An Algebraic Approach for Analyzing of Legal Requirements

Alexander Letichevsky, Oleksandr Letychevskyi
Glushkov Institute of Cybernetics of NAS of Ukraine
Kyiv,Ukraine
aaletichevsky78@gmail.com, lit@iss.org.ua

Vladimir Peschanenko, Maxsim Poltorackij
Kherson State University
Kherson, Ukraine
vladim@ksu.ks.ua, max1993poltorackij@gmail.com

***Abstract*—In this paper, we present an Algebraic Programming System in which the algebraic approach is used to formally prove the correctness or irregularity of tax actions for tax payers corresponding to tax laws. The given taxation example illustrates the application of this system to detect the tax code inconsistencies in the decisions made by the Ukrainian Taxation Office. The Algebraic Programming System is used to detect such conflicts or prove correctness using automatic reasoning from the formalized legal requirements.**

***Keywords*—algebraic programming, legal requirements, algebra of behavior**

I. Introduction

An algebraic programming system (APS) was developed by a group of researchers at the Glushkov Institute of Cybernetics. In addition to the application of this system in the field of software engineering, it is also used as a tool to process the formal presentation of legal requirements.

In this paper, we present the technology for an APS, in which the algebraic approach is used to formally prove the accuracy or irregularity of tax payer actions in accordance with the Tax Code of Ukraine.

Two significant problems arose during our research. The first problem is a formalization of the articles of the tax code and the second problem is the analysis of the created model of the laws.

For the last 15 years, significant efforts have been directed toward developing models and methods intended to retrieve and analyze information from text sources [2]. The source text, originally presented in a natural language, is converted into a specific controlled natural language (CNL) [3]. The text obtained from the CNL, with the use of some grammatical framework (GF) [4], is converted automatically into text in a formal (CL) language [5]. The resulting text in the formal CL is analyzed using a variety of CLAN tools [6], which can build counter-examples if conflicts are detected.

It is well known that legal documents are interrelated and cannot be considered in isolation. At the same time, the relationship between the statements in the legal text are presented by means of cross-references, which significantly affect the processes of structuring the organization, automatic generation of annotations for legal texts, and the extraction of the basic concepts (subjects, rights, duties, etc.).

Thus, the detection and resolution of cross-references in legal documents is a basic problem that needs to be resolved. Different approaches to the study of this problem in well-investigated American and European legislation [7] show that algebraic methods, based on the development and analysis of some calculus [8], play a significant role in legal document processing.

A legal analysis system helps lawyers conduct a legal analysis of deliberate crime cases. In legal theory, the facts of a case are presented in the form of semantic networks. The system provides the user with some results, including the logic of the legal decisions and the conclusion on which they are based. It substantiates the conclusion by providing links to legal judgments and citing supporting legal documents.

Our approach involves the interactions between a formal knowledge model presented in a formal language and a deductive system that uses well-known satisfiability modulo theories (SMT) solvers, such as Z3 [9] and CVC [10], along with special solvers that were implemented in the scope of the APS.

II. Formal Presentation of Tax Code Articles

The most recent version of the Tax Code of Ukraine was developed in 2010, and it has been subjected to significant changes since then. Nevertheless, it has many shortcomings, especially inconsistencies between articles and the omission of situations that could be treated differently. Moreover, the number of additional acts and explanations from the Taxation Office make it difficult to understand complex tax-related situations.

We formalized a Tax Code subset that contains all the articles belonged to the Value Added Tax (VAT) and regulations from additional by-laws.

To develop the algebraic model, we used two-sorted algebra that contains the algebra of behavior and a logical language with arithmetic, set-theoretic, and logic operations. The formulae in this language are constructed from operations and predicates, and they use boolean, integer, real, enumerated, and symbolic types of data [11].

The behavior algebra [12] is represented by a set of actions and behaviors with prefixing operation and non-deterministic choice. It also includes the parallel and sequential composition of processes.

We consider the interaction of agents that interact with one another. Every agent is some type of formal entity that has attributes and properties presented by formulas in logic language that could change its behavior. All the agents interact with the environment, which contains all the information that does not belong to the interacting agents.

The theory of interaction between agents and environments is known as Insertion Modelling [13]; it was developed as a generalization of the interaction of transition systems or automaton networks. An Insertion Modelling System (IMS) was developed as an extension of APS.

