Automorphisms of Types and Cryptography

Sergei Soloviev

- **Main point** of this reflection is that we may use morphisms (transformations) of types in cryptography if we want to encrypt structured objects (elements of a type) instead of mere "texts".

  Example. Consider the type

$$P = (A \to A) \to ... \to (A \to A) \to (A \to A).$$

  The elements of the "premises" are functions $f_i : A \to A$, and the "conclusion" consists of $f : A \to A$ as well. So an element $F : P$ is, e.g., the composition operator, $Ff_1...f_n = f : A \to A$. In an encrypted form the order of f_i may be changed and the composition $Ff_{\sigma(1)}...f_{\sigma(n)}$ useless for an unauthorized user.
- **In Type Theory** one may introduce categorical structure taking types as objects and closed terms $t : A \to B$ as morphisms (up to usual equivalence relation based on normalization).

  Identity Id_A is represented by $\lambda x : A.x : A \to A$.

  As usual, $t : A \to B, t^{-1} : B \to A$ are mutually inverse isomorphisms if $t^{-1} \circ t \equiv id_A$ and $t \circ t^{-1} \equiv id_B$.

  E.g., $\lambda z : B_1 \to (B_2 \to C).\lambda x_2 : B_2.\lambda x_1 : B_1.(zx_1x_2)$ is an isomorphism from $B_1 \to (B_2 \to C)$ to $B_2 \to (B_1 \to C)$

  It works for different systems of Type Theory and λ-calculus.
- **An automorphism** is an isomorphism $t : A \to A$§ e.g., $\lambda z : B \to (B \to C).\lambda x_2 : B.\lambda x_1 : B.(zx_1x_2)$ (i.e., we take $B_1 = B_2$ above).

  For each type A the automorphisms $A \to A$ form the group of automorphisms $Aut(A)$.

  It may be seen as a subcategory of the groupoid of isomorphisms, i.e., the subcategory of the category of types and deductions - or terms - with the same objects and only isomorphisms as morphisms.

  If we fix a type, we may consider also the groupoid $Gr(A)$ of types isomorphic to A.
- **Relevant theorems.**

  Theorem (Dezani-Ciancaglini, [1]). $\beta\eta$-invertible terms in the untyped λ-calculus are exactly the finite hereditary permutations (f.h.p.).

On calls *erasure* elimination of all type information from λ-terms in typed λ-calculus.

Based on this theorem, it is possible to establish that the (well-typed) terms in many systems of typed λ-calculus are isomorphisms iff their erasures are f.h.p.

Theorem (Soloviev [5]). The groups $Aut(A)$ for simple types A are (up to isomorphism of groups) exactly the groups that may be obtained from symmetric groups by cartesian product and wreath product.

(By an old Jordan's theorem: they are exactly the groups of automorphisms of finite trees. Not C_3 for example.)

Theorem (Soloviev [5]). For every finite group G there exists a type A_G in second order typed λ-calculus (system F) such that the group $Aut(A_G)$ is isomorphic to G. *Idem* for Z. Luo's typed logical framework *LF* with dependent types.

- **Interest to cryptography.** Illustration: **ElGamal** cryptosystem (cf. [3, 4]).

  The protocol may use the iterations of a distinguished automorphism $g : A \to A$, where g^m is $g \circ ... \circ g$ (m times).

  Private Key: $m, m \in N$. **Public Key:** g and g^m.

  Encryption. To send a message $a : A$ (in our approach it is not a plain text, but an element of type A, and may have more complex structure) Bob computes g^r and g^{mr} for a random $r \in N$. The ciphertext is $(g^r, g^{mr}a)$.

  Decryption. Alice knows m, so if she receives the ciphertext $(g^r, g^{mr}a)$, she computes g^{mr} from g^r, then $(g^{mr})^{-1}$, and then computes a from $g^{mr}a$.

  Remark. We do not consider here the cryptosystems like **MOR** based on a more sophisticated group theory [4] but they, too, can be represented in type theory using the results of [5].

  By encoding a finite cyclic group of prime order as a group of automorphism of some type we can implement ElGamal (or any other cryptographic protocol based on finite groups) since the composition and inverse of type automorphisms (represented by finite hereditary permutations [2]) can be computed in linear time.

  However, finite cyclic groups are not the only possibility. Let us recall that the longest period in the symmetric group S_n is given by Landau function $\sim n^{\sqrt{n}}$. It corresponds to $Aut(a \to ... \to a \to p)$. This gives an idea of the length of the periods of automorphisms $f \in Aut(A)$. Maximal period of an element in the group $Aut(A)$ may be quite high.
- **This approach is not yet "ready to use".** One of the difficulties: for a type isomorphism/automorphism in *normal form* it is easy to compute the inverse. And this is not good for cryptography.
- **To be explored:** why then the terms used for encoding have to be in normal form? Normalization of a term may give "size explosion". And reduction sequence may be quite long (more that exponential). So it would be good if applying the Encryption and Decryption we might avoid normalization.
- **Example.** (Not directly related to isomorphisms.)

It is well known that natural numbers may be represented by so called *Church numerals*. In their typed version (for a type A) the number n is represented by $\lambda x : A \to A.\lambda y : A.(x(...(xy)...)$ (x is repeated n times). This is obviously related to "unary representation" of natural numbers. The term above is normal. One may define arithmetical operations and other arithmetical functions by application of other λ-terms to numerals. (In fact, using untyped lambda calculus and corresponding version of numerals, any partial recursive function may be represented.)

However, one may define also λ-terms representing binary notation. It is enough to apply (in appropriate order) two terms t_0 and t_1 representing multiplication by 2 and multiplication by 2 plus 1, to the term representing 0. Curious observation is that binary numbers are thus represented by terms that are not normal. If we normalize them we obtain exponentially longer presentation by standard Church numerals. However, to execute arithmetical operations we do not need to normalize these terms. The familiar algorithms for binary numbers may be adjusted.

- **Conclusion.** Principal idea - to use encoding that preserves functional correctness of programs. It may be done by type automorphisms represented by λ-terms. We had shown that standard cryprographic schemes (like ElGamal) may be adjusted to this case. Some other problems that have to be solved to make the scheme workable (e.g., the use of normal versus non-normal terms) were outlined.

References

[1] Dezani-Ciancaglini, M. (1976) Characterization of normal forms possessing inverse in the $\lambda - \beta - \eta$-calculus. *Theoretical Computer Science* **2** 323-337.

[2] Di Cosmo, R. (1995) *Isomorphisms of types: from lambda-calculus to information retrieval and language design.* Birkhauser.

[3] Hoffstein, J., Pipher, J. and Silverman, J.H. (2008) An introduction to mathematical cryptography. Springer, New York.

[4] Mahalanobis, A. (2015) The MOR cryptosystem and finite p-groups. *Contemp. Math.*, **633**, 81-95.

[5] Soloviev, S., Malakhovski, J. (2018) Automorphisms of Types and Their Applications. (Engl.) Zap. N. Semin. POMI, 468: 287-308.

[6] Soloviev, S. (2019) Automorphisms of Types in Certain Type Theories and Representation of Finite Groups. *Math. Structures in Computer Science*, 29(4): 511-551.

Sergei Soloviev
IRIT
Universtité Paul Sabatier Toulouse 3
Toulouse, France
e-mail: `Sergei.Soloviev@irit.fr`