \lambda z(\textbf{on}\ z\ \textbf{sit}\ x))\big)\big) \quad (7)$$

$$\mathtt{a}_{(et)(et)t}\ \lambda y\big(\mathtt{no}_{(et)(et)t}\ \mathtt{hat}_{et}\ \lambda x(\mathtt{with}_{e(et)et}\ x_e\ \mathtt{man}_{et}\ y_e)\big)\ \mathtt{sleep}_{et} \quad (8)$$

In contrast to the surface level, **no hat** takes the widest scope in the LLFs. The reason is usage of the terms with syntactic types. While using the terms of semantic types, it is possible that **no hat** takes a narrow scope, see (8). But in case of the syntactic types, **no hat** can not be type-raised in the PP because a determiner of type $(\mathtt{n},(\mathtt{np},\mathtt{s}),\mathtt{s})$ cannot take a term of type $(\mathtt{np},t)$ or (e,t) for its second argument.[16]

It is obvious that if the sentences were understood with "*no hat*" and "*a backwards hat*" having the widest scope, then they would be inconsistent. This is why the prover classifies the problem as contradiction. The proofs for both versions of LLFs were found in 5 rule applications. The alignment of VP_1 does not affects the proof search as the relevant terms "*no hat*" and "*a backwards hat*" are analyzed and contrasted to each other before VP_1 is processed.

Conclusion: the desirable scope order for quantifiers is not obtained due to less-flexible syntactic types, which in the end leads to the wrong prediction. This mistake seems minor taking into account that the similar

[16]This issue can be solved by introducing a semantic counterpart of the determiner that is of type $(et)(et)t$, but this itself will further require introduction of semantic counterparts of other terms. The latter complicates the proof search.

problems, e.g., SICK-8562 in Table 3, receive mixed judgments (neutral or contradiction) in SICK.

SICK-7402 GOLD: `neut.` LangPro(C&C/EasyCCG): `cont./cont.`

There is [[no man] and [child kayaking through gentle waters]]
A man and a young boy are riding in a yellow kayak

SICK-7402: the problem is neutral as the sentences are informative with respect to each other, but the prover identifies it as contradiction. The reason is the wrong derivation trees where "*no*" scopes only over "*man*". In this way, the premise implies that there is no man while the conclusion asserts the contrary—a man is riding in a kayak. The prover identifies this inconsistency and classifies the problem as contradiction.
Conclusion: the mistakes by the C&C and EasyCCG parsers misled the prover. In general, it is very rare that the mistake by a parser leads to a false positive.

FraCaS-58 GOLD: `neut.` LangPro(C&C/EasyCCG): `neut./ent.`

Most Europeans [who are resident in Europe] can travel freely within Europe
Most Europeans can travel freely within Europe

FraCaS-58: the judgments based on the ccLLFs and easyLLFs differ from each other. The rationale for the proof found over the easyLLFs is the non-restrictive (i.e. appositional) interpretation of the relative clause. Since the EasyCCG derivations were not used during development of LLFgen, it failed to correct the EasyCCG derivation for the premise.
Conclusion: LLFgen could not correct an unobserved mistake in the EasyCCG derivation. As a result, the mistake caused the false proof for entailment.

SICK-5264 GOLD: `neut.` LangPro(C&C/EasyCCG): `ent./ent.`

A person is folding a sheet
A person is folding a piece [of paper]$_1$

SICK-5264: both decisions of the prover are false according to the gold label. Different analyses of PP_1—as a noun argument by C&C and as an NP modifier by EasyCCG—do not affect the final judgments because all the argument PPs are also treated as modifier PPs with the help of the rules.

The reason for the contradiction proofs is that "*a sheet*" is "*a piece of paper*" according to the multi-sense approach: in WordNet "*sheet*" has a sense that is a hyponym of a sense of "*paper*", and there is the tableau rule that identifies "*a piece of paper*" as "*paper*". The proofs are found in 18 rule appreciations.

Table 3. The false positive examples and the problems with noisy gold (G) labels. The problems are drawn from SICK. The words that were related to each other by LangPro (LP) are in bold while the unrelated ones in italic.

ID	G/LP	Premise	Conclusion
1405	N/E	A **prawn** is being cut by a woman	A woman is cutting **shrimps**
1481	N/E	A deer is jumping over a **wall**	The deer is jumping over the **fence**
1777	N/E	A **boy** is happily playing the piano	A piano is being played by a **man**
4443	N/E	A man is singing to a **girl**	A man is singing to a **woman**
2870	N/C	Two people are riding a **motorcycle**	Nobody is riding a **bike**
2868	E/N	Two people are *stopping* on a **motorcycle**	Two people are *riding* a **bike**
6258	E/N	A *policeman* is sitting on a **motorcycle**	The cop is sitting on a *police* **bike**
344	N/C	P: An Asian woman in a crowd is not carrying a black bag C: An Asian woman in a crowd is carrying a black bag	
545	N/C	P: A woman is standing and is not looking at the waterfall C: A woman is sitting and looking at the waterfall	
8913	N/C	A couple is not looking at a map	A couple is looking at a map
363	C/C	A soccer ball is not rolling into a goal net	A soccer ball is rolling into a goal net
1989	C/C	A girl is playing the guitar	A girl is not playing the guitar
8562	C/N	P: A man *in a hat* is standing outside of a green jeep C: A man *with no hat* is standing outside of a green jeep	

Conclusion: the multi-sense approach makes a co-reference that leads the prover to find a proof for contradiction.

The other false positives that were proved in the same vein as SICK-5264 are give in the upper part of Table 3. The problems were proved due to the relations, like **wall** $\leq$ **fence** and **girl** $\leq$ **woman**, licensed by the multi-sense approach. Notice noise with respect to **motorcycle** $\leq$ **bike** relation. While SICK-2870 rejects it, SICK-2868 and 6258 presuppose the relation. Unfortunately, our prover was not able to capture the latter two entailments as it failed to relate other lexical entries.

On the SICK dataset, the prover rarely finds false proofs and when it does, the multi-sense approach or the noisy labels are the reason in around 80% of the cases. ccLangPro has no false proofs on the first section of FraCaS. On the other hand, easyLangPro finds two false proofs due to non-restrictive relative clauses (e.g., FraCaS-58) and one due to a wrong analysis of the expression "*at most*".

5.3 False neutrals

There can be several reasons for a false neutral: starting from the mistakes by the CCG parsers finishing with a poor strategy for proof search. The prover shows a large number of false neutrals on SICK. In order to find out the reason behind it, we randomly draw 200 problems from SICK-train and analyzed the false positives found there. Around a half of the false positives

Table 4. Examples of false neutrals from SICK. The factors for the failure (Fail) are noisy gold labels (G), the mistakes by the parsers (P), a lack of rule (R) and a lack of knowledge (K). Each problem is marked with the reason of failure.

ID	G	Fail	Premise	Conclusion
4720	E	G	A *monkey* is practicing martial arts	A *chimp* is practicing martial arts
4275	E	R	*A man and a woman* are shaking hands	*Two persons* are shaking hands
4553	E	R	P: A man is emptying a *container made of plastic* C: A man is emptying a *plastic container*	
2763	C	K	A man and woman *are talking*	A man and a woman *are silent*
4974	C	K	Someone is holding a *hedgehog*	Someone is holding a *small animal*
6447	C	P	P: [A small boy [in a yellow shirt]] is laughing on the beach C: There is no small boy [in a yellow shirt [laughing on the beach]]	

were due to knowledge sparsity (see some of the examples in Table 4). A lack of the tableau rule was a reason for a quarter of the problems. This kind of problems also include the cases where an absent paraphrase can be captured with a schema, e.g., X *made of* $Y \rightarrow YX$, like in SICK-4553. The entailments concerning the cardinality need assist from the tableau rules too. The rest of the false neutrals are evenly provoked by noisy gold labels and the mistakes coming from the parsers.

6 Conclusion

We presented the tableau-based theorem prover for natural language, called LangPro. The prover is able to reason over wide-coverage natural language text with the help of the CCG parsers and the module LLFgen producing the logical forms. After training on the SICK and FraCaS dataset, LangPro achieves competitive results with respect to the state-of-the-art RTE systems. The noteworthy virtues of the prover are (i) the almost perfect precision (despite the noisy gold labels of SICK, nearly 98% of the proofs are correct), (ii) the expressive higher-order logic with *natural-looking* formulas, (iii) the proof search strategy specially suited for natural reasoning and (iv) the explanatory decision procedure based on the analytic tableau method. On the other hand, the prover is a rule-based system with a pipeline architecture, which makes it brittle and expensive for training. To the best of our knowledge, LangPro is the only wide-coverage RTE system that is based on natural logic and is able to reason over multiple premises.

For each pair of the human and prover judgments, several textual entailment problems were discussed in details. Due to the accurate nature of LangPro, special attention was paid to the false predictions. For SICK problems, knowledge sparsity was identified as one of the key factors for

the relatively low recall.

In future work, we plan to address the observed problematic issues: knowledge acquisition, word sense disambiguation and generation of semantically adequate LLFs. We intend to populate the knowledge base with further relations from WordNet (e.g., similarity for adjectives and entailment and causation for verbs) and to explore other lexical or phrasal databases for future integration. Both SICK and FraCaS data contain relatively short sentences. It would be interesting to test the prover and the LLF generator module against more naturally occurring text, for instance, the RTE datasets collected from newswire text. The quality of LLFs can be improved by exploring the n-best derivations of EasyCCG.

References

1. Abzianidze, L. A Tableau Prover for Natural Logic and Language. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp.2492–2502. ACL (2015)

2. Abzianidze, L. Towards a Wide-coverage Tableau Method for Natural Logic. In: Murata, T., Mineshima, K., Bekki, D. (eds.), *New Frontiers in Artificial Intelligence*, LNCS, vol. 9067, pp. 66–82. Springer Verlag (2015)

3. Abzianidze, L. A Pure Logic-Based Approach to Natural Reasoning. In Brochhagen, Th., Roelofsen, F., Theiler, N. (Eds.) *Proceedings of the 20th Amsterdam Colloquium*, pp 40-49. ILLC, University of Amsterdam (2015)

4. Angeli, G., Manning, C.D. NaturalLI: Natural Logic Inference for Common Sense Reasoning. *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*. pp. 534–545, ACL (2014)

5. van Benthem, J.F.A.K. Essays in Logical Semantics. Reidel, Dordrecht (1986)

6. Bos, J., Markert, K. Recognising textual entailment with logical inference. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 628-–35. ACL (2005)

7. Clark, S., Curran, J.R. Wide-Coverage Efficient Statistical Parsing with CCG and Log-linear Models. *Computational Linguistics*, 33(4) (2007)

8. Dagan, I., Glickman, O., Magnini, B. The PASCAL recognizing textual entailment challenge. In Quioñero-Candela et al. (eds.),

First PASCAL Machine Learning Challenges Workshop, pp. 177-–190. Springer-Verlag (2006).

9. Dagan, I., Roth, D., Sammons, M., Zanzotto, F.M. Recognizing Textual Entailment: Models and Applications. *Morgan and Claypool Publishers* (2013)

10. D'Agostino, M., Gabbay, D.M., H ahnle, Posegga, J. (eds.) Handbook of Tableau Methods. Springer (1999)

11. Cooper, R., Crouch, D., van Eijck, J., Fox, C., van Genabith, J., Jaspars, J., Kamp, H., Milward, D., Pinkal, M., Poesio, M., Pulman, S. Using the Framework. *Technical Report LRE 62-051 D-16.* The FraCaS Consortium (1996)

12. Fellbaum, Ch. (eds.): WordNet: an Electronic Lexical Database. *MIT press* (1998)

13. Honnibal, M., Curran, J.R., Bos, J. Rebanking CCGbank for Improved NP Interpretation. In *Proceedings of the 48th ACL Conference*, pp. 207–215 (2010)

14. Mineshima, K., Martínez-Gómez, P., Miyao, Y., Bekki, D. Higher-order Logical Inference with Compositional Semantics. In *Proceedings of the 48th ACL Conference*, pp. 2055–2061, ACL (2015)

15. Lakoff, G. Linguistics and Natural Logic. In Davidson, D., Harman, G. (eds.) *Semantics of Natural Language*. Synthese Library, vol. 40, pp. 545–665, Springer (1972)

16. Lewis, M., Steedman, M. A* CCG Parsing with a Supertag-Factored Model. In *Proceedings of the EMNLP 2014*, pp. 990-–1000. ACL (2014)

17. Lewis, M., Steedman, M. Combined Distributional and Logical Semantics. *Transactions of the Association for Computational Linguistics (TACL)*, vol. 1, pp. 179–192. ACL (2013)

18. MacCartney, B., Manning, C. D. Natural logic for textual inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pp. 193-–200. ACL (2007)

19. MacCartney, B., Manning, C. D. Modeling Semantic Containment and Exclusion in Natural Language Inference. In *Proceedings of Coling-08*, Manchester, UK (2008)

20. Marelli, M., Menini, S., Baroni, M., Bentivogli, L., Bernardi, R., Zamparelli, R. SemEval-2014 Task 1: Evaluation of Compositional Distributional Semantic Models on Full Sentences through Semantic Relatedness and Textual Entailment. *Proceedings of the 8th SemEval* (2014).

21. Marelli, M., Menini, S., Baroni, M., Bentivogli, L., Bernardi, R., Zamparelli, R. A SICK Cure for the Evaluation of Compositional Distributional Semantic Models. In *Proceedings of LREC*, Reykjavik (2014)

22. Muskens, R. An Analytic Tableau System for Natural Logic. In: Aloni, M., Bastiaanse, H., de Jager, T., Schulz, K. (eds.) *Logic, Language and Meaning*. LNCS, vol. 6042, pp. 104–113. Springer, Heidelberg (2010)

23. Navigli, R., Litkowski, K. C., Hargraves, O. Semeval-2007 Task 07: Coarse-Grained English All-Words Task. In: *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval)*, pp. 30–35. Prague, (2007)

24. Snyder, B., Palmer, M. The English All-Words Task. In: *Proceedings of the 3rd International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (SensEval-3)*, pp. 41–43. Barcelona (2004)

25. Steedman, M. The Syntactic Process. *MIT press* (2000)

26. Sánchez Valencia, V. Studies on Natural Logic and Categorial Grammar. *PhD dissertation*, University of Amsterdam (1991)

27. Tian, R., Miyao, Y., Matsuzaki, T. Logical Inference on Dependency-based Compositional Semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pp. 79–89, ACL (2014)

STAR TYPES: A TYPE SYSTEM FOR PATTERN CALCULUS

Besik Dundua[1,2] Mikheil Rukhaia[2] Lali Tibua[2]

[1] FBT, International Black Sea University
Agmashenebeli Alley 13km., 0131 Tbilisi, Georgia
[2] VIAM, Ivane Javakhishvili Tbilisi State University
University str. 2, 0186 Tbilisi, Georgia

Abstract

The pattern calculus described in this paper integrates the functional mechanism of the lambda-calculus and the capabilities of pattern matching with star types. Such types specify finite sequences of terms and introduce non-determinism, caused by finitary matching. We parametrize the calculus with an abstract matching function and prove that for each concrete instance of the function with a finitary matching, the calculus enjoys subject reduction property.

1 Introduction

Pattern calculi extend λ-calculus by permitting to abstract over arbitrary terms, not only over variables. For instance, an expression $\lambda_{\{x\}}(f\,x).(x\,a)$ is a term in a pattern calculus, where $(f\,x)$ is called the pattern. Some of other instances of pattern calculi are λ-calculus with patterns [15], pure pattern calculus [12, 13], pattern-based calculi with finitary matching [1], ρ-calculus [5], λ-calculus with first-order constructor patterns [16].