Usually, there are two kinds of agents in a tax code: the tax payer and the taxation office. The tax payer performs some activities, such as making payments, charging of taxes, and obtaining finance on his account. The taxation office checks the correctness of the taxes and imposes penalties corresponding to the tax code. We can also consider the set of tax payers that interact with each other.

An agent could be represented in the APS (IMS) system as an object with a number of attributes. When this term is applied to a tax payer, we have a multilevel structure with the following fields for example:

```
agent TaxPayer : obj (
        Activity : obj (
                Code : int,
                VATpayer : bool, … )
        Account: obj (
                TotalIncome : (int)->int,
                TaxCredit : int, … )  … )
```

The properties of the agents can be written as noted in the following examples.

TaxPayer.VATpayer==true

TaxPayer.TaxCredit>2000000

In APS (IMS), these properties are written as the set of formulas that could be assigned as axioms (that could not be changed) or as initial properties.

Agents interact with each other and change their states, which is represented by their properties. The interactions or actions of agents are characterized by changes in the environment that is represented by precondition and post-condition formulas in the basic language.

As an illustration, consider the fragment from a formal model of the procedure in which a person registers as a payer of the VAT, as defined in articles 180–183 of the Tax Code of Ukraine. The source for the model development is the texts of the tax code articles in natural language. It anticipates the use of an intermediate language that is similar to the natural language. In this language, the legal items are reformulated as "if ... then ...", which is convenient for the creation of the next formal presentation. It is implemented with an exact reference to the Tax Code of Ukraine or other laws referred therein. In the next stage, the formal specifications that describe the actions of the agents are constructed. Below, we present an example of some of the items of article 183 in an intermediate language that will be used to create a formal model. These items specify the actions of a person that should be registered as a tax payer and the actions of the local authorities.

Rule 1. If the tax payer is not a VAT payer whose total income during last 12 months is more than 1,000,000 Ukrainian Hrivna (UAH), then the tax payer must be registered as a VAT payer.

Rule 2. If the tax payer that is to be registered as a VAT payer did not submit the application to the State Authority before the deadline (T_CRITICAL), then the tax payer is responsible for a tax payment violation.

Rule 3. If the tax payer is to be registered as a VAT payer and did not miss T_CRITICAL deadline, then the tax payer must submit the application to the State Authority before the T_CRITICAL deadline.

Rule 4. If the tax payer must submit the application to the State Authority, then the tax payer does so.

Rule 5. If the State Authority receives the application from the tax payer, then the State Authority registers the Tax Payer as a VAT payer.

A formal presentation of these rules is provided below:

```
a1. (TaxPayer.VATPayer==false)&&
    (TaxPayer.TotalIncome(t-365,t)>1000000)&&
    (t>365)->(TaxPayer.VATPayer== true)
a2. (TaxPayer.Responsibility==false)&&
    (TaxPayer.VATPayer==true)&&
    (TaxPayer.SubmitApplication==false)&&(t>T_CRITICAL)
    ->(TaxPayer.Responsibility ==true)
a3. (TaxPayer.MustSubmitApplication==false)&&
    (TaxPayer.VATPayer==true)&&(t<=T_CRITICAL)
    ->(TaxPayer.Must-SubmitApplication==true)
a4. (TaxPayer.SubmitApplication==false)&&
    (TaxPayer.Must-SubmitApplication==true)
    ->(TaxPayer.SubmitApplication== true)
a5. (TaxPayer.RegisteredVAT==false)&&
    (TaxPayer.Submit-Application==true)
    ->(StateAthority.RegisteredVAT(TaxPayer) ==true)
```

It is important to note that we added the implicit preconditions that are absent in the articles in the Tax Code of Ukraine but are necessary for triggering of the state, for example if Tax Payer become responsible for tax payment violation then he was not before.

This sequence of actions is defined using behavior algebra expressions.

B0 = a1.B1,

B1 = a3.(a2.B3 + B3),

B3 = a4.a5.Bend

For more a natural presentation of the sequence of actions, we used Use Case Maps (UCM) [14] diagrams that could be converted into algebraic formulas. Every node in the UCM diagram is associated with rules (or an action in behavior algebra terms). The actions, a1 and a3, are dependent on time. If the formula has a time attribute, then the edge in the UCM diagram implicitly presents the changing of time as an assignment $t = t + \Delta t$.

