LLM-based DatalogMTL Modelling of MiCAR-compliant Crypto-Assets Markets

Andrea Colombo[1,*], Teodoro Baldazzi[2], Luigi Bellomarini[3], Andrea Gentili[3] and Emanuel Sallinger[4,5]

[1]Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria, Milan, Italy
[2]Università Roma Tre, Department of Computer Science and Engineering, Rome, Italy
[3]Banca d'Italia, Rome, Italy
[4]TU Wien, Faculty of Informatics, Vienna, Austria
[5]University of Oxford, Department of Computer Science, Oxford, UK

Abstract

Recent extensions of Datalog that consider the temporal dimension as a first-class citizen have unlocked the possibility of using its temporal variants, such as DatalogMTL, to model and reason about complex financial domains. Very relevant ones are crypto-activity markets, which, according to the recent Markets in Crypto-Assets Regulation (MiCAR) of the EU, are described by white papers published by crypto-assets issuers.

In particular, the issuers publish semi-structured information about the assets they are willing to offer. Then, the assets are implemented in decentralized finance contexts (i.e., in a blockchain) as executable scripts known as smart contracts. However, these scripts are often criticized for their complexity, which makes them challenging to understand and communicate. On the other hand, in our experience, the availability of a declarative and executable representation of a crypto-activity market fosters a better understanding of that market as well as improved transparency, reproducibility and, as a consequence, increased fairness. These characteristics are of major interest to the financial authorities for example for supervision purposes.

In this paper, we study the problem of automatically translating textual descriptions of crypto-assets, written according to the MiCAR specifications, into DatalogMTL programs that represent and capture the respective crypto-activity market. To this end, we opt for a machine translation approach and leverage a Large Language Model. We discuss promising techniques and preliminary experimental results.

Keywords

DatalogMTL, crypto assets, MiCAR, large language models

1. Introduction

Knowledge Representation and Reasoning (KRR) formalisms, such as Datalog and its extensions, are gathering increasing attention in industrial contexts. This is particularly evident in the financial sector, with applications in the so-called Decentralized Finance (DeFi), a novel financial paradigm that leverages distributed ledger technologies to offer services without intermediaries. Logical languages such as Datalog effectively balance expressive power and computational complexity, enabling, thanks to their extension to the temporal dimension, the creation of concise and efficiently executable formalizations of smart contracts, i.e., executable scripts that implement a financial asset [1, 2, 3]. Additionally, the inherent step-by-step nature of logical reasoning aligns closely with concepts of explainability, thereby supporting transparency in decision-making activities. The declarative paradigm promotes simplicity, trustworthiness, compactness, and code comprehensibility, making it algorithm-independent and more aligned with high-level specifications, policies, and standards. Among these, the recent Markets in Crypto-Assets Regulation (MiCAR) of the European Union [4] introduces rules for crypto-activity issuers, such as the requirements for white papers, i.e., semi-structured documents containing all information about the crypto assets being offered or admitted to trading. The availability of such

Datalog 2.0 2024: 5th International Workshop on the Resurgence of Datalog in Academia and Industry, October 11, 2024, Dallas, Texas, USA

*Corresponding author.
andrea1.colombo@polimi.it (A. Colombo); teodoro.baldazzi@uniroma3.it (T. Baldazzi); luigi.bellomarini@bancaditalia.it (L. Bellomarini); andrea.gentili@bancaditalia.it (A. Gentili); sallinger@dbai.tuwien.ac.at (E. Sallinger)

Part	Description	Fields
A	Information about the issuer	27
B	Information about the e-money token	4
C	Information about the admission to trading	5
D	Rights and obligations of the token	11
E	Information about the underlying technology	10
F	Information on the risks	4
G	Information on sustainability	1

Part	Description	Fields
A	Information about the issuer	27
B	Information about the asset referenced token	7
C	Information about the admission to trading	29
D	Rights and obligations of the token	33
E	Information about the underlying technology	9
F	Information on the risks	7
G	Information on the reserve of assets	7

Table 1
Templates for e-money (left) and asset-references (right) token white paper contents.

templates is appealing if combined with the impressive power of state-of-the-art Large Language Models (LLMs), which have proved to be capable of understanding and manipulating complex unstructured data as text. One of the most promising applications is using LLMs to analyze white papers, their compliance with rules, and their features, which may impact financial markets, raising the interest of financial authorities.