Pattern calculi are expressive enough to encode term rewriting systems correspond to contracting symbols given in [18, 17]. Typed pattern based calculi are formalism for functional programming languages. Therefore, after studying properties of untyped pattern calculus parameterized with finitary matching [1], it is natural to integrate a type system in it and consider a typed version of the calculus. This paper presents star typed pattern calculus with finitary matching and introduces a corresponding subtyping relation. Star types allow to control non-determinism in pattern calculus influenced from finitary matching. We prove that star typed pattern calculus with finitary matching enjoys subject reduction property.

A distinctive feature of our pattern calculus is star types, extending simple types. For instance, we may have a type α^*, which, intuitively, represents a finite sequence of terms (aka a hedge) of type α; or a type

$\alpha_1 \to \alpha_2^* \to \alpha$, which represents a variadic function whose first argument should have the type α_1, followed by arbitrarily many arguments of type α_2. Star types naturally introduce subtyping, based on the monoidal structure of hedges. Fixed arity types (e.g., the binary type $(\theta_1^* \to \alpha_1) \to \theta_2 \to \alpha_2$) and their starred versions form the upper set in the preorder generated by the subtype ordering. We type variables only with types from this set, while constants can get arbitrary types, which may contain stars (e.g., $(\theta^* \to \alpha)^* \to \alpha$) but are not starred themselves (e.g., not $(\theta^* \to \alpha)^*$).

Typed pattern calculi have been studied in [14, 6, 3, 11], just to name a few. To the best of our knowledge, systems with star types have not been considered in this context. Star types (and, in general, regular expression types) proved to be useful for XML processing. XDuce [10], CDuce [4], XHaskell [19], XCentric [7], PρLog [9, 8] are some examples of such applications. Hence, our work also provides a bridge between pattern-calculi (which model pattern-matching in functional programming languages) and XML-processing languages.

2 A Non-deterministic Pattern Calculus

2.1 Untyped Terms

We start with defining the syntax of our untyped pattern calculus. The alphabet consists of the set $\mathcal{X}$ of variables and $\mathcal{F}$ of constants. They are disjoint and countably infinite. The symbols x and f range over $\mathcal{X}$ and $\mathcal{F}$, respectively. *Terms* are defined by the following grammar:

$$A, B ::= x \mid f \mid (A\,B) \mid \lambda_{\mathcal{V}} A.B$$

where $(A\,B)$ is an *application* and $\lambda_{\mathcal{V}} A.B$ is an *abstraction*. We call the term A in the abstraction a *pattern* and the finite set $\mathcal{V}$ of variables is supposed to specify which variables are bound by the abstraction. Application associates to the left, therefore we can write $(A\,B_1 \cdots B_n)$ for $((\cdots(A\,B_1)\cdots)B_n)$. When there is no ambiguity, the outermost parentheses are omitted as well. The letters A, B, C, D are used for terms and the set of terms is denoted by $\mathcal{T}$.

The *sets of free and bound variables* of a term D, denoted $\mathtt{fv}(D)$ and $\mathtt{bv}(D)$ respectively, are defined inductively as follows:

$$\begin{array}{ll}
\mathtt{fv}(x) = \{x\} \qquad \mathtt{fv}(f) = \varnothing & \mathtt{bv}(x) = \varnothing \qquad \mathtt{bv}(f) = \varnothing \\
\mathtt{fv}(A\,B) = \mathtt{fv}(A) \cup \mathtt{fv}(B) & \mathtt{bv}(A\,B) = \mathtt{bv}(A) \cup \mathtt{bv}(B) \\
\mathtt{fv}(\lambda_{\mathcal{V}} A.B) = (\mathtt{fv}(A) \cup \mathtt{fv}(B)) \backslash \mathcal{V} & \mathtt{bv}(\lambda_{\mathcal{V}} A.B) = \mathtt{bv}(A) \cup \mathtt{bv}(B) \cup \mathcal{V}
\end{array}$$

Note that, unlike the λ-calculus, we abstract not only on variables but on terms. Usually, terms are denoted by capital letters. We adopt Barendregt's variable name convention [2], i.e., free and bound variables have different names. This can be fulfilled by renaming bound variables. As usual, we identify terms modulo α-equivalence.

Hedges are finite (possible empty) sequences of terms. For readability, we put in brackets if they have more than one element, e.g., $\langle A, B\rangle$. For the empty hedge we use $\langle\rangle$. We use letter h to denote hedge.

The notions of free and bound variables are extend to hedges in a natural way: $\mathtt{fv}(\langle A_1, \ldots, A_n\rangle) = \cup_{i=1}^{n}\mathtt{fv}(A_i)$ and $\mathtt{bv}(\langle A_1, \ldots, A_n\rangle) = \cup_{i=1}^{n}\mathtt{bv}(A_i)$.

A *substitution* σ is a mapping from variables to hedges such that all but finitely many variables are mapped to themselves. If $x_1, \ldots, x_n$, $n \geqslant 0$ are all the variables for which $\sigma(x_i) \neq x_i$, then we write σ in the form of the finite set of pairs $\{x_1 \mapsto \sigma(x_1), \ldots, x_n \mapsto \sigma(x_n)\}$. The sets $Dom(\sigma) = \{x_1, \ldots, x_n\}$ and $Ran(\sigma) = \{\sigma(x_1), \ldots, \sigma(x_n)\}$ are called the *domain* and the *range* of σ, respectively. The set $Var(\sigma)$ is defined as $Var(\sigma) = Dom(\sigma) \cup \mathtt{fv}(Ran(\sigma))$. The Greek letters $\sigma, \rho, \varphi, \vartheta$ are used to denote substitutions.

The *restriction* of a substitution σ to a set of variables V, denoted $\sigma|_V$, is defined as $\sigma|_V(x) = \sigma(x)$ if $x \in V$, and $\sigma|_V(x) = x$ otherwise.

The *application* of a substitution φ to a term D replaces each *free* occurrence of a variable v in D with $\varphi(v)$. It is defined inductively:

$$
\begin{array}{ll}
x\sigma = \sigma(x), \text{ if } x \in Dom(\sigma). & (A\,B)\sigma = A\sigma\,B\sigma, \text{ if } B \notin \mathcal{X}. \\
x\sigma = x, \text{ if } x \notin Dom(\sigma). & (A\,x)\sigma = A\sigma\,B_1 \cdots B_n, \\
f\sigma = f. & \qquad \text{if } \sigma(x) = \langle B_1, \ldots, B_n\rangle, n \geqslant 0. \\
(\lambda_{\mathcal{V}} A.B)\sigma = \lambda_{\mathcal{V}} A\sigma.B\sigma. & (A\,x)\sigma = A\sigma\,x, \text{ if } x \notin Dom(\sigma).
\end{array}
$$

In the abstraction, it is assumed that $Var(\varphi) \cap \mathtt{bv}(\lambda_{\mathcal{V}} A.B) = \varnothing$. This can be achieved by properly renaming the bound variables. Hence, the equality here is α-equivalence.

The application of a substitution φ to a hedge $\langle s_1, \ldots, s_n\rangle$ is defined as $\langle s_1, \ldots, s_n\rangle\varphi = \langle s_1\varphi, \ldots, s_n\varphi\rangle$, where we write $\langle s_1\varphi, \ldots, s_{i-1}\varphi, t_1, \ldots, t_m, s_{i+1}\varphi, \ldots, s_n\varphi\rangle$ when $s_i\varphi = \langle t_1, \ldots, t_m\rangle$.

Evaluation in the pattern calculus is defined by a binary relation β_{p} on terms. It defines the way how pattern-abstractions are applied. The relation is parametrized by a *pattern matching function* $\mathcal{S}ol$, which takes as parameters two terms A, B and set of variables $\mathcal{V}$ and computes a finite set of substitutions. We denote it by $\mathcal{S}ol(A \lessdot_{\mathcal{V}} B)$. β_{p} is written in the form of a reduction rule:

$$\beta_{\mathsf{p}}: \quad (\lambda_{\mathcal{V}} A.B)\,C \to B\sigma, \text{ where } \sigma \in \mathcal{S}ol(A \lessdot_{\mathcal{V}} C) \text{ and } B\sigma \text{ is a term.}$$

The condition "$B\sigma$ is a term" is important to make sure that terms are reduced to terms, not to arbitrary hedges. A reducible expression, or *redex,* is any expression to which this rule applies. A binary relation of compatibility $\to_R$ on hedges is defined with the help of the following inference rules:

$$\frac{A \to_R A'}{AB \to_R A'B} \qquad \frac{A \to_R A'}{BA \to_R BA'} \qquad \frac{B \to_R B'}{\lambda_{\mathcal{V}} A.B \to_R \lambda_{\mathcal{V}} A.B'}$$

$$\frac{A \to_R A'}{\lambda_{\mathcal{V}} A.B \to_R \lambda_{\mathcal{V}} A'.B} \qquad \frac{A \to_R B}{\langle h_1, A, h_2\rangle \to_R \langle h_1, B, h_2\rangle}$$

In what follows, $\to_{\beta_C}$ denotes the compatible closure of the β_{p} relation and $\twoheadrightarrow_{\beta_C}$ denotes the reflexive and transitive closure of $\to_{\beta_C}$. The definition of $\twoheadrightarrow_{\beta_C}$ is extended to substitutions having the same domain by setting $\varphi \twoheadrightarrow_{\beta_C} \varphi'$ if for all $x \in Dom(\varphi) = Dom(\varphi')$, we have $x\varphi \twoheadrightarrow_{\beta_C} x\varphi'$.

2.1.1 Typed Terms.

Let $\mathbb{A}$ be a nonempty set of type atoms. The set of *types* over $\mathbb{A}$, denoted $\mathbb{T}_{\mathbb{A}}$ or simply $\mathbb{T}$, is defined inductively with the help of the type constructors $\to$ and $*$: $\alpha \in \mathbb{A} \Rightarrow \alpha \in \mathbb{T}$, $\theta_1, \theta_2 \in \mathbb{T} \Rightarrow (\theta_1 \to \theta_2) \in \mathbb{T}$, and $\theta_1, \theta_2 \in \mathbb{T} \Rightarrow (\theta_1^* \to \theta_2) \in \mathbb{T}$.

The set of *star types* (over $\mathbb{A}$), denoted $\mathbb{T}^{\star}_{\mathbb{A}}$ or simply $\mathbb{T}^*$, is defined inductively as $\theta \in \mathbb{T} \Rightarrow \theta \in \mathbb{T}^*$ and $\theta \in \mathbb{T} \Rightarrow \theta^* \in \mathbb{T}^*$. Note that if a type does not contain a star type, then it is a standard simple type.

We define yet another set of types, which we call *fixed arity types* and denote by $\mathbb{F}$. It is the smallest set with the properties $\alpha \in \mathbb{A} \Rightarrow \alpha \in \mathbb{F}$ and $\theta \in \mathbb{T}, \varphi \in \mathbb{F} \Rightarrow (\theta \to \varphi) \in \mathbb{F}$. Each type in $\mathbb{F}$ has the form $\theta_1 \to (\cdots \to (\theta_n \to \alpha)\cdots)$. The set of *starred fixed arity types* is denoted by $\mathbb{F}^*$. We have $\mathbb{F} \subset \mathbb{T}$ and $\mathbb{F}^* \subset \mathbb{T}^*$.

The letter α will be used to denote elements from $\mathbb{A}$, the letters θ, δ for elements of $\mathbb{T}$, Θ for elements of $\mathbb{T}^*$, φ for elements of $\mathbb{F}$, and Φ for elements of $\mathbb{F}^*$. As usual, $\Theta_1 \to \cdots \to \Theta_n \to \theta$ stands for $(\Theta_1 \to (\Theta_2 \to \cdots \to (\Theta_n \to \theta)\cdots))$.

The *subtyping relation* is the preorder generated by the relation $\leqslant$ defined as:

$$\theta_1^* \to \theta_2 \leqslant \theta_2 \qquad \theta_1^* \to \theta_2 \leqslant \theta_1^* \to \theta_1^* \to \theta_2 \qquad \theta \leqslant \theta^*$$

$$\theta_1^* \leqslant \theta_2^* \text{ if } \theta_1 \leqslant \theta_2 \qquad \Theta_1 \to \theta_1 \leqslant \Theta_2 \to \theta_2 \text{ if } \Theta_2 \leqslant \Theta_1 \text{ and } \theta_1 \leqslant \theta_2$$

We denote the subtyping relation with $\leqslant$ as well. The following lemma characterizes fixed arity types in $\mathbb{T}$ with respect to $\leqslant$:

Lemma 1. *For all types $\theta \in \mathbb{T}$, there exists $\varphi \in \mathbb{F}$ such that $\theta \leqslant \varphi$.*

Proof. By structural induction on θ. □

Corollary 1. *For all types* $\Theta \in \mathbb{T}^*$*, there exists* $\Phi \in \mathbb{F}^*$ *such that* $\Theta \preccurlyeq \Phi$*.*

The next lemma states that $\mathbb{F}$ is an upper set in the preorder $\mathbb{T}$:

Lemma 2. *For all types* $\varphi \in \mathbb{F}$ *and* $\theta \in \mathbb{T}$*, if* $\varphi \preccurlyeq \theta$*, then* $\theta \in \mathbb{F}$*.*

Proof. By structural induction on φ. □

Corollary 2. *For all types* $\Phi \in \mathbb{F}^*$ *and* $\Theta \in \mathbb{T}^*$*, if* $\Phi \preccurlyeq \Theta$*, then* $\Theta \in \mathbb{F}^*$*.*

We assume that each $f \in \mathcal{F}$ has the unique associated type, $type(f) \in \mathbb{T}$. A *type assignment statement* is an expression of the form $x : \Phi$ or $A : \Theta$ with $A \in \mathcal{T} \backslash \mathcal{X}$, $\Phi \in \mathbb{F}^*$, $\Theta \in \mathbb{T}^*$. The types Φ, Θ are the *predicate* and x, A are the *subject* of the statement. A *declaration* is a statement whose subject is a variable. A *basis* Γ is a set of declarations with distinct variables as subjects. By $Subject(\Gamma)$ we denote the set of variables that are subjects of the declarations in Γ: $Subject(\Gamma) = \{x \mid x : \Phi \in \Gamma\}$. Note that the variables are assigned fixed arity types or starred fixed arity types, while constants may have an arbitrary associated type from $\mathbb{T}$. A statement $A : \Theta$ is *derivable* from a basis Γ, written $\Gamma \vdash A : \Theta$, if $\Gamma \vdash A : \Theta$ can be produced by the following rules:

$$\frac{}{\Gamma, x : \Phi \vdash x : \Phi}(\text{VAR}) \qquad \frac{}{\Gamma \vdash f : type(f)}(\text{FUN})$$

$$\frac{\Gamma \vdash A : \Theta \to \theta \quad \Gamma \vdash B : \Theta}{\Gamma \vdash AB : \theta}(\text{APP}) \qquad \frac{\Gamma \vdash A : \Theta_1 \quad \Theta_1 \preccurlyeq \Theta_2}{\Gamma \vdash A : \Theta_2}(\text{SUB})$$

$$\frac{\Gamma, \Delta \vdash A : \theta_1 \qquad \Gamma, \Delta \vdash B : \theta_2 \qquad Subject(\Delta) = \mathcal{V}}{\Gamma \vdash \lambda_{\mathcal{V}} A.B : \theta_1 \to \theta_2}(\text{ABS})$$

The type assignment statement and the derivability relation extend to hedges:

$$\frac{\Gamma \vdash A_1 : \Theta_1, \ldots, \Gamma \vdash A_n : \Theta_n \qquad \Theta_1 \preccurlyeq \Theta, \ldots, \Theta_n \preccurlyeq \Theta}{\Gamma \vdash \langle A_1, \ldots, A_n \rangle : \Theta}(\text{HED})$$

From this definition, by Corollary 2 we have the lemma:

Lemma 3. *If* $\Gamma \vdash x : \Theta$*, then* $\Theta \in \mathbb{F}^*$*.*

Example 1. Let $type(f) = \alpha_1 \to \alpha_1$, $type(g) = \alpha_2^* \to \alpha_1^* \to \alpha_2$, $type(a) = \alpha_1$, $type(b) = \alpha_2$, and $\Gamma = \{x : \alpha_1, y : \alpha_2^*, z : \alpha_1 \to \alpha_2\}$. Then some examples of derivable statements are

- $\Gamma \vdash x : \alpha_1$, $\Gamma \vdash x : \alpha_1^*$, and $\Gamma \vdash y : \alpha_2^*$.