The UCM diagram presented below illustrates the previous example.

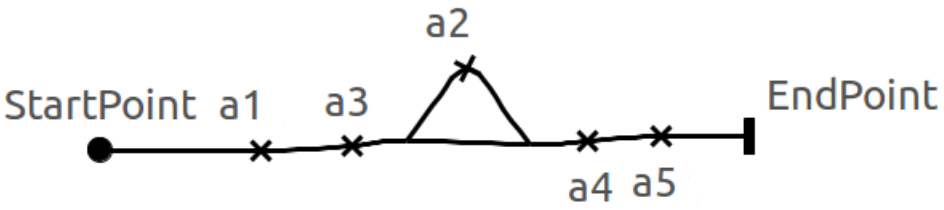

Fig. 1. Example of UCM diagram.

III. ANALYSIS OF THE ALGEBRAIC SPECIFICATIONS

Having made a formal presentation of the articles from the Tax Code of Ukraine we can provide their symbolic modeling in APS.

Fixing the initial formula, we can apply action where the precondition is true and then we can obtain the traces. A trace consists of a sequence of actions. For example, there will be two traces from the previous example: a1, a3, a4, a5 and a1, a3, a2, a4, a5.

For this purpose, APS has an engine that provides for an unfolding of the behavior into traces. Symbolic modelling transforms the initial state based on the pre- and post-conditions. It is performed by means of predicate transformers [15], which is the function that transforms the formulas. The rule is applicable and the action is admissible if the formula of the state is satisfiable. We used solving machines, such as the Z3 Microsoft prover and the CVC solver, to define satisfiability. APS also has its own proving machines for some theories with restrictions.

After modelling the specification, we can check different properties in the traces, such as safety, liveness, deadlocks, and non-determinisms. However, the space of the states during modelling could be too large, so we can consider the possibility of detecting issues statically.

Let us consider OR-fork construct of the UCM diagram where we have a branch with different rules applying..

Non-determinism could occur if the following formula of the inconsistency of preconditions is true.

Let Pre(a) be a precondition of the formal rule. Then, the inconsistency on the OR-fork construct containing rules a1,a2, …is detected if Pre(a1)∧Pre(a2)∧… is satisfiable. As applied to the Tax Code of Ukraine, this situation expresses the ambiguity in the articles.

Incompleteness is defined by the following formula used to OR-fork construct in a UCM diagram.

$\sim(Pre(a1) \vee Pre(a2) \vee \ldots) = 1$

If the formula is satisfiable, then a deadlock or a situation has been detected that is not reflected in the Tax Code.

The shortcoming of static methods is that the state presented by formula for inconsistency or incompleteness could be unreachable. Thus, it is necessary to obtain a counter example that leads to this state. This can be done using forward or backward modelling. Backward modelling generates traces from the detected property to the initial state by means of a backward predicate transformer.

IV. PROVING COMPLIANCE TO THE TAX CODE FORMAL MODEL

We will present some case from a legal practice that will illustrate the use of the APS and the formalized articles of the Tax Code of Ukraine.

A foreign investment enterprise bought cars during the financial year. The total cost of the cars was 823726,70 UAH. The tax payer did not carry this sum to a tax credit. It was included in the total production expenses of the enterprise. During the documentary check, the Taxation Office defined the tax liability on the profit, which is 300654,10 UAH, including a penalty of 64544,10 UAH. The tax payer appealed the decision, and the court considered the situation and then passed a judgment to cancel the Taxation Office's resolution.

The conflict occurred because of the inconsistency between two articles in the Tax Code of Ukraine: Article 5.3 an Article 7.2.

Article 5.3: "Any goods that have been purchased during the taxable period for production or non-production usage should not be referred to production expenses."

This article could be formally presented as:

a1. (TaxPayer.Goods==ANY)&&(TaxPayer.GoodsCost==X)
->(TaxPayer.TotalTaxableExpenses==X)

Article 7.2: "The expenses of tax payer for purchasing of cars (except taxi service enterprises) refer to the production expenses that does not refer to the tax credit."