Leveraging Large Language Models for the automatic translation of natural language descriptions into executable code on a blockchain is a novel and unexplored field due to the challenges of (i) converting text to programming languages an LLM has rarely seen, and (ii) ensuring the fidelity and quality of the translated content [5]. Recent works, such as SolMover [6], build frameworks that, by employing LLMs, try to understand coding concepts and use them to translate from a code like Solidity into a non-trained source language. While results are promising, this approach is a code-to-code translation, thus not directly a natural language (NL)-to-code translation. Other works are based on a human-LLM continuous dialogue, such as via ChatGPT, that supports the writing of Solidity code for smart contracts, although human intervention and correction are still required [7].

Contribution. In this short work, we present our preliminary approach that aims at translating white papers into an executable and inherently transparent language as DatalogMTL. Our approach leverages the semi-structured format enforced by the MiCAR to build a pipeline that supports a pre-trained LLM agent in the translation task by (i) reducing the amount of information passed to the model, filtering in only specific sections of the white paper, and (ii) implementing token-specific pre- and post-processing steps that help the LLM achieve an executable output.

2. Background

In this section, we briefly recall DatalogMTL foundations required for its use in crypto-activity modelling, i.e., to model smart contracts. Then, we outline the main aspects of interest introduced by the MiCAR.

DatalogMTL. DatalogMTL [8, 9] is a recently developed extension of Datalog with operators from *Metric Temporal Logic* (MTL) interpreted over the rational timeline and with stratified negation under stable model semantics. In [10], we showed how the only required temporal operator for implementing financial markets in DatalogMTL is the use of $\boxminus_{[t_1,t_2]}$ over a punctual interval, i.e., up to $t_1 = t_2$. Informally speaking, given the rule $\boxminus_{[t_1,t_2]} A_1 \rightarrow A_2$ and a time interval $[x, y]$, the $\boxminus$ operator defines the satisfaction of an atom A_2 within the interval $[x+t_2, y+t_1]$, based on the continuous satisfaction of an atom A_1 within the interval $[x, y]$, where $x + t_2 \leq y + t_1$. The idea is that, by considering punctual intervals, we can model the evolving market at any timestamp and create rules that react in case an event occurs, such as a transfer of assets between two market participants.

The MiCA Regulation in the EU. The Markets in Crypto-Assets (MiCA) regulation, enacted by the European Union, covers three categories: asset-referenced tokens, e-money tokens, and other crypto-assets. Since the scope of the third category is broader, in this work we will focus only on the first two more specific tokens. For each crypto asset category, the MiCAR contains norms regarding the

content of the white paper. While no official template has been published yet, we report in Table 1 the likely structure of the templates, following the consultations that are in progress [11].

3. LLM-based White Paper Translation Pipeline

The problem of translating an arbitrary NL whitepaper into a DatalogMTL specification is inherently complex and our experience shows that pure *machine translation approaches*, where the LLM is directly tasked with the NL-to-DatalogMTL translation goal (even when fine-tuning is involved) are not effective and lead to arbitrary programs, highly affected by hallucinations and overall incorrect. To overcome these difficulties, we suggest a paradigm shift and follow a *template-based technique*, implemented in the pipeline shown in Figure 1.

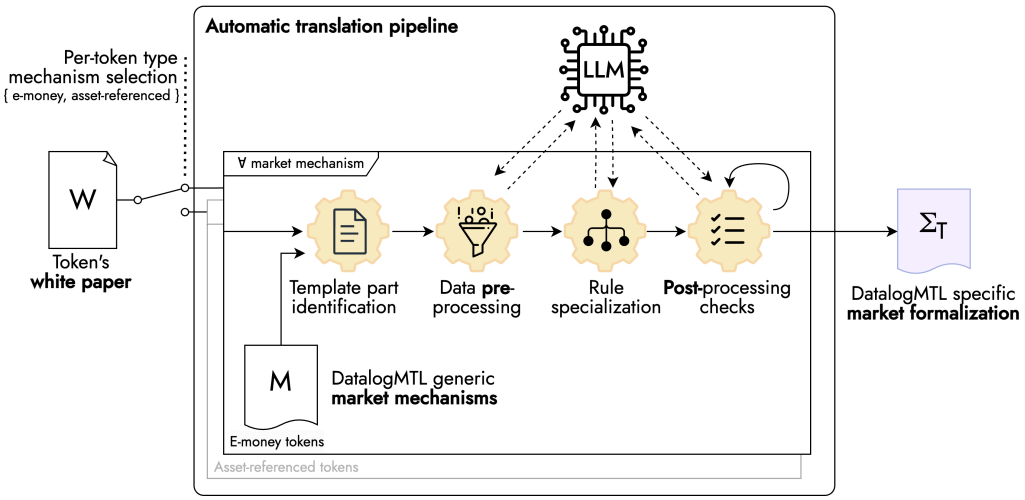

Figure 1: Proposed approach to convert white papers into DatalogMTL programs