- $\Gamma \vdash z : (\alpha_2^* \to \alpha_1) \to \alpha_2$ and $\Gamma \vdash z : (\alpha_2^* \to \alpha_2^* \to \alpha_1) \to \alpha_2$.
- $\Gamma \vdash (g\, b) : \alpha_2^* \to \alpha_1^* \to \alpha_2$, $\Gamma \vdash (g\, b) : \alpha_1^* \to \alpha_2$, and $\Gamma \vdash (g\, b) : \alpha_2$.
- $\Gamma \vdash (g\, (f\, a)) : \alpha_1^* \to \alpha_2$ and $\Gamma \vdash (g\, (f\, a)) : \alpha_2$.
- $\Gamma \vdash \langle y, b, (g\, b), (g\, (f\, a))\rangle : \alpha_2^*$.

We say that $\langle A_1, \ldots, A_n\rangle$ is a Γ-*typed hedge* iff there exists $\Theta \in \mathbb{T}^*$ such that $\Gamma \vdash \langle A_1, \ldots, A_n\rangle : \Theta$. Respectively, A is a Γ-*typed term* iff there exists $\theta \in \mathbb{T}$ such that $\Gamma \vdash A : \theta$. Under this definition, every Γ-typed term is also a Γ-typed hedge, but not vice versa. For instance, if $x : \varphi^\star \in \Gamma$, then x is a Γ-typed hedge, but not a Γ-typed term.

A *typed hedge* (resp. *typed term*) is a Γ-typed hedge (term) for some Γ. We use the letter h for typed hedges and the letters M, N, P, Q, W for typed terms.

Given a basis Γ, we define a Γ-*typed substitution* σ_Γ as a substitution from Γ-typed variables to Γ-typed hedges such that types are preserved. Type preservation means that for each variable x there exist types Φ and Θ with $\Theta \leqslant \Phi$ such that $x : \Phi \in \Gamma$ and $\Gamma \vdash \sigma_\Gamma(x) : \Theta$. The subscript Γ from σ_Γ is sometimes omitted, if it is clear from the context.

Note that the application of a Γ-substitution σ to a Γ-typed hedge maps Γ-typed hedges to Γ-typed hedges. Also, Γ-typed terms are mapped to Γ-typed terms (and not to arbitrary Γ-typed hedges).

Example 2. Let f, g, a, b, and Γ be defined as in Example 1. Let also $M = \lambda_{\{x\}}(f\, x).(g\, y\, x)$ and $\sigma = \{x \mapsto (f\, a), y \mapsto \langle b, (g\, b), (g\, (f\, a))\rangle\}$. Then we have $M\sigma = \lambda_{\{x\}}(fx).(g\, b\, (g\, b)\, (g\, (f\, a))\, x)$.

2.1.2 Reduction.

For a given Γ, a Γ-typed version of the pattern matching function $\mathcal{S}ol$ takes as parameters two Γ-typed terms P, Q and Γ-typed set of variables $\mathcal{V}$ and computes a finite set of Γ-typed substitutions. We denote it by $\mathcal{S}ol(P \lessdot^\Gamma_\mathcal{V} Q)$, dropping Γ when it does not cause a confusion. Then for Γ-typed terms P, N, Q and a Γ-typed substitution σ, β_p can be written as

$$\beta_\mathsf{p}: \quad (\lambda_\mathcal{V} P.N)\, Q \to N\sigma, \text{ where } \sigma \in \mathcal{S}ol(P \lessdot_\mathcal{V} Q).$$

Note that there is no need to require $N\sigma$ to be a term explicitly, because this property always holds due to the fact that the application of a Γ-typed substitution to a Γ-typed term gives a Γ-typed term.

3 Subject Reduction

The first interesting property of our calculus is subject reduction (SR), which essentially says that the $\twoheadrightarrow_{\beta_C}$ relation preserves types. SR is based on two lemmas: the Generation Lemma and the Substitution Lemma.

Lemma 4 (Generation Lemma)**.** *Let* Γ *be a basis.*

- *If* $\Gamma \vdash x : \Phi$*, then there exists* $\Phi' \leqslant \Phi$ *such that* $(x : \Phi') \in \Gamma$*.*
- *If* $\Gamma \vdash MN : \Theta$*, then there exist* Θ' *and* $\theta \leqslant \Theta$ *such that* $\Gamma \vdash M : \Theta' \to \theta$ *and* $\Gamma \vdash N : \Theta'$*.*
- *If* $\Gamma \vdash \lambda_{\mathcal{V}} P.N : \Theta$*, then there exist* θ_1, θ_2*, and* Δ *such that* $\theta_1 \to \theta_2 \leqslant \Theta$*,* $\mathit{Subject}(\Delta) = \mathcal{V}$*,* $\Gamma, \Delta \vdash P : \theta_1$ *and* $\Gamma, \Delta \vdash N : \theta_2$*.*
- *If* $\Gamma \vdash \langle M_1, \ldots, M_n \rangle : \Theta$*, then there exist* $\Theta_1 \leqslant \Theta, \ldots, \Theta_n \leqslant \Theta$ *such that* $\Gamma \vdash M_1 : \Theta_1, \ldots, \Gamma \vdash M_n : \Theta_n$*.*

Proof. By induction on the length of the type derivation. □

Lemma 5 (Substitution Lemma)**.** *Let* Γ *be a basis and* σ *be a* Γ*-typed substitution. If* $\Gamma \vdash M : \theta$*, then there exists* $\delta \leqslant \theta$ *such that* $\Gamma \vdash M\sigma : \delta$*.*

Proof. By structural induction on M. When $M = x$, either $x\sigma = x$ and $\delta = \theta$, or $x\sigma \neq x$ and by Lemma 4, there exists $\Phi \leqslant \theta$ such that $x : \Phi \in \Gamma$. Since σ is Γ-typed, there exists $\Phi' \leqslant \Phi$ such that $\Gamma \vdash x\sigma : \Phi'$ and we can take $\delta = \Phi' \leqslant \theta$.

The proof is easy for $M = f$ and $M = \lambda_{\mathcal{V}} P.N$. We only consider the case $M = M_1 x$. For $M = M_1 M_2$ with $M_2 \notin \mathcal{X}$ the proof is similar.

Let $M = M_1 x$. by Lemma 4, there exist Θ, θ' such that $\Gamma \vdash M_1 : \Theta \to \theta'$, $\Gamma \vdash x : \Theta$, and $\theta' \leqslant \theta$. Then $\Theta \in \mathbb{F}^*$. First, assume $\Theta = \varphi^*$ for some φ. Let $x\sigma = \langle N_1, \ldots, N_n \rangle$, $n \geqslant 0$. Then $(M_1 x)\sigma = ((M_1\sigma)N_1 \cdots N_n)$. By the IH there exists $\delta' \leqslant \varphi^* \to \theta'$ such that $\Gamma \vdash M_1\sigma : \delta'$. If $n = 0$, this already proves the lemma, because from $\delta' \leqslant \varphi^* \to \theta' \leqslant \theta'$, the SUB rule gives $\Gamma \vdash M_1\sigma : \theta'$, and we can take $\delta = \theta'$. If $n > 0$, by the SUB rule, we have $\Gamma \vdash M_1\sigma : \varphi^* \to \theta'$ and, eventually, $\Gamma \vdash M_1\sigma : \varphi^* \to \cdots \to \varphi^* \to \theta'$ for n-fold application. Since $\Gamma \vdash \langle N_1, \ldots, N_n \rangle : \varphi^*$, by HED, there exist $\Theta_1, \ldots, \Theta_n$ such that $\Gamma \vdash N_i : \Theta_i \leqslant \varphi^*$ for all $1 \leqslant i \leqslant n$. Then by SUB we have $\Gamma \vdash N_i : \varphi^*$ for all $1 \leqslant i \leqslant n$. Applying APP n-times, we get $\Gamma \vdash ((M_1\sigma)N_1 \ldots N_n) : \theta'$ and we can take $\delta = \theta' \leqslant \theta$.

Now assume $\Theta = \varphi$ for some φ. Then $\Gamma \vdash x\sigma : \varphi$. By the IH, $\Gamma \vdash M_1\sigma : \delta'$ and $\delta' \leqslant \varphi \to \theta'$. By SUB, $\Gamma \vdash M_1\sigma : \varphi \to \theta'$. Then APP gives $\Gamma \vdash (M_1\sigma)x\sigma : \theta'$ and we take $\delta = \theta' \leqslant \theta$. □

□

Theorem 1 (Subject Reduction)**.** *If* $M_1 \twoheadrightarrow_{\beta_C} M_2$ *and* $\Gamma \vdash M_1 : \Theta$*, then* $\Gamma \vdash M_2 : \Theta$*.*

Proof. We prove $\Gamma \vdash M_2 : \Theta$ from $\Gamma \vdash M_1 : \Theta$ and $M_1 \to_{\beta_C} M_2$. Then the theorem follows by induction on the length of the reduction sequence $M_1 \twoheadrightarrow_{\beta_C} M_2$.

We proceed by induction on the derivation of $\Gamma \vdash M_1 : \Theta$. When $\Gamma \vdash M_1 : \Theta$ is an axiom, then M_1 is either x or f and it can not be reduced by $\to_{\beta_C}$. Hence, the theorem follows trivially, since $M_1 \to_{\beta_C} M_2$ is not possible. When $\Gamma \vdash M_1 : \Theta$ is $\Gamma \vdash N_1 N_2 : \Theta$, then by Lemma 4 there exist Θ' and $\theta \preccurlyeq \Theta$ such that $\Gamma \vdash N_1 : \Theta' \to \theta$ and $\Gamma \vdash N_2 : \Theta'$. We have the following cases:

- $M_2 = N_1' N_2$ with $N_1 \to_{\beta_C} N_1'$, or $M_2 = N_1 N_2'$ with $N_2 \to_{\beta_C} N_2'$. In these cases we apply the IH to get $\Gamma \vdash M_2 : \theta$. Then SUB gives $\Gamma \vdash M_2 : \Theta$.

- $N_1 = \lambda_{\mathcal{V}} P.Q$ and $M_1 = (\lambda_{\mathcal{V}} P.Q) N_2 \to_{\beta_C} M_2 = Q\sigma$ where $\sigma \in \mathcal{S}ol(P \lessdot_{\mathcal{V}} N_2)$. σ is a Γ, Δ-based substitution, where $Subject(\Delta) = \mathcal{V}$. By Lemma 4, there exist Θ' and $\theta \preccurlyeq \Theta$ such that $\Gamma \vdash (\lambda_{\mathcal{V}} P.Q) : \Theta' \to \theta$ and $\Gamma \vdash N_2 : \Theta'$. Again, by Lemma 4 there exist θ_1, θ_2 such that $\theta_1 \to \theta_2 \preccurlyeq \Theta' \to \theta$, $\Gamma, \Delta \vdash P : \theta_1$, and $\Gamma, \Delta \vdash Q : \theta_2$. By the subtyping rules, we get $\Theta' \preccurlyeq \theta_1$ and $\theta_2 \preccurlyeq \theta$. By Lemma 5, we have $\Gamma, \Delta \vdash Q\sigma : \delta$ with $\delta \preccurlyeq \theta_2$. By $\mathbf{C}_0$, on the one hand we have $Dom(\sigma) = \mathcal{V}$ and on the other hand we have $Ran(\sigma) \cap \mathcal{V} = \varnothing$, hence we conclude $\mathtt{fv}(Q\sigma) \cap \mathcal{V} = \varnothing$. Since $\mathcal{V} = Subject(\Delta)$, from $\Gamma, \Delta \vdash Q\sigma : \delta$ we get $\Gamma \vdash Q\sigma : \delta$ with $\delta \preccurlyeq \theta_2$. We also know $\theta_2 \preccurlyeq \theta \preccurlyeq \Theta$. Hence, by SUB we conclude $\Gamma \vdash Q\sigma : \Theta$.

When $\Gamma \vdash M_1 : \Theta$ is $\Gamma \vdash \lambda_{\mathcal{V}} P.Q : \Theta$, then by Lemma 4 there exist $\theta_1 \to \theta_2 \preccurlyeq \Theta$ and Δ such that $Subject(\Delta) = \mathcal{V}$, $\Gamma, \Delta \vdash P : \theta_1$, and $\Gamma, \Delta \vdash Q : \theta_2$. If $M_2 = \lambda_{\mathcal{V}} P'.Q$ with $P \to_{\beta_C} P'$ or $M_2 = \lambda_{\mathcal{V}} P.Q'$ with $Q \to_{\beta_C} Q'$, then by the IH and ABS we get $\Gamma \vdash M_2 : \theta_1 \to \theta_2$. Finally, SUB gives $\Gamma \vdash M_2 : \Theta$. □ □

Example 3. If variables were permitted to have arbitrary types instead of fixed arity types or starred fixed arity types, then SR would not hold: Assume $(x : \alpha^* \to \alpha) \in \Gamma$ and $type(a) = \alpha$. Then we have $\Gamma \vdash x : \alpha \to \alpha$, $\Gamma \vdash x : \alpha$, $\Gamma \vdash \lambda_{\{x\}} x.(x\,x) : \alpha \to \alpha$ and, finally, $\Gamma \vdash (\lambda_{\{x\}} x.(x\,x) a) : \alpha$. However, $(a\,a)$, that is obtained by reducing $(\lambda_{\{x\}} x.(x\,x)\,a)$, is not typeable anymore.

Acknowledgments

This research has been partially supported by the Shota Rustaveli National Science Foundation of Georgia under the grant FR-19-18557, and by the Shota Rustaveli National Science Foundation of Georgia and Turkish Scientic and Technological Research Council joint grant 04/03.

References

1. Alves, S., Dundua, B., Florido, M., and Kutsia, T. Pattern-based calculi with finitary matching. *Log. J. IGPL 26*, 2 (2018), 203–243.

2. Barendregt, H. *The Lambda Calculus: Its Syntax and Semantics*. North-Holland, 1984. Revised edition.

3. Barthe, G., Cirstea, H., Kirchner, C., and Liquori, L. Pure patterns type systems. In *POPL* (2003), A. Aiken and G. Morrisett, Eds., ACM, pp. 250–261.