This article could be formally presented as:

a2. (TaxPayer.Goods==CARS)&&
(TaxPayer.GoodsCost==X)&& ~(TaxPayer.Service==TAXI)
->(TaxPayer. TotalProductionExpenses==X)

We also considered some reduced rules for the end of the taxable period that define the payment of taxes.

a3. (t == TAX_PAYMENT_TIME) ->(TaxPayer.Taxes ==
TaxPayer.ProfitTax+TaxPayer.VATTax)&&
(TaxPayer.ProfitTax==
Coefficient*TaxPayer.TotalTaxableExpenses)

The fragment of the Tax Code of Ukraine belonging to these articles can be presented as the following UCM diagram.

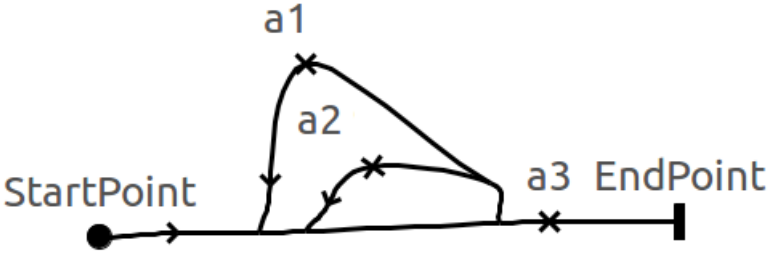

Fig. 2. UCM of fragment of the Tax Code.

The case from the point of view of a taxpayer is presented using the following sequence of actions presented by formula:

Action 1: (TaxPayerGoods==CARS)&&
(TaxPayerGoodsCost==823726,70)&&
(TaxPayer.TotalProductionExpenses== 823726,70)

Action 2: TaxPayer.Taxes==0

The case from the point of view of the Taxation Office is presented using the following sequence.

Action 1: (TaxPayerGoods==ANY)&&
(TaxPayerGoodsCost==823726,70)&&
(TaxPayer.TotalTaxableExpenses== 823726,70)

Action 2: TaxPayer.Taxes==300 654,10

If we apply symbolic modelling correspondingly to every case, we can detect that both cases are applicable and consistent with the Tax Code of Ukraine.

The inconsistency of the articles could be detected using a static method. We can see that Pre(a1)∧Pre(a2) is satisfiable, and a1 and a2 cause the different post-conditions. Moving from this state to the initial condition, we can obtain two different traces that correspond to the considered cases.

V. IMPLEMENTATION

The proposed approach has been applied for the formalization and analysis of the Tax Code of Ukraine. The Tax Code consists of 357 articles written on 780 pages. The articles are structured in 20 sections, with the major ones focusing on VAT and profit taxes. Around 30% of the articles present a definition of the terms used in the Tax Code. The process of formalization involves extracting two kinds of formal statements. The first kind is non-behavioral statements that define the relations between agent attributes. These articles also define the attributes, objects, and its components. We gather these non-behavior statements in a set of axioms, which will be checked for consistency in the APS system. The second kind of formal statement is behavioral; these statements are presented as UCM diagrams with nodes (responsibilities) that contain actions with pre- and post-conditions. Each node of the UCM diagram is mapped to the corresponding article of the Tax Code. Axioms shall be consistent with the state of the environment during symbolic modeling. Two specialists were involved in the process—an algebraist with full-time employment and a specialist in jurisprudence with a good background in economic and accounting.

Using APS, we simulated this model to generate possible behavior scenarios for debugging. The obtained traces presented the sequences of the possible actions of agents, especially of tax payers. The traces contained possible commercial operations and corresponding tax payments in the defined tax periods. Traces were verified by a third person with an economic background, and the corresponding corrections in the model were made.

The next checking of the model was conducted for the detection of inconsistency and incompleteness in formal statements. Three major findings on incompleteness and three on inconsistencies in the behavioral requirements for the VAT section were found and presented to lawyers. Eighteen minor findings were found, and these were mostly attributed to the ambiguous interpretation of articles because of language issues. Some inconsistencies could be resolved through other by-law documents that may also be formalized and included in the model. The other sections in the model are currently being analyzed.

VI. CONCLUSION

The considered approach could be useful for the further formalization and processing of legal requirements. A formal model of requirements is developed in the terms of theory of interactions of agent and environments, and it serves as the input of the APS(IMS) that was used earlier for the verification of formal requirements in software engineering. The symbolic modeling of formal requirements based on theory of predicate transformers and invariant computations gives the possibility to cover more scenarios of model behavior.