4. Benzaken, V., Castagna, G., and Frisch, A. CDuce: an XML-centric general-purpose language. In *ICFP* (2003), C. Runciman and O. Shivers, Eds., ACM, pp. 51–63.

5. Cirstea, H., and Kirchner, C. The rewriting calculus - parts I and II. *Logic Journal of the IGPL 9*, 3 (2001).

6. Cirstea, H., Kirchner, C., and Liquori, L. Rewriting calculus with(out) types. *Electr. Notes Theor. Comput. Sci. 71* (2002), 3–19.

7. Coelho, J., and Florido, M. Xcentric: A logic-programming language for xml processing. In *PLAN-X* (2007), pp. 93–94.

8. Dundua, B., Kutsia, T., and Marin, M. Strategies in Pρlog. In *WRS* (2009), M. Fernández, Ed., vol. 15 of *EPTCS*, pp. 32–43.

9. Dundua, B., Kutsia, T., and Reisenberger-Hagmayer, K. An overview of pρlog. In *Practical Aspects of Declarative Languages - 19th International Symposium, PADL 2017, Paris, France, January 16-17, 2017, Proceedings* (2017), Y. Lierler and W. Taha, Eds., vol. 10137 of *Lecture Notes in Computer Science*, Springer, pp. 34–49.

10. Hosoya, H., and Pierce, B. C. Xduce: A statically typed xml processing language. *ACM Trans. Internet Techn. 3*, 2 (2003), 117–148.

11. Jay, B. *Pattern Calculus*. Springer, 2009.

12. Jay, C. B., and Kesner, D. Pure pattern calculus. In *ESOP* (2006), P. Sestoft, Ed., vol. 3924 of *LNCS*, Springer, pp. 100–114.

13. Jay, C. B., and Kesner, D. First-class patterns. *J. Funct. Program. 19*, 2 (2009), 191–225.

14. Kesner, D., Puel, L., and Tannen, V. A typed pattern calculus. *Inf. Comput. 124*, 1 (1996), 32–61.

15. Klop, J. W., van Oostrom, V., and de Vrijer, R. C. Lambda calculus with patterns. *Theor. Comput. Sci. 398*, 1-3 (2008), 16–31.

16. Peyton Jones, S. L., and Wadler, P. *The Implementation of Functional Programming Languages*. Prentice Hall, 1987, ch. 4: Structured Types and the Semantics of Pattern Matching.

17. Pkhakadze, S. A n. bourbaki type general theory and the properties of contracting symbols and corresponding contracted forms. *Georgian Mathematical Journal 6*, 2 (1999), 179–190.

18. Sh, P. Some problems of the notation theory. In *Proceedings of I. Vekua Institute of Applied Mathematics of Tbilisi State University, Tbilisi* (1977).

19. Sulzmann, M., and Lu, K. Z. M. XHaskell - adding regular expression types to Haskell. In *IFL* (2007), O. Chitil, Z. Horváth, and V. Zsók, Eds., vol. 5083 of *LNCS*, Springer, pp. 75–92.

IN THE EUROPEAN UNION WITH THE GEORGIAN AND ABKHAZIAN LANGUAGES — AIMS, PROBLEMS, RESULTS, AND RECOMMENDATIONS OF THE COMPLETE TECHNOLOGICAL SUPPORT OF THE GEORGIAN AND ABKHAZIAN LANGUAGES

Konstantine Pkhakadze Merab Chikvinidze Giorgi Chichua
Shalva Malidze David Kurtskhalia Constantine Demurchev
Nodar Okroshiashvili Beso Mikaberidze

Center for Georgian Language Technology, Georgian Technical University
Kostava str. 77, 0160 Tbilisi, Georgia.
gllc.ge@gmail.com

Abstract

Together with Meta-Net's publications "Europe's Languages in the Digital Age" and "Strategic Research Agenda for Multilingual Europe 2020" this paper is mainly based on the long-term projects "Technological Alphabet of the Georgian Language" and "Plan-Program for the Complete technology support of the Abkhazian Language" of the center for Georgian language technology of the Georgian technical university. Namely, by this center, On May 17, 2019, a report "In the European Union with Georgian and Abkhazian languages – Aims and Problems of Complete Technology Support of Georgian and Abkhazian Languages" was presented in the center for innovation and high technologies of the Georgian national academy of sciences. Thus, in the paper, which is an extend publication version of the above-mentioned report: A reality of high-level danger of digital extinction of the Georgian and Abkhazian languages is proved; Aims, problems and results of complete technology support of the Georgian and Abkhazian languages are overviewed; Recommendations, which have been supported by # 53 protocol from 2019 on 17 May of the center for innovation and high technologies of the Georgian national academy of sciences, are reported.

1 Introduction

On 27 October 2017, on the Abkhazian Language Day, during government meeting, the former prime minister of Georgia Giorgi Kvirikashvili "congratulated all and, especially, Abkhazian brothers and sisters on the Abkhazian

Language Day. The prime minister considers that state program for protection and development of Abkhazian language will create a solid foundation in the confidence building process. The program provides for a variety of activities. Naturally, a relevant budget will be allocated and I am convinced that the society will be actively engaged in the implementation of this program. I believe that our honest and comprehensive approach towards Abkhazian language and culture will be pivotal in terms of restoring the burnt bridges, which are crucial for our unity."[1]

Thus, the above quoted is a very clear confirmations of the right attitude of the Georgian state towards the Abkhazian language. However, if we take into account article 37 of the "organic law of Georgia on official language"[2] according to which "unified program of the official language" aimed at the protection and development of the Georgian state languages – of the Georgian and Abkhazian languages should has been acted from the February 14, 2016, but, unfortunately, it is not acting till today.[3]

The statements of the president of Georgia Mrs. Salome Zurabishvili, which are quoted below, are also very clear confirmations of the right attitude of the Georgian state not only towards the Abkhazian and Georgian languages, but, in general, towards the Caucasian languages too:

1. "Today, as never before, the Abkhazian language and identity need protection."[4]

2. "Georgia's second state language, Abkhazian, is under huge threat today. The State of Georgia, its Constitution protects the Abkhazian language, but it is disappearing. Therefore, as President of Georgia, I feel a special responsibility. In all my official visits, I discuss the grave situation the Russification policy has created for the Abkhazian people, their culture and language."[5]

[1] See at the address http://gov.ge/index.php?lang_id=ENG&sec_id=463&info_id=62686 – State Program for Protection and Development of Abkhazian Language to be Launched.

[2] We will very shortly overview this very important law below.

[3] For clarity, we emphasize that we have no doubt about the sincerity of the former prime minister of Georgia in relation to the Abkhazian language. Thus, this convinces us that he was not completely informed by his surroundings on this issue, since on May 27, 2016 we provided extensive and comprehensive information on this issue in an official letter, where we asked him to accelerate an activation of the "unified program of the official language" in a timely manner.

[4] See at the address https://www.president.gov.ge/eng/prezidenti/inauguracia.aspx – 18 December 2018, President Salome Zourabichvili's Inauguration Day.

[5] See at the address https://www.president.gov.ge/eng/pressamsakhuri/siakhleebi/saqartvelos-prezidentma-saqartvelos-moqalaqeebs-da.aspx – 26 May, 2019, Independence Day.

3. "Defense of the state language is my constitutional obligation and is both a high responsibility and a great pride. Language is the state's core. A unified state cannot exist without a language. Language unifies society and its second characteristic is that it carries a nation's culture. Culture cannot exist without language. Today's globalization and technologies create large challenges for languages."[6]

4. "The Georgian scientific community requests the senior officials to establish a coordination center for Georgian and Caucasus studies in order to preserve Caucasian languages and cultures in the age of globalization."[7]

Together with above quoted completely right views of the President of Georgia Mrs. Salome Zurabishvili, it becomes more unclear why "unified program of the official language" aimed at protecting of the Georgian and Abkhazian languages is not acting till today.[8]

In addition, the fact, that Georgian state has really very responsible and right attitude to the Georgian and Abkhazian languages, is also proven by the fact, that the organic law of Georgia on official Language was approved finally on 22 July 2015.[9] For more clarity, below, we have quoted the 37th and 4th articles of this law:

"article 4 – Status of the official language: 1. In accordance with Article 8 of the Constitution of Georgia, the official Language of Georgia is Georgian and the official language of the Autonomous Republic of Abkhazia is Georgian as well as Abkhazian. 2. The State ensures the protection of the constitutional status of the official language throughout the whole territory of Georgia. 3. The State continually facilitates the preservation and exploration of the Kartvelian Languages and dialects, which is one of the most significant preconditions for maintaining viability and resilience of the official language."

[6]See at the address https://www.president.gov.ge/eng/pressamsakhuri/siakhleebi/%E2%80%8Bsalome-zurabishvili-mshobliuri-ena-chveni-identob.aspx – May 31, 2019, at the conference "State language for civic integration and human capital development in Georgia".

[7]See at the address https://www.president.gov.ge/eng/pressamsakhuri/siakhleebi/salome-zurabishvili-unda-gaaqtiurdes-mushaoba-saqa.aspx – 12 October, 2019, President Salome Zourabichvili on Intensify Scientific Studies of Georgia and Caucasus and Provide Objective Information to International Community.

[8]If the "Unified program of the official language" was be acting, we would be having today very well grounding for the implementation of very important researches for the protecting and developing all other Caucasian languages too, need of which is fully rightly emphasized the President of Georgia Mrs. Salome Zurabishvili.

[9]See at the address https://matsne.gov.ge/en/document/view/2931198?publication=3

"article 37 – Unified Program of the Official Language: 1. The Department of Official Language shall submit the unified programme of the official language to the Government of Georgia for approval upon the recommendation of the Experts Committee of the official language; 2. Public authorities designated by the Government of Georgia shall ensure implementation of the unified programme of the official language. 3. The unified programme of the official language aims to: a) meet maximum requirements of persons who are willing to learn the official language; prepare methodological and educational resources in order to teach the official language and to raise linguistic culture; teach the official language in accordance with contemporary requirements, and to introduce a bilingual teaching programme for the groups of linguistic minorities; b) conduct researches in the field of the structure, history and functional features of the official language in a consistent manner; ensure a unified lexicography of the contemporary Georgian language and provide a full set of norms and terminology standards of the Georgian literary language; c) provide complete technological support of the official language; create contemporary and comprehensive digital database of linguistic data (texts); develop search, analytical-operative and translation software. 4. The unified programme of the official language is funded by the State Budget of Georgia. 5. The State shall enhance the attraction of investments for the implementation of the unified programme of the official language."

All of the above, as whole, makes it very clear that in the rapidly forthcoming digital age the Georgian state take care for protection and development of its state languages – Georgian and Abkhazian languages. This, on the one hand, is obvious truth, but, on the other hand, it must be strongly emphasized that, today, 2020 is almost over and, nevertheless, the most important article of the "organic law of Georgia on official language" – the article 37 is still out of force.[10] Thus, the main reason why we are publishing

[10] According to our information, the most active supporters of the organic law of Georgia on the official language together with Georgian national academy of sciences was former prime minister of Georgia Irakli Garibashvili (20.11.2013 - 29.12.2015). In additions to this it must be underlined, that we addressed an official letter to the former prime minister (from October 2012 to November 2013) of Georgia, Mr. Bidzina Ivanishvili (together with us this letter was signed by Lasha Abzianidze and Aleksandre Maskharashvili), in March 2013, where we asserted the reality of the danger of digital extinction of the Georgian language in the rapidly forthcoming digital age and, accordingly, we asked for timely initiation of the necessary measures to protect from this danger the state languages of Georgia. Also, at that time, our friend, famous political figure Pridon Sakvarelidze was member of the parliament committee of education, science and culture (17.11.2012 - 26.10.2015) and in the March of the same 2013 we had a long conversation with him about the dangers of digital death of languages. During this conversation we introduced him the alarming results of META-NET two-years research "Europe's Languages in the Digital Age" as well as the alarming low level of the technological support of

another paper dedicated to the aims of defence of Georgian and Abkhazian languages from the danger of the digital extinctions is that, unfortunately, "unified program of the official language", which was to be put "into force from 14 February 2016", not only does not act, but, moreover, it are not elaborated yet.[11]

But, it should be noted also, that in 2019, with financial and institutional support of Shota Rustaveli Georgian national science foundation, Georgian national academy of sciences and Georgian technical university the First Tbilisi International Summer School "Logic, Language, Artificial Intelligence" was organized, dedicated to the memory of Shalva Pkhakadze.[12] The summer school was aimed to provide the complete technological support of Georgian and Abkhazian languages. – This gives hope, that defense of Georgian state languages is really important for Georgian state and that in the near future according to the article 37 of the "organic law of Georgia on official language" the "unified program of the official language" will be elaborated. – In any case, this is our goal and we will try everything to make it happen in the very near future.

Georgian language. As a result all of these, on July 8, 2013, with the initiative of Fridon Sakvarelidze and with the support of the former chairman of the parliament of Georgia David Usupashvili (2012 - 2016) and former chairperson of the education, science and culture committee of parliament of Georgia Ivane Kiguradze (2012 - 2016) parliamentary conference "Georgian Language – Challenges of the 21st Century" was been organized, where we made a report "The Technological Alphabet of The Georgian Language – The One of The Most Important Georgian Challenge of The XXI Century". We say all this in order to make it clear to the readers that the first parliamentary and governmental team of the "Georgian Dream – Democratic Georgia" were working very intensive and fruitful for the aims of defense of Georgian state languages from the danger of digital extinction, which, we repeat, is clearly evidenced by the organic law of Georgia on the official Language finally approved on July 22, 2015 (this fact together the fact that Saakashvili's government completely blocked the ongoing processes for the protection and development of the Georgian language, is an important feature of the European and national nature of both the old and the new government of Georgia).

[11] By this we want to say that, the second parliamentary and governmental team of the "Georgian Dream – Democratic Georgia" (this refers to the period from 30 December 2015 to 2 September 2019, when prime ministers of Georgia were Giorgi Kvirikashvili and Mamuka Bakhtadze (On September 7, 2018, we addressed an official letter to him with the aims of defence state languages of Georgia from danger of digital extinction, but there was not any result!)) had severely hampered the rapid progress made in previous years towards the aims of defense of Georgian state languages from the danger of digital extinction, because of which increased clearly the quality of dangers of digital extinction, which are faced Georgian and Abkhazian languages today.

[12] See at the address `https://geoanbani.com/TbiLLAI/`

2 Aims and Problems of the Complete Technological Support of the Georgian and Abkhazian languages

2.1 The Aims of the Complete Technological Support of the Georgian and Abkhazian languages

The long-term projects "Technological Alphabet of the Georgian Language" (launched from 2012) and "Plan-Program for the Complete Technology Support of the Abkhazian Language" (launched from 2015) are aimed at providing complete technological support of the Georgian and Abkhazian languages. At the same time, a complete technological support of the Georgian and Abkhazian languages implies, first, to construct Georgian and Abkhazian technological alphabets, in other words, this implies to build computer systems almost completely knowing Georgian and Abkhazian languages and, second, this implies to equip these Georgian and Abkhazian computer systems with multilingual translation abilities [1–4].

Our above-mentioned aims are fully in line with the very important European aims of the META-NET,[13] which were first voiced in the guideline style publication "Strategic Research Agenda for Multilingual Europe 2020"[14] of the same center. Truly, in this for us very important publication, to the question "What are language technologies?" is given the following answer: "Language technologies are technologies for automatically analysing and generating the most complex information medium in our world, human language, in both its spoken and written forms (as well as sign language)." In addition to this, to prove that our aims are really fully in line with the very important European aims of the meta-network, we quote here from the paper "Strategic Research Agenda for Multilingual Europe 2020", according to which the META-NET has aimed to the "multilingual European society, in which all citizens can use any service, access all knowledge, enjoy all media and control any technology in their mother tongues. This will be a world in which written and spoken communication is not hindered anymore by language barriers and in which even specialised high-quality translation will be affordable".[15]

[13] Multilingual Europe Technology Alliance Network of Excellence, shortly META-NET (http://www.meta-net.eu/), is dedicated to building the technological foundations of a multilingual European information society.

[14] "Strategic Research Agenda for Multilingual Europe 2020" (http://www.meta-net.eu/sra) presented by the META Technology Council is a guideline style publication edited by Georg Rehm and Hans Uszkoreit. It was published on December 1, 2012 and is the result of a more then two years discussion between hundreds of experts from research and industry.

[15] On the matter also noteworthy is the publication "Language technologies for a mul-

Thus, it is clear from the above, that the aims of the long-term projects "Technological Alphabet of the Georgian Language"[16] and "Plan-Program for the Complete Technology Support of the Abkhazian Language"[17] are in fully line with the most important European aims of the META-NET, as well as that these long-term projects are aimed to reach the perfect solutions of the problems of construction artificial intelligence systems for the Georgian and Abkhazian languages, which are the problems of the top difficulties laid in interdisciplinary area of logic, language, and artificial intelligence.

---

tilingual Europe" (https://langsci-press.org/catalog/book/106), published in 2018 and edited by Georg Rehm, Felix Sasaki, Daniel Stein and Andreas Witt, in which to the same question – "What are language technologies?" is given the following answer: "In the next IT revolution computers will master our languages. Just as they already understand measurements and formats for dates and times, the operating systems of tomorrow will know human languages." It should be emphasized here that the exact same vision for the computers of the future was first expressed in the publication of the META-NET "Strategic Research Agenda for Multilingual Europe 2020". – Also, we must emphasized here that the exact same vision for the computers of the future was declared by us firstly in 2001 [4, 5].

[16]The long-term project "Technological Alphabet of the Georgian Language" was elaborated on the basis of the aims [5–8] and results [9–16] of state priority program "Free and Complete Programming Inclusion of a Computer in the Georgian Natural Language System" of the Ivane Javakhishvili Tbilisi state university, which, in turn, was elaborated by K. Pkhakadze in 2000-2002 years with the same aims i.e. with aims of construction computer system almost completely knowing Georgian language. This state priority program was leading by K. Pkhakadze in 2002-2007 years at the Ilia Vekua institute of applied mathematics, in 2008-2009 years at the open institute for the Georgian language, logic and computer and in 2009-2010 years at the Saint Andrew the first-called Georgian university of the patriarchate of Georgia. – To make it more clear to the reader the aims and content of this state priority program developed almost twenty years ago, we are quoting from Pkhakadze's article "Globalization, Georgian Language and State Priority Program – Free and Complete Programming Inclusion of a Computer in the Georgian Natural Language System" published in 2005: "the aims outlined here are to equip computers with the general lingual-logical thinking skills. This means to build thinking machines, that in future will be full-scale intellectual partners of humans. Thus, the Georgian thinking machine is the necessity without which our state, linguistic and cultural existence in the forthcoming age of thinking machines may be even more marginalized, than we can imagine today" [7].

[17]It should also be noted that since 2015, the long-term project "Plan-Program for the Complete Technology Support of the Abkhazian Language" has been separated from the long-term project "Technological Alphabet of the Georgian Language" as an independent project. We have also to emphasize that it is impossible to construct a computer system, which will have almost complete knowledge of the Abkhazian language without the direct involvement in this researches specialists who are naturally familiar with the Abkhazian language! – Therefore, at this stage, one of the main aims of the "Plan-Program for the Complete Technology Support of the Abkhazian Language" is to ensure the voluminous involvement of Abkhazian sciencyists in the research planned by the project. Accordingly, if today there are not sufficient amount of such scientists among Abkhazians, then we consider it as our duty to speed up their upbringing and training [3, 4, 17].

At the same time, and it is also very clear, that the solutions of these problems have especially high importance for cultural future of the Georgian and Abkhazian languages, because of only in the case of perfect solutions of these problems it will be possible to defense cultural future of Georgian and Abkhazian languages in rapidly forthcoming digital age and, accordingly, to enter in the European union, in more general, in the future cultural digital world with technologically completely supported Georgian and Abkhazian languages.[18] – It is very clear that in the digital age – in the age of such computers, which will have an almost complete knowledge of languages, those languages, which do not have such lingual computers, will be completely marginalized from the global cultural processes!

Thus, to summarize, it can be said, that complete technological support of the Georgian and Abkhazian languages – in other words – the long-term projects "Technological Alphabet of the Georgian Language" and "Plan-Program for the Complete Technology Support of the Abkhazian Language" aimed, first, to protect Georgian and Abkhazian languages from danger of the digital extinction and, second, to enter Georgian state in the European Union with completely supported Georgian and Abkhazian languages, that, we think, in whole, is necessary for the preservation of Georgian and Abkhazian cultural identity in today very rapidly forthcoming digital age.

[18] In addition to full support in democratic and anti-occupation movements one of the main reasons why we are so keen in the enter in the European Union is an already very strictly defined language policy of European Union, which is based on eco-linguistic views and directly aimed on the defense of the different languages and identities. – Below are given two fragments from "Strategic Research Agenda for the Multilingual Europe 2020" done by META technology council as short confirmations of the already mentioned: 1. "Everybody must have the chance to communicate efficiently in the enlarged EU. This does not only affect those who already are multilingual but also those who are monolingual or linguistically less skilled. The media, new technologies and human and automatic translation services can bring the increasing variety of languages and cultures in the EU closer to citizens and provide the means to cross language barriers. They can also play an important role to reduce those barriers and allow citizens, companies and national administrations to exploit the opportunities of the single market and the globalising economy. Faced with the globalising online economy and ever-increasing information in all imaginable languages, it is important that citizens access and use information and services across national and language barriers, through the internet and mobile devices. Information and communication technologies (ICT) need to be language-aware and promote content creation in multiple languages." 2. "The Council of the European Union ... encourage[s] the development of language technologies, in particular in the field of translation and interpretation, firstly by promoting cooperation between the Commission, the Member States, local authorities, research bodies and industry, and secondly by ensuring convergence between research programmes, the identification of areas of application and the deployment of the technologies across all EU languages."

2.2 The Problems of the Complete Technological Support of the Georgian and Abkhazian Languages

Problems of complete technological support of the Georgian and Abkhazian languages are mainly conditioned, first, by difficulties of constructing technological alphabets of the Georgian and Abkhazian languages, in other words, by difficulties of constructing computer systems almost completely knowing the Georgian and Abkhazian languages, second, by the absence of a unified Georgian research center, which will be targeted to solve these very difficult and very important problems, and, third, by low quality of the support of Georgian and Abkhazian languages with the language technologies and resources. Because of that, that from these three the first two are without doubt [17–20], below, we will try to prove the third.[19]

Thus, below, on the basis of the very alarming results of the META-NET's two-year research "Europe's Languages in the Digital Age"[20] and, also, META-NET's press-release "At Least 21 European Languages in Danger of Digital Extinction – Good News and Bad News on the European Day of Languages"[21] we will try to prove, that Georgian state languages – Georgian and Abkhazian languages are in the high level danger of digital extinction in the rapidly forthcoming digital age.

On 20 September, 2012 – on the European Day of Languages, on the basis of the alarming results of the study "Europe's Languages in the Digital Age" was published a very alarming press-release "At least 21 European languages in danger of digital extinction – good news and bad news on

[19] It is also clear and, accordingly, it is not controversial, that for countries with such a small population as our country, the most optimal and effective way to solve such problems is the timely formation of the unified research center targeted to solve these very difficult and very important problems. For more clarity, the fact, that, for one side, in July 22, 2015 the Georgia law on official language was finally approved, and, for the second side, till today i.e. during four years, there is not been elaborated the "unified program of the official language", makes very clear that in our country the wrong attitudes prevail over in these very important issues during past four years.

[20] A all European study conducted by META-NET in 2010-2012, "European Languages in the Digital Age" (`http://www.meta-net.eu/whitepapers/overview`), also known as the White Paper Series, was published in 32 volumes and includes 31 European language. These languages are the Basque, Bulgarian, Catalan, Croatian, Czech, Danish, Dutch, English, Estonian, Finnish, French, Galician, German, Greek, Hungarian, Icelandic, Irish, Italian, Latvian, Lithuanian, Maltese, Norwegian (bokmål), Norwegian (nynorsk), Polish, Portuguese, Romanian, Serbian, Slovak, Slovene, Spanish, Swedish, and Welsh languages.

[21] META-NET Press Release "At Least 21 European Languages in Danger of Digital Extinction – Good News and Bad News on the European Day of Languages" (`http://www.meta-net.eu/whitepapers/press-release`) reports on the state of 30 European languages with respect to Language Technology and explains the most urgent risks and chances. The series covers all official EU Member State languages and several other languages spoken in Europe.

the European day of languages", according to which "a new study by Europe's leading Language Technology experts warns", that "most European languages are unlikely to survive in the digital age".

This two-years "study, prepared by more than 200 experts and documented in 30 volumes of the meta-net white paper series, assessed language technology support for each language in four different areas: automatic translation, speech interaction, text analysis and the availability of language resources. A total of 21 of the 30 languages (70%) were placed in the lowest category, "support is weak or non-existent" for at least one area by the experts. Several languages, for example, Icelandic, Latvian, Lithuanian and Maltese, receive this lowest score in all four areas. On the other end of the spectrum, while no language was considered to have "excellent support", only English was assessed as having "good support", followed by languages such as Dutch, French, German, Italian and Spanish with "moderate support". Languages such as Basque, Bulgarian, Catalan, Greek, Hungarian and Polish exhibit "fragmentary support", placing them also in the set of high-risk languages."

The results of this META-NET's two-year study, "European Languages in the Digital Age" are summarized in the below given Table 1, which we took from a for us very important paper "Strategic Research Agenda for Multilingual Europe 2020".[22]

To make more visible the results presented by the table 1, we rated "excellent support" at 5 points, "good support" at 4 points, "moderate support" at 3 points, "fragmentary support" at 2 points, "weak or no support" at 1 point and, after, on base of table 1, we generated below given table 2 named by us as "State of language technology support for Georgian, Abkhazian and 31 European languages in four different areas in points and percents".

In this table we assessed language technology support for the Georgian and Abkhazian languages on base of results of the long-term projects "Technological Alphabet of Georgian Language" and "Plan-Program for Complete Technology Support of the Abkhazian Language" [1, 3, 5], which are overviewed in next paragraph of this paper.

Thus, the table 2 built by us below together with the previous assessments of the META-NET makes it clear that European languages, which

[22]However, we have added the Welsh language to the table on the basis of the results of paper "The Welsh Language in the Digital Age" (http://www.meta-net.eu/whitepapers/volumes/welsh) prepared by Jeremy Evas. At the same time, we underline, that according to the Europeans themselves, this table shows alarming differences in technology support between languages. English is ahead from the other languages. A few larger languages have a good technology support, when smaller or very small languages have substantial gaps. Many languages lack basic technologies and language resources. Others have these ones, but lack basic semantic methods and tools. Therefore, it is clear, that most of European languages are in danger of digital extinctions.

Support	Excellent	Good	Moderate	Fragmentary	Weak/none
Machine Translation		English	French, Spanish	Catalan, Dutch, German, Hungarian, Italian, Polish, Romanian	Basque, Bulgarian, Croatian, Czech, Danish, Estonian, Finnish, Galician, Greek, Icelandic, Irish, Latvian, Lithuanian, Maltese, Norwegian, Portuguese, Serbian, Slovak, Slovene, Swedish, Welsh
Speech		English	Czech, Dutch, Finnish, French, German, Italian, Portuguese, Spanish	Basque, Bulgarian, Catalan, Danish, Estonian, Galician, Greek, Hungarian, Irish, Norwegian, Polish, Serbian, Slovak, Slovene, Swedish	Croatian, Icelandic, Latvian, Lithuanian, Maltese, Romanian, Welsh
Text Analytics		English	Dutch, French, German, Italian, Spanish	Basque, Bulgarian, Catalan, Czech, Danish, Finnish, Galician, Greek, Hungarian, Norwegian, Polish, Portuguese, Romanian, Slovak, Slovene, Swedish	Croatian, Estonian, Icelandic, Irish, Latvian, Lithuanian, Maltese, Serbian, Welsh
Language Resources		English	Czech, Dutch, French, German, Hungarian, Italian, Polish, Spanish, Swedish	Basque, Bulgarian, Catalan, Croatian, Danish, Estonian, Finnish, Galician, Greek, Norwegian, Portuguese, Romanian, Serbian, Slovak, Slovene	Icelandic, Irish, Latvian, Lithuanian, Maltese

Table 1: State of language technology support for 30 European languages in four different areas.

summary point are less to 10 point (see below given table) i.e. Hungarian, Polish, Czech, Galician, Catalan, Portuguese, Swedish, Romanian, Slovenian, Slovak, Greek, Finnish, Danish, Bulgarian, Basque, Norwegian, Estonian, Serbian, Irish Croatian, Icelandic, Latvian, Lithuanian, Maltese, and Welsh languages are appreciated as languages under danger of digital extinction.

This fact together of the fact, that language technology support of the Georgian and Abkhazian language is less than technology support of almost any aforementioned European languages, which are in danger of digital extinction, proves clearly a reality of the high-level danger of digital extinction

Language	**speech interaction**	**automatic translation**	**text analysis**	**language resources**	**summary points / percent**
1. English	4	4	4	4	16(-4) = 80% (-20%)
2. French	3	3	3	3	12(-8) = 60% (-40%)
3. Spain	3	3	3	3	12 (-8) = 60% (-40%)
4. German	3	2	3	3	11 (-9) = 55% (-45%)
5. Dutch	3	2	3	3	11 (-9) = 55% (-45%)
6. Italian	3	2	3	3	11 (-9) = 55% (-45%)
7. Hungarian	2	2	2	3	9 (-11) = 45% (-55%)
8. Polish	2	2	2	3	9 (-11) = 45% (-55%)
9. Czech	3	1	2	3	9 (-11) = 45% (-55%)
10. Galician	2	1	3	3	9 (-11) = 45% (-55%)
11. Catalan	2	2	2	2	8 (-12) = 40% (-60%)
12. Portuguese	3	1	2	2	8 (-12) = 40% (-60%)
13. Swedish	2	1	2	3	8 (-12) = 40% (-60%)
14. Romanian	1	2	2	2	7 (-13) = 35% (-55%)
15. Slovenian	2	1	2	2	7 (-13) = 35% (-55%)
16. Slovak	2	1	2	2	7 (-13) = 35% (-55%)
17. Greek	2	1	2	2	7 (-13) = 35% (-55%)
18. Finnish	2	1	2	2	7 (-13) = 35% (-55%)
19. Danish	2	1	2	2	7 (-13) = 35% (-55%)
20. Bulgarian	2	1	2	2	7 (-13) = 35% (-55%)
21. Basque	2	1	2	2	7 (-13) = 35% (-55%)
22. Norwegian	2	1	2	2	7 (-13) = 35% (-55%)
23. Estonian	2	1	1	2	6 (-14) = 30% (-70%)
24. Serbian	2	1	1	2	6 (-14) = 30% (-70%)
25. Irish	2	1	1	1	5 (-15) = 25% (-75%)
26. Croatian	1	1	1	2	5 (-15) = 25% (-75%)
27. Icelandic	1	1	1	1	4 (-16) = 20% (-80%)
28. Latvian	1	1	1	1	4 (-16) = 20% (-80%)
29. Lithuanian	1	1	1	1	4 (-16) = 20% (-80%)
30. Maltese	1	1	1	1	4 (-16) = 20% (-80%)
31. Welsh	1	1	1	1	4 (-16) = 20% (-80%)
32. Georgian	0.75	0.75	0.75	0.75	3 (-17) = 15% (-85%)
33. Abkhazian	0.25	0.25	0.25	0.25	1 (-19) = 5% (-95%)

Table 2: State of language technology support for Georgian, Abkhazian and 31 European languages in four different areas in points and percents

of the technologically insufficiently supported Georgian and Abkhazian languages in the rapidly forthcoming digital age.

In order to make the reality of the threats already presented by us regarding the Georgian and Abkhazian languages even more credible, below are quoted from "Strategic research agenda for multilingual Europe 2020". – Only quick view at them will make clear a very high responsible attitude of the Europeans themselves are experiencing about fate of their languages in very rapidly forthcoming digital age:

Denmark: "If we have the ambition to use the Danish language in the technological universe of the future, an effort must be made now to maintain

and further develop the knowledge and expertise that we already have. Otherwise we run the risk that only people who are fluent in English will profit from the new generations of web, mobile and robot technology which are up and coming." – Sabine Kirchmeier-Andersen (Director of the Danish Language Council).

Portugal: "Language technology is of utmost importance for the consolidation of Portuguese as a language of global communication in the information society." – Pedro Passos Coelho (Prime Minister of Portugal).

Czech Republic: "META-NET brings a significant contribution to the technological support for languages of Europe and as such will play an indispensable role in the development of multilingual European culture and society." – Ivan Wilhelm (Deputy Minister for Education, Youth and Sport).

Greece: "Further support to language technologies safeguards the presence of Greek language and culture in the digital environment". – George Babiniotis (Minister of Education, Lifelong Learning and Religious Affairs).

European Commission: "Having worked on automatic media analysis for many years and in tens of languages, we are painfully aware of the lack of text analysis tools and resources in most languages. META-NET's analysis is very accurate. Language Technology is a key enabling ingredient for future generations of IT. Languages for which no tools and resources will exist soon will not participate in the next major technological developments." – Ralf Steinberger (Joint Research Centre, IPSC - GlobeSec - Open-Source Text Information Mining and Analysis, Ispra, Italy).

Estonia: "If we do not implement the development plan for language technology or do not cooperate with other countries in the same direction, in the future Estonian will be marginalised in information society." – Development Plan of the Estonian Language 2011–2017.

France: "META-NET provides an invaluable contribution to the development of a genuine European strategy in support to multilingualism, based on existing technologies while encouraging the development of new innovative technologies." – Xavier North (Délégué Général à la Langue Française et aux Langues de France).

Malta: "The technology support for the Maltese language should serve our language to be continuously cultivated, used and placed on the same level as other languages." – Dolores Cristina (Minister for Education and Employment).

Lithuania: "Conserving Lithuanian for future generations is a responsibility of the whole of the European Union. How we proceed with developing information technology will pretty much determine the future of the Lithuanian language." – Andrius Kubilius (Prime Minister of the Republic of Lithuania).

Ireland: "Language technology is no longer a luxury for most European

languages – it is now essential to their survival as viable means of expression across the whole range of areas from business to the arts, and this is as much the case for Irish as any other European language." – Ferdie Mac an Fhailigh (CEO, Foras na Gaeilge).

Slovenia: "It is imperative that language technologies for Slovene are developed systematically if we want Slovene to flourish also in the future digital world." - Danilo Türk (President of the Republic of Slovenia).

Iceland: "Language technology is an essential tool in a variety of linguistic research, and supports the official Icelandic policy of promoting the national language in all aspects of communication." - Guðrún Kvaran (Chair of the Icelandic Language Council).

Netherlands and Flanders (Belgium): "It remains extremely important that citizens can use their native language in all circumstances, including when they deal with modern ICT and leisure devices. But usually English speaking people are the first to benefit from such an evolution. Not only does this pose a danger of reducing the overall functionality of a language (and an impoverishment of an entire culture), but also it threatens those groups in society that do not master the universal language. Therefore, R&D programmes that support the local language are needed. Also, in the future, the Dutch Language Union will continue to emphasise this issue." – Linde van den Bosch (General Secretary of the Dutch Language Union, 2004-2012).

Poland: "Language technologies are more and more present in our everyday life. For their presence to be rational and functional, for it to serve the needs of the economy, as well as the social and cultural life well, further large-scale work in this area is needed." - Michał Kleiber (President of the Polish Academy of Sciences).

Luxembourg: "This is a European challenge of enormous importance!" – Roman Jansen-Winkeln (CTO, Belingoo Media Group)

Germany: "Europe's multilingualism and our scientific expertise are the perfect prerequisites for significantly advancing the challenge that language technology poses. META-NET opens up new opportunities for the development of ubiquitous multilingual technologies." – Annette Schavan (Minister of Education and Research)

UK: "The work of META-NET is an important step towards a future in which Language Technology will be all around us, allowing us to collaborate, conduct business and share knowledge with friends and colleagues, whether or not we speak the same language." – David Willets (Minister of State for Universities and Science, Department for Business, Innovation and Skills).

Latvia: "For such small languages like Latvian keeping up with the ever increasing pace of time and technological development is crucial. The only way to ensure future existence of our language is to provide its users

with equal opportunities as the users of larger languages enjoy. Therefore, being on the forefront of modern technologies is our opportunity." – Valdis Dombrovskis (Prime Minister of Latvia).[23]

The above passages are texts of 2012 year, and these texts make it clear how Europeans themselves were concerned about the fate of their languages in the digital age in 2012! – This fact together the fact that technological support of the Georgian, even more, of the Abkhazian, lags far behind from the European languages that are in danger of digital extinction, clearly shows the reality of the high-level danger of the digital extinction, which is faced to Georgian, even more, Abkhazian language!

In order to make even more understandable the validity of the above-mentioned opinions of high-ranking European politicians and experts, and also in order to make even more credible the reality of the high-level threats of digital death facing Georgian and Abkhazian languages today, below, we have quoted a short fragments from above already overviewed press-release: 1. "The results of our study are most alarming. The majority of European languages are severely under-resourced and some are almost completely neglected. In this sense, many of our languages are not yet future-proof." – The author of the quote Prof. Hans Uszkoreit, coordinator of META-NET, scientific director of the German Research Center for Artificial Intelligence, editor of the META-NET publication "Europe's Languages in the Digital Age". 2. "There are dramatic differences in language technology support between the various European languages and technology areas. The gap between big and small languages still keeps widening. We have to make sure that we equip all smaller and under-resourced languages with the needed base technologies, otherwise these languages are doomed to digital extinction." – The author of the quote Dr. Georg Rehm, researcher of the German Research Center for Artificial Intelligence, co-editor of the META-NET publication "Europe's Languages in the Digital Age".

These expert views of Hans Ushkorait and Georg Rehm, a well-known

[23] Here, in the interests of the question, we decided to once again bring an excerpt from the speech of the President of Georgia, Salome Zurabishvili, on 31 May, 2019, at the conference "State Language for Civic Integration and Human Capital Development in Georgia", where she says: "Defense of the state language is my constitutional obligation and is both a high responsibility and a great pride. Language is the state's core. A unified state cannot exist without a language. Language unifies society and its second characteristic is that it carries a nation's culture. Culture cannot exist without language. Today's globalization and technologies create large challenges for languages." – on this background, or with this completely correct and acceptable position of one of the highest political figures of the Georgian state, it is completely incomprehensible the fact that "Unified program of the official language", which are aimed for the protection of the Georgian state languages – Georgian and Abkhazian languages is not working, but, even more, it is not yet elaborated!

European experts in artificial intelligence and language technoligies together with all above already said, we think, is quite enough for any Abkhazian and Georgian who thinks about the fate of their languages to deeply understand that the low technological suport of Georgian, even more, Abkhazian language is a very significant challenge, which, in turn, fully substantiates, that "State Program for the Protection and Development of Georgian and Abkhazian Languages", in other words, "unified program of the official language", which is already legalized by the organic law of Georgia on official language has the high national importance and, accordingly, there is urgent need to develop it in very near future!

3 Results of the Complete Technological Support of the Georgian and Abkhazian languages

As we have already mentioned, from technological support point of view, the Georgian and Abkhazian languages alarmingly lag behind compared to almost any of those European languages, which, according to the well-known research "Europe's Languages in the Digital Age" done by Meta-Net with financial support of Euro Structures are under the danger of digital extinction in the rapidly forthcoming digital age.[24]

This clearly indicates the necessity of the overcoming this lagging as soon as it is possible. Thus, the long-term projects "Technological Alphabet of the Georgian Language" and "Plan-Program for Complete Technology Support of the Abkhazian Language" of the Scientific-Educational Center for the Georgian Language Technology[25] are aimed at reducing this alarming lagging and, therefore, these projects are directly engaged with the aims of saving the Georgian and Abkhazian languages from the danger of digital extinction and, also, with the aim of joining the European Union, in more general, in the future cultural world with the technologically completely supported Georgian and Abkhazian languages, in other words, with the Georgian and Abkhazian technological alphabets – with intellectual computer systems knowing Georgian and Abkhazian languages almost completely and perfectly.

[24] Compared to Georgian and Abkhazian languages, other Kartvelian and Caucasian languages are even more backward from that European languages, which are under the danger of digital extinction. Therefore, below, we have also briefly reviewed the technological systems developed by us for Mingrelian, Chechen, Kabardian and Lezgian languages, which system are unique in the sense that there are not any other such type systems for these languages.

[25] The Center for Georgian Language Technology acting with the aims of defence Georgian state languages was established at Georgian Technical University in 29 December, 2010 with direct support of the rector of university Academician Archil Prangishvili.

The fact, that language technology support of the Georgian and Abkhazian languages is really much less than technology support of European languages, is also proved by below listed language technology systems, which are been created within above shortly overviewed long-term projects.

Namely, these language technology systems are created on the basis of that new methods of Pkhakadze-Chikvinidze, Pkhakadze-Chichua, Pkhakadze-Malidze, which are elaborated on the basis of Logical Grammar of the Georgian Language K. Pkhakadze and, also, on the base of different today well-known tools and platforms within already successfully completed and till now ongoing subprojects of the long-term projects "The Technological Alphabet of the Georgian Language" and "The Plan-Program for the Complete technology support of the Abkhazian Language" [1–4,21–27]. They are:

1. FR/362/4-105/12 project "Foundations of Logical Grammar of Georgian Language and its Application in Information Technology" (Project supervisor – K. Pkhakadze; Project was funded by the Shota Rustaveli National Scientific Foundation);

2. AR_№048-13 project "Internet Versions of a Number of Developable (Learnable) Systems Necessary for Creating The Technological Alphabet of the Georgian Language" (Project supervisor – K. Pkhakadze; Project was funded by the Georgian Technical University);

3. DO/308/4-105/14 project "In the European Union with the Georgian Language, i.e., the Doctoral Thesis – Georgian Grammar Checker (Analyzer)" (PhD student – M. Chikvinidze, Scientific Supervisor – K. Pkhakadze; PhD was jointly funded by the Shota Rustaveli National Scientific Foundation and Georgian Technical University);

4. DO/305/4-105/14 project "In the European Union with the Georgian Language, i.e., the Doctoral Thesis – Georgian Speech Synthesis and Recognition" (PhD student – G. Chichua, Scientific Supervisor – K. Pkhakadze; PhD was jointly funded by the Shota Rustaveli National Scientific Foundation and Georgian Technical University);

5. AR_122/4-105/14 Project "One More Step Towards Georgian Talking Self-Developing Intellectual Corpus" (Project supervisor – K. Pkhakadze; Project was jointly funded by the Shota Rustaveli National Scientific Foundation and Georgian Technical University);

6. PHDF-18-1228 project "In the European Union with Georgian and Abkhazian Languages, i.e. the Doctoral Thesis – Elaboration of the New Developing Tools and Methods of the Georgian Smart Corpus

and Improvement of Already Existing Ones" (PhD student – Sh. Malidze, Scientific Supervisor – K. Pkhakadze; PhD was funded by the Shota Rustaveli National Scientific Foundation);

7. PhD thesis "Methods and Tools for the Automatic Intellectual Classification of Georgian Texts" (PhD student – C. Demurchev, Scientific Supervisor – K. Pkhakadze);

8. PhD thesis "Georgian-Mathematical Automated Multilingual Semantic Translator" (PhD student – N. Okroshiashvili, Scientific Supervisor – K. Pkhakadze).

9. PhD thesis "Formalism and Applications of Georgian Language Processing by Machine Learning Methods" (PhD student – B. Mikaberidze, Scientific Supervisor – K. Pkhakadze).

Thus, below is listed Georgian and Abkhazian language technology systems, which are cretead within above listed sub-projects [2,3,17–27]:

In the area of speech interaction:

1. Trial-applied Georgian spoken browser. – This system, which is inbuilt in the Georgian universal smart corpus, is unique in the sense that there is not any other such type Georgian system;
2. Trial-applied voice managed Georgian reader. – This system is unique in the sense that there is not any other such type Georgian system;
3. Trial voice managed Abkhazian reader. – This system is unique in the sense that there is not any other such type Georgian system;
4. Trial-applied mobile and internet versions of the Georgian spoken assistant for speech disorder persons. – These systems are unique in the sense that there are not any other such type Georgian system;
5. Trial versions of the adapted Georgian Internet, Wikipedia, and Computer. – These systems are unique in the sense that there are not any other such type Georgian systems.

In the area of automatic translation:

1. Trial self-developing Georgian-Mathematical translator. – This system, which is inbuilt in the Georgian universal smart corpus, is unique in the sense that there is not any other such type Georgian system;

2. Trial self-developing two-way Georgian-English-German semantic translator based on the mathematical language. – This system, which is inbuilt in the Georgian universal smart corpus, is unique in the sense that there is not any other such type Georgian system;
3. Trial-applied internet and mobile versions of Georgian multilingual voice lexicon. – These systems, which are based on google translate, are unique in the sense that there are not any other such type Georgian systems;
4. Trial internet and mobile versions of hybrid Georgian multilingual spoken assistant and google translate. – These systems, which are based on google translate, are unique in the sense that there are not any other such type Georgian systems;
5. Trial internet and mobile versions of multilingual textual and voice messages between Georgian smart papers. – These systems, which are inbuilt in the Georgian universal smart corpus and based on google translate, are unique in the sense that there are not any other such type Georgian systems.

In the area of text analysis:

1. Trial-applied versions of the automatic processing texts and websites for automatic creation of the self-developing corpuses. – These systems, which are inbuilt in the Georgian universal smart corpus, are unique in the sense that there are not any other such type systems in Georgia not for Kartvelian not for Caucasian languages;
2. Trial-applied and trial versions of taggers, descriptors and generators for Georgian N, A and V type words. – These systems, which are inbuilt in the Georgian universal smart corpus, are unique in the sense that there are not any other such type Georgian systems;
3. Trial-applied Georgian self-developing orthograph checker. – This system, which is inbuilt in the Georgian universal smart corpus, is unique in the sense that there is not any other such type Georgian system;
4. Trial Abkhazian self-developing orthograph checker. – This system, which is inbuilt in the Georgian universal smart corpus, is unique in the sense that there is not any other such type Abkhazian system;

5. Trial Georgian self-developing syntax checker. – This system, which is inbuilt in the Georgian universal smart corpus, is unique in the sense that there is not any other such type Georgian system;
6. Trial versions of the Georgian text logical analyzer and question answerer. – These systems, which are inbuilt in the Georgian universal smart corpus, are unique in the sense that there are not any other such type Georgian systems;
7. Trial versions for automatic generating and testing of the Georgian logical tasks and analogies with voice managed tools. – These systems are unique in the sense that there are not any other such type Georgian systems;
8. Trial-applied version of the Georgian printed and scanned texts recognition. – This system is not unique, because that there is existed some such type trial-applied systems for Georgian language constructed on the base of different well-known platforms, but our approach is differed from all others, because of our system is constructed independently from other existed platforms;
9. Trial version of the Abkhazian printed and scanned texts recognition. – According to our information, this system is unique in the sense that there is not any other such type Abkhazian system;
10. Trial system of Georgian smart paper and smart Journal. – These systems, which are inbuilt in the Georgian universal smart corpus, are unique in the sense that there are not any other such type Georgian systems;
11. The trial-applied system for Georgian texts classification. – This system, which is constructing as inbuilt tool in the Georgian universal smart corpus for the aims of logical, intellectual, and thematical classifications of the texst of the corpus, is unique in the sense that there is not any other such type Georgian system.

In the area of language resources:

1. Trial-applied version of Georgian universal (i.e. multilingual and multimodal) smart (i.e. self-diveloping, interactive, intellectual, communicative) corpus (`http://corpus.ge/`). – This corpus,[26] which is first self-developing, interactive, intelectual, multilingual

[26] In spite of the fact, that by now this corpus is only step toward the complete Georgian Universal Smart Corpus, It is the largest Georgian corpus (For today, and today is 19 Juli, 2020 it contains 331 782 593 word-tokens, among them 6 104 732 are different ones).

and multimodal Georgian corpus, is unique in the sense that there is not any other such type corpus for Georgian;[27]

2. Trial-applied version of the self-developing Georgian corpus, which is inbuilt in the Georgian universal smart corpus as separately cluster. – It develops its own volume and intellectual (ortografic, syntactic, logic, semantic, statistic) abilities itself. Thus, this corpus is unique in the sense that there is not any other such type Georgian corpus;

3. Trial version of the self-developing Abkhazian corpus, which is inbuilt in the Georgian universal smart corpus as separately integrated cluster. – It develops itself its own volume and intellectual (ortografic, statistic) abilities. Thus, this corpus is unique in the sense that there no any other such type Georgian corpus;

4. Trial versions of the self-developing Chechen, Kabardian, Lezgian and Mingrelian corpuses. – These corpuses are first self-developing corpuses for these Kartvelian and Caucasian languages. They develop themself their own volumes and intellectual (statistic) abilites. Thus, these corpus are unique in the sense that there are no any other such type Kartvelian and Caucasian corpuses;

5. Trial parallel corpuses of Georgian-English and Georgian-German words. – They can develop themself on the basis of internet sources. Also, they have the opportunity to rate a quality of the translation and in the case of a translation error users may correct the incorrect translations. Thus, these corpuses are uniqe in the sence that there are not any other such type Georgian corpuses;

6. Trial versions of the Georgian and Abkazian titrated speech data. – The corpuses, which are constructed for the aims of improve quality of our Georgian and Abkhazian TTS and STT systems, conatin segmentators and generators for the Georgian and Abkhazian titrated speech data. Thus, these trial corpuses are unique

[27] The deal is that, this Georgian universal smart corpus, from one side, is a corpus, but, at the same time, on the basis of it inbuilt technology systems is equipped with self-developing, interacting, translating, analyzing, and communicating abilities, is also a laboratorial prototype of the united Georgian state internet smart network. We believe, that such type state networks in future – in forthcoming digital age, in other words, in the age of computers, which will have almost complete knowledge of languages, must be property of the all various different states and not only of a private company or only a few different states. For more clarity: For us it is very clear, that the Georgian technological alphabet i.e. the computer system almost completely knowing Georgian language must be property of the Georgian state.

in the sense that there are not any other such type Georgian and Abkhazian corpuses;

7. Trial tools for constructing Chechen, Kabardian, Lezgian and Mingrelian titrated speech data. – These tools are are unique in the sense that there are not any other such type tools for these Kartvelian and Caucasian languages;

8. Foundations of Logical Grammar of Georgian Language. – This is a first mathematical theory – mathematical grammar for the Georgian language. This theory, on the basis of which it is constructed more part from above listed systems, from one side, is based on the Shalva Pkhakadze's Notation Theory, and, from second side, it is based on the natural mathematical specifics of Georgian langauge.

Thus, the above listed unique technologies and resources for the Georgian and Abkhazian languages show us that if we have only trial-applied and trial systems for Georgian and Abkhazian languages, for English as well as for many other European languages are already prepared the same type applied and trial-applied systems with sufficiently high quality! – This is the reason why in the table 2 we have appreciated the technology support of Georgian language by 15%, and of Abkhazian language by 5%. – These facts prove reality of very high-level danger of digital extinction of the Georgian and Abkhazian languages in the rapidly forthcoming digital age in full.

At the same time, the fact that the above listed language technologies and language resources were elaborated by a very small group with a very small funding, argues that it will be possible for our country to provide complete technological support of Georgian and Abkhazian languages in the case of appropriate state care on these issues, which, in our opinion, should be firstly reflected in the timely formation of the "Research Institute for Technological Development and Cultural Defense of Georgian State Languages".

4 Conclusion or Our Main Recommendations for the aims of Complete Technological Support of the Georgian and Abkhazian languages

As it was already mentioned, on May 17, 2019, a report "In the European Union with the Georgian and Abkhazian languages – Aims and Problems of Complete Technology Support of Georgian and Abkhazian Languages" was

presented at the center for innovation and high technologies of the Georgian national academy of sciences by the center for Georgian language technology of the Georgian technical university.

Taking into consideration the aims of defense of the state languages of Georgia – Georgian and Abkhazian languages from danger of digital extinction and, also, taking into consideration the arguments presented in this paper above, by the #53 protocol, the Center for innovation and high technologies of Georgian national academy of science agreed with our recommendations made by us at this presentation fully. Therefore, and it is very natural, that the same recommendations in shorted way is presented below as our main recommendations.

Thus, for the aims of Complete Technological Support of the Georgian and Abkhazian languages we are recommending:

1. The timely formation the "Research Institute for Technological Development and Cultural Defense of Georgian State Languages";

2. The timely define "Technological development and cultural defense of Georgian state languages" as a one of main priority of the Shota Rustaveli National Science Foundation of Georgia;

3. To held the annual conferences "Georgian state languages in the digital age: cultural and technological aspects" twice a year (on April 14 – in Georgian language day, and on October 27 – in Abkhazian language day). – This will allow us constantly observe dynamics and results of processes current with aims of the cultural defense and technological development of the Georgian state languages.

In addition we underline, and it is very clear, that by the taking these above listed recommendations into account, first, it will become very clear that in the forthcoming digital age Georgian state really take care on the protecting the Georgian and Abkhazian languages from the rapidly increasing danger of the digital extinction, and, second, it will also become very clear to Abkhazians, which reside in occupied Abkhazia, that for the Georgian state is a vital important to preserve Abkhazian language and identity. – Thus, and it is very clear also, that all abovementioned in sum will be very important step into the direction of reconstruction of today's broken bridge of the Georgian-Abkhazian relations. – The fact together with all the above-mentioned recommendations should be considered also as the first important step toward the cultural protection and technological development of all other Kartvelian and Caucasian languages, that, we think, is a matter of the common Caucasian commitment of the Georgia.

The last: In the preamble of the "organic law of Georgia on official language" there is stressed the following: "The Georgian language is a historical

and cultural heritage of Georgia and it is essential for its statehood. It is an element of common national identity for all citizens of Georgia. The State of Georgia performs all its functions in this language, supports it and supports the policy for the development and functioning of this language as of the official language of the State". – Thus, here, according to this very important content of the preamble of the "organic law of Georgia on official language", we direct to the Government of Georgia with our main recommendation: For the aim to defense the state languages of Georgia from danger of digital extinction it is vitally necessary immediate elaboration of the "unified program of the official language" with the necessary involvement of Georgian specialists of artificial intelligence and language technologies, which are working as in Georgia, as well as out of Georgia.

And finally, we underline one more time, that a computer, which will have almost complete knowledge of a language, must be property of that national state in confine of which this language is used! – We believe that this is the only path to healthy and peaceful globalization.

References

1. Pkhakadze K., Chichua G., Chikvinidze M., Maskharashvili A. The Technological Alphabet of The Georgian Language – Aims, Methods and Results, *Reports of Enlarged Session of the Seminar of VIAM*, Volume 27, 48-51, 2013.

2. Pkhakadze K., Chikvinidze M., Chichua G., Malidze Sh., Kurtskhalia D., Demurchev C., Oqroshiashvili N. In the European Union with Georgian and Abkhazian languages – Aims and Problems of Complete Technology Support of Georgian and Abkhazian Languages. *Bulletin of the Georgian National Academy of Sciences*, (in publishing). 2020.

3. Malidze Sh., Pkhakadze K., Chikvinidze M., Chichua G., Kurtskhalia D. The final summary of the results of the Phd Project "In the European Union with Georgian and Abkhazian languages, i.e The Doctoral thesis – Elaboration of the New Developing Tools and Methods of the Georgian Smart Corpus and Improvement of Already Existing Ones". *Journal Georgian Language and Logic*, Vol. 12, pp.62-164, 2019-2020.

4. Pkhakadze K. Free and Complete Programming Inclusion of a Computer in Georgian Natural Language System. *Newspaper "Resonance"*, 242-2362, 2001.

5. Pkhakadze K., Ivanishvili M. Toward the formal-logical adequate of the natural Georgian language system. Georgian-European University, Paris, *Journal "Iveria"*, Vol. VI, 129-148. 2001.

6. Pkhakadze K. Globalization, The Georgian Language and The State Priority Program "Free and Complete Programming Inclusion of a Computer in Georgian Natural Language System". *Journal Georgian Language and Logic*, vol. 2, pp.1-11, 2005.

7. Pkhakadze K. TSU State Priority Program "Free and Complete Programming Inclusion of a Computer in Georgian Natural Language System" – Aims, Results and Perspectives. *Main and Additional text-books in Contemporary Mathematical linguistics*, Vol.1, pp.1-18, 2005.

8. Pkhakadze K. On the Linguistic Relation and Logical Declension in the Georgian Language. *Journal Georgian language and Logic*, Vol.1, pp.19-77, 2005.

9. Pkhakadze K., Chichua G., Vashalomidze A., Gabunia K., Abzianidze L., Maskharashvili A., Chikvinidze M. The Grounding Questions of The Mathematical Theory of The Georgian Language and Thinking and Some Subsystems of The 1st Version of the Voice Managed Georgian Intellectual Computer System. *Proceeding of The I International Conference of Chikobava institute of Linguistics on Georgian Language and Modern Technology*, pp.60-66, 2009.

10. Pkhakadze K., Chichua G., Chikvinidze M., Maskharashvili A. The Short Overview of the Aims, Methods, and Results of the Logical Grammar of the Georgian Language. *Reports of Enlarged Session of the Seminar of VIAM*, Volume 26, pp.58-64, 2012

11. Pkhakadze K. The Technological Alphabet of The Georgian Language – The One of The Most Important Georgian Challenge of The XXI Century. *Proceedings of The Parliament Conference The Georgian Language – The Challenge of the 21th Century*, pp.98-105, 2013.

12. Pkhakadze K., Chankvetadze G., Tibua L., Ivanishvili M., Lekiashvili L., Soselia E. About Main Ideas of Direct Formal-Logical Description of the Georgian Language. *Proceedings of VIAM*, Vol. 53, pp.33-40, 2003.

13. Pkhakadze K., Chankvetadze G., Tibua L., Ivanishvili M., Lekiashvili L., Soselia E. About the Main Ideas of the Direct Formal-Logical Description of the Georgian Natural Language System

Through the Examples. *Proceedings of the V Tbilisi Symposium on Language, Logic and Computation*, pp.129-137, 2003.

14. Pkhakadze K. On the Linguistic Relation and Logical Declension in the Georgian Language. *Journal Georgian language and Logic*, Vol.1, pp.19-77, 2005.

15. Pkhakadze K., Abzianidze L., Maskharashvili A. Georgian Language's Theses. *Reports of the Seminar of VIAM*, Vol. 34, pp.108-121, 2008.

16. Pkhakadze K., Chichua G., Abzianidze L., Maskharashvili A. 1-Stage Voice Managed Georgian Intellectual Computer System. *Reports of the Seminar of VIAM*, Vol. 34, 96-107, 2008.

17. Pkhakadze K., Chikvinidze M., Chichua G., Kurtskhalia D., Malidze Sh. The Open Letter to the Georgian Parliament, Government, National Academy of Sciences and Georgian and Abkhazian Society i.e. Key Principles of the Unified program of Complete technological supporting of the official language of Georgia (Georgian, Abkhazian) i.e. In the Future Cultural World with the Completely Supported Georgian and Abkhazian Languages. *Journal Georgian Language and Logic*, Vol.11 (II Part), pp.121-164, 2017-2018.

18. Pkhakadze K., Chikvinidze M., Chichua G., Beriasvili I., Kurtskhalia D. Take Care on the Georgian Language. *Proceedings Of Georgian National Academy of Sciences*, pp.149-158, 2016.

19. Pkhakadze K., Chikvinidze M., Chichua G., Kurtskhalia D. The Open letter to the Georgian Society, Or, Already, It Is Really the Time To take Care for the Georgian Language. *Journal Georgian Language and Logic*, Vol.9, pp.131-158, 2015-2106.

20. Pkhakadze K., Chikvinidze M., Chichua G., Beriasvili I., Kurtskhalia D. A Logical Grammar of Georgian Language: Foundations and Applications – First Part – Preface. *Journal Georgian Language and Logic*, Vol.9, pp.4-130, 2015-2106.

21. Pkhakadze K., Chikvinidze M., Chichua G. The Project "Foundations of Logical Grammar of Georgian Language and Its Application in Information Technology" and Doctoral Themes "Georgian grammar checker (analyzer)" and "Georgian speech synthesis and recognition". *Journal Georgian Language and Logic*, pp.21-36, 2013-2014.

22. Pkhakadze K., Chikvinidze M., Chichua G., Maskharashvili A., Beriasvili I. An Overview of the Trial Version of the Georgian Self-Developing Intellectual Corpus Necessary for Creating Georgian text Analyzer, Speech Processing, and Automatic Translation Ssystems. *Reports of Enlarged Session of the Seminar of VIAM*, Volume 28, pp.70-75, 2014.

23. Pkhakadze K., Chikvinidze M., Chichua G., Beriasvili I., Kurtskhalia D. The Aims and First Results of the Project "One More Step Towards Georgian Talking Self-Developing Intellectual Corpus". *Proceedings of the I international Conference on Language and Modern Technology*, pp.108-112, 2015.

24. Pkhakadze K., Chikvinidze M., Chichua G., Beriasvili I., Kurtskhalia D. In the European Union with Georgian language – the Aims and Basements of the Project "One More Step Towards Georgian Talking Self-Developing Intellectual Corpus". *Reports of Seminar of VIAM*, Volume 29, pp.37-43, 2015.

25. Pkhakadze K., Chikvinidze M., Chichua G., Kurtskhalia D., Malidze Sh. Georgian Universal Smart Corpus and in It Inbuilt Abkhazian, Chechen, Kabardian, Lezgian, and Mingrelian Self-Developing Corpuses – Results, Perspeqtives. *Reports of Enlarged Sessions of the Seminar of VIAM*, Volume 32, pp.55-58, 2018.

26. Pkhakadze K., Chikvinidze M., Chichua G., Kurtskhalia D., Malidze Sh., N. Oqroshiashvili, Demurchev C. The Trial Versions of the New Developing Tools of the Georgian Universal Smart Corpus. *Reports of Enlarged Sessions of the Seminar of VIAM*, Volume 33, pp.50-53, 2019.

27. Pkhakadze K., Chikvinidze M., Chichua G., Beriasvili I., Kurtskhalia D., Malidze Sh. The Georgian Intellectual Web Corpus: Aims, Methods, Recommendation. *Additional Publication of the Journal Georgian Language and Logic*, pp.4-320, 2017.

შალვა ფხაკაძე - ცხოვრებისეული მონაცემების მოკლე ბიოგრაფიული მიმოხილვა

ხ. რუხაია, კ. ფხაკაძე

წინამდებარე ნაშრომი, რომელიც ეძღვნება საქართველოში კლასიკური მათემატიკური ლოგიკის სკოლის ფუძემდებლის, საქართველოს მეცნიერებათა დამსახურებული მოღვაწის, შალვა ფხაკაძის დაბადებიდან 100 წლის საიუბილეო თარიღს, ეყრდნობა ოთარ ჭანკვეტაძის, ხიმურ რუხაიასა და კონსტანტინე ფხაკაძის ადრინდელ პუბლიკაციებს სახელწოდებით "შალვა ფხაკაძე" (თბილისის სახელმწიფო უნივერსიტეტი, ვეკუას სახელობის გამოყენებითი მათემატიკის ინსტიტუტი, 1-50, 1999) და "შალვა ფხაკაძის სამეცნიერო-პედაგოგიური მოღვაწეობის მოკლე მიმოხილვა" (საქართველოს მათემატიკოსთა კავშირის V ყოველწლიური საერთაშორისო კონფერენციის თეზისები, 31-38, 2014). ამრიგად, ნაშრომში მოკლედ არის მიმოხილული შალვა ფხაკაძის ძირითადი ბიოგრაფიული მონაცემები. გარდა ამისა, ნაშრომის დამასრულებელ ნაწილში პირველად არის წარმოდგენილი მისი ზოგადი, სხვა სიტყვებით რომ ვთქვათ, ბუნებრივი სემანტიკური პროგრამა, რომელიც მან მათემატიკის დაფუძნების მიზნით შეიმუშავა. ასევე, აქ მოკლედ არის წარმოდგენილი ახალი თეორიული ხედვა სხვადასხვა ბუნებრივი ენების წარმოშობის შესახებ, რომელიც ძირეულად ეყრდნობა მისეულ აღნიშვნათა თეორიასა და ბუნებრივ სემანტიკურ პროგრამას.

* * *

ზომად სივრცეთა ლოგიკა ბუნებირივი ენების სემანტიკისთვის

ჟან-ფილიპ ბერნარდი, რ. ბლანკი, ა. მასხარაშვილი

ჩვენ განვსაზღვრავთ ზომად სივრცეთა ლოგიკას (LMS) და მოვიყვანთ არგუმენტებს იმის სადემონსტრაციოდ რომ იგი წარმოადგენს ისეთ

ფორმალიზმს, რომელსაც შესწევს ბუნებირვ-ენობრივ ფენომენთა ჯეროვანი გამოხატველობითი უნარი. LMS შთაგონებულია რამდენიმე სხვადასხვა წყაროდან. იგი გადაწყვეტადია (ისევე როგორც აღწერების ლოგიკა); იგი იყენებს სიგმა სივრცეებს (მსგავსად მარტინ-ლიოფის ტიპთა თეორიისა); იგი შეიცავს სივრცის ზომის ცნებას (იხ. [6]) და ასევე შეუძლია გამოხატოს ზომათა შეფარდება, რაც იძლევა იმის საშუალებას რომ გამოვსახოთ ხდომილების ალბათობა LMS მეშვეობით. მიუხედავად იმისა, რომ LMS წარმოადგენს საკმარისად გამომხატველ ფორმალიზმს, ხაზგასასმელია რომ იგი ლაკონურად აღწერადია და ამასთან აქვს ზუსტი სემანტიკური ინტერპრეტაცია ინტეგრალურ აღრიცხვაში. ყველა ამ თვისებიდან გამომდინარე, ჩვენ ვიმედოვნებთ რომ LMS ითამაშებს თავის სათანადო როლს ბუნებრივი ენების სემანტიკის დაფუძნებაში.

* * *

პარალელური შინაარსების ბანკი: რამდენიმე ენის ერთდროული სემანტიკური აღწერის სამუშაო ჩარჩო

ლ. აბზიანიძე, რიკ ვან ნოორდ, ჩ. ვანგ, ი. ბოს

სტატიაში ზოგადად აღწერილია ის იდეები რომლებიც საფუძვლად უდევს პარალელური შინაარსების ბანკს -- სამუშაო ჩარჩო რომლის მიზანიც არის რომ გააადვილოს არაინგლისურენოვანი ტექსტების აღწერა კომპოზიციური სემანტიკით. აღწერა არის ნახევრად ავტომატური და შედგება ლინგვისტური ინფორმაციის შვიდი ფენისგან: დანაწევრება, ნიშნების შემოტანა, სემანტიკური დანიშვნა, სიტყვების მნიშვნელობების დადგენა, სინტაქსური ანალიზი, თემატური როლების მინიჭება და კორეფერირება. შინაარსების ბანკში ახალი ენის ტექსტების დამატება შესაძლებელია იმ პირობით რომ ამ ტექსტებთან ერთად უნდა მოხდეს მისი ინგლისური თარგმანის დამატებაც. ახალი ენის დამატება ამავდროულად წარმოშობს საინტერესო გამოწვევებს იმ ლინგვისტური წინაპირობებისთვის რომლებიც საფუძვლად უდევს პარალელური შინაარსების ბანკს.

* * *

რეკურსიული პროგრამული სქემების ნორმალიზაციის გადაწყვეტადობის მოკლე დამტკიცება

ზ. ხასიდაშვილი

სტატიაში მოყვანილია რეკურსიული პროგრამული სქემების ნორმალიზაციის გადაწყვეტადობის მოკლე და მარტივი დამტკიცება. ამასთანავე, აღნიშნული დამტკიცებიდან მიიღება ალგორითმი, რომელიც ნებისმიერ რეკურსიულ პროგრამულ სქემას ეფექტურად გარდაქმნის დაუყვანად სქემად, სადაც უმოკლესი მანორმალიზებელი დაყვანები ადვილად აიგება.

* * *

უნიფიკაცია α-ეკვივალენტობის მიხედვით მათემატიკურ ასისტენტ-სისტემაში

თ. კუცია

სტატიაში შესწავლილია უნიფიკაცია α-ეკვივალენტობის მიხედვით ისეთ ენაში, სადაც ერთმანეთთან შერწყმულია პერმისიული ნომინალური ტერმები და მომდევრობის ცვლადები. უნიფიკაციის ასეთი ამოცანები სათავეს იღებს დედუქციის პრობლემებიდან, რომელთა გადაჭრაში მათემატიკური ასისტენტ-სისტემა Theorema გამოიყენება. მათი ამოხსნის ალგორითმი, რომელიც ამ სტატიაშია შემოტანილი, ერთმანეთს უთავსებს პერმისიული ნომინალური უნიფიკაციისა და შეზღუდულსიგრძიანი მიმდევრობების უნიფიკაციის ალგორითმებს. იგი ჩერდება, არის კორექტული, მინიმალური და

აკმაყოფილებს სისრულის თვისების შეზღუდულ ვარიანტს. გარდა ამისა, განხილულია უნიფიკაციის პრობლემათა ორი სპეციალური შემთხვევა, რომელთათვისაც მიმდევრობათა სიგრძის შეზღუდვა შეიძლება მოიხსნას. ესენია (1) შეთანადების ფრაგმენტი და (2) ფრაგმენტი, რომელშიც მიმდევრობის ცვლადები ტერმებში ბოლო არგუმენტის პოზიციას იკავებენ. ამ სპეციალური შემთხვევებისთვის შემუშავებულია უნიფიკაციის მინიმალური და სრული ალგორითმები. ყველა აღწერილი ალგორითმი იმპლემენტირებულია და ჩართულია Theorema-ს უნიფიკაციის პაკეტში.

* * *

სინტაქსისა და სემანტიკის ურთიერთმართება: მონტეგიუს გრამატიკიდან დღემდე

ა. მასხარაშვილი

ჩვენ მიმოვიხილავთ ბუნებირვი ენის სინტაქსისა და სემანტიკის ურთიერთმიმართების შესწავლის მიზნებით წარმოებულ იმ კვლევებს, რომლებიც აღმოცენდნენ ლოგიკის წიაღში. ჩვენი მთავარი აქცენტი მონტეგიუს ნაშრომებზეა, რომელიც ლიტერატურაში მონტეგიუს გრამატიკადაა ცნობილი და რომელიც პრიველწყარო მნიშვნელობისად არის ფართოდ აღიარებული ბუნებირივი ენების ლოგიკური შესწავლის მიმართულებით. ასევე განვიხილავთ შედარებით ახალ მიდგომებს, რომლებსაც იდეოლოგიურ საფუძვლად უდევთ მონტეგიუს გრამატიკა, აბსტრაქტული კატეგორიული გრამატიკების მაგალითზე. ეს საშუალებას მოგვცემს მოკლედ აღვწეროთ ხედვები რომელსაც გვთავაზობს მონტეგიუს გრამატიკებზე დაფუძნებული თანამედროვე კვლევები გამოთვლითი ენათმეცნიერების ისეთი კლასიკური პრობლემების გადაწყვეტებში როგორებიცაა ბუნებირივი ენის ტექსტების გენერაცია და ბუნებირივი ენის ტექსტების ანალიზი. კერძოდ, მონახაზის დონეზე აღვწერთ აბსტრაქტული კატეგორიული გრამატიკების გამყენების გზებს დისკურსის (ტექსტის) შინაარსისა და ფორმის ურთიერთმიმართების შესწავლისას.

* * *

ტექსტში ლოგიკური გამომდინარეობის გადაწყვეტა ბუნებრივი ენის თეორემების მამტკიცებლით

ლ. აბზიანიძე

ჩვენ წარმოგიდგენთ ბუნებრივი ენის თეორემების მამტკიცებელს და ვაჩვენებთ თუ როგორ შეიძლება მისი დახმარებით ტექსტში სხვადასხვა ტიპის ლოგიკური გამომდინარეობების დადგენა. თეორემების მამტკიცებელი დაფუძნებულია ბუნებრივი ლოგიკის ტაბლო სისტემაზე რომელიც თავის მხრის ოპერირებს წინადადებების ლოგიკურ ფორმებზე. სტატიაში განხილულია რიგი ლოგიკური გამომდინარეობები და მათი დადგენა მამტკიცებლის საშუალებით. განხილული მაგალითები მოიცავს როგორც სწორად ასევე არასწორად გამოცნობილ ლოგიკურ გამომდინარეობებს. დეტალური ანალიზი გვიჩვენებს რომ მცდარი დამტკიცებები, რომლებიც საკმაოდ მცირე რაოდენობით არის, ძირითადად განპირობებულია სიტყვების არასწორი მნიშვნელობების გამოყენებით ან შეცდომებით რომელიც დაშვებულია ლოგიკური გამომდინარეობების სტანდარტით დადგენილ სწორ პასუხებში. დამტკიცებების ვერ მოძიების მთავარი მიზეზი არის მამტკიცებლში ცოდნის უკმარისობა.

* * *

ვარსკვლავიანი ტიპები: ტიპტთა სისტემა თარგების არღიცხვისთვის

ბ. დუნდუა, მ. რუხაია, ლ. ტიბუა

ნაშრომში განხილულია თარგების აღრიცხვის ფორმალიზმი, რომელიც აერთიანებს ლამბდა აღრიცხვის ფუნქციონალურ მექანიზმს და ვარსკვლავიანტიპიანი ტერმების შეთანადების შესაძლებლობებს. ვარსკვლავიანი ტიპები სპეციფიკაციას უკეთებენ ტერმების სასრულ მიმდევრობებს და იწვევენ არადეტერმინისტულ გამოთვლას, რაც განპირობებულია სასრული შეთანადებით. ჩვენ მოვახდინეთ თარგების აღრიცხვის პარამეტრიზირება შეთანადების აბსტრაქტული ფუნქციით და ვაჩვენეთ, როცა ფუნქცია არის სასრული შეთანადება, მაშინ აღრიცხვას გააჩნია სუბიექტზე დაყვანის თვისება.

* * *

ქართული და აფხაზური ენებით ევროკავშირში - ქართული და აფხაზური ენების სრული ტექნოლოგიური უზრუნველყოფის მიზნები, პრობლემები, შედეგები და რეკომენდაციები

კ. ფხაკაძე, მ. ჩიქვინიძე, გ. ჩიჩუა, შ. მალიძე, დ. კურცხალია,
კ. დემურჩევი, ნ. ოქროშიაშვილი, ბ. მიქაბერიძე

მეტა-ქსელის პუბლიკაციებთან „ევროპის ენები ციფრულ ეპოქაში“ და „მეტა-ქსელის სტრატეგიული კვლევითი გეგმა 2020 წლის მრავალენოვანი ევროპისათვის“ ერთად ეს ნაშრომი ძირეულად ეყრდნობა საქართველოს ტექნიკური უნივერსიტეტის ქართული ენის ტექნოლოგიზების სასწავლო-სამეცნიერო ცენტრის გრძელვადიან პროექტებს ”ქართული ენის ტექნოლოგიური ანბანი” და „აფხაზური ენის სრული ტექნოლოგიური უზრუნველყოფის გეგმა-პროგრამა“. კერძოდ, 2019 წლის 17 მაისს, საქართველოს მეცნიერებათა ეროვნული აკადემიის ინოვაციებისა და მაღალი ტექნოლოგიების ცენტრში ქართული ენის ტექნოლოგიზების სასწავლო-სამეცნიერო ცენტრმა წარადგინა მოხსენება "ქართული და აფხაზური ენებით ევროკავშირში ანუ ქართული და აფხაზური ენების სრული ტექნოლოგიური

უზრუნველყოფის მიზნები და პრობლემები". ამრიგად, წინამდებარე ნაშრომი, რომელიც ამ ზემოაღნიშნული მოხსენების ვრცელ საპუბლიკაციო ვერსიას წარმოადგენს, დასაბუთებულია ქართული და აფხაზური ენების ციფრული კვდომის მაღალი საფეხურის საფრთხეების რეალობა; მიმოხილულია ქართული და აფხაზური ენების სრული ტექნოლოგიური უზრუნველყოფის მიზნები, პრობლემები და შედეგები; წარმოდგენილია რეკომენდაციები, რომლებსაც სრულად დაეჭირა მხარი საქართველოს მეცნიერებათა ეროვნული აკადემიის ინოვაციებისა და მაღალი ტექნოლოგიების 2019 წლის 17 მაისის სხდომის #53 ოქმით.