In the context of the Tax Code, we use static formal methods that are realized in APS to detect errors in the model. The specific of model is that many non-behavioral statements are formalized as the properties of the agents' attributes. The APS utilizes methods to check the consistency of non-behavioral statements, especially proving of satisfiability by using deductive solvers. A symbolic modeling of a legal requirements model requires consistency in the current state of agents with a large number of non-behavioral statements. The special predicate transformers facilitate efficiency in the next state's computation.

In general, the APS(IMS) approach proved to be efficient enough. We did not encounter exponential explosion, and proving with scenario generation was performed within a reasonable time. The next step is to provide an efficient method of formalization that takes much time. Further studies could focus on the efficient processing of agent properties and attributes, as well as the generation of formal statements.

REFERENCES

[1] APS and IMS are best for rewriting and modelling [www.apsystem.org.ua]

[2] C. Manning, P. Raghavan and H. Schutze, "Introduction to Information Retrieval", in*: Cambridge University Press*, 544 p., 2009.

[3] A. Wyner, K. Angelov, , G. Barzdins, D. Damljanovic, B. Davis, N. Fuchs, S. Hoefler, K. Jones, K. Kaljurand, T. Kuhn, M. Luts, J. Pool, M. Rosner, R. Schwitter, J. Sowa, "On controlled natural language," in: *LNCS/LNAI*, vol. 5972, pp. 281-289, 2010.

[4] M. Hamdaqa and A. Hamou-Lhadj, "An approach based on citation analysis to support effective handling of regulatory compliance," in: *Future Generation Computer Systems*, vol. 27, pp. 395–410, 2011.

[5] C. Prisacariu and G. Schneider, "A Formal Language for Electronic Contracts," in*: LNCS*, vol. 4468, pp. 174–189, 2007.

[6] S. Fenech, G. J. Pace, G. Schneider, "CLAN: A Tool for Contract Analysis and Conflict Discovery," in: LNCS, vol. 5799, pp. 90-96, 2009.

[7] E. de Maat, R. Winkels and T. van Engers, "Automated detection of reference structures in law," in: *Proc. of the 19th Annual Conference on Legal Knowledge and Information Systems*, pp. 41–50, 2006.

[8] Nicolas Sannier, Morayo Adedjouma, Mehrdad Sabetzadeh, Lionel C. Briand: "An automated framework for detection and resolution of cross references in legal texts" in Requir. Eng. 22(2), pp. 215-237, 2017.

[9] L. de Moura, N. Bjørner, "Z3: An Efficient SMT Solver," in: LNCS, vol. 4963, pp. 337-340, 2008.

[10] C. Barrett, C. L. Conway, M. Deters, L. Hadarean, D. Jovanović, T. King, A. Reynolds, C. Tinelli, "CVC4," in: LNCS, vol. 6806, pp. 171-177, 2011.

[11] A. Letichevsky, J. Kapitonova, O. Letychevskyi, V. Volkov, S. Baranov, V.Kotlyarov, T. Weigert, "Basic Protocols, Message

Sequence Charts, and the Verification of Requirements Specifications,“ in: Computer Networks, pp. 662-675, 2005.

[12] A. Letichevsky, “Algebra of behavior transformations and its applications,” in: V.B. Kudryavtsev and I.G. Rosenberg eds. *Structural theory of Automata, Semigroups, and Universal Algebra, NATO Science Series II. Mathematics, Physics and Chemistry*, vol. 207, pp. 241-272, 2005.

[13] A. Letichevsky and D. Gilbert, “Interaction of agents and environments,“ in: *Resent trends in Algebraic Development technique, LNCS 1827* (D. Bert and C. Choppy, eds.), Springer-Verlag, 1999.

[14] Daniel Amyot, Gunter Mussbacher: “User Requirements Notation: The First Ten Years, The Next Ten Years”, (Invited Paper). JSW 6(5): pp. 747-768, 2011.

[15] A.A. Letichevsky, A.B. Godlevsky, O.O. Letychevskyi, S.V. Potienko, V.S. Peschanenko, ”Properties of Predicate Transformer of VRS System,” in: Cybernetics and System Analyses, vol. 4, pp. 3-16, 2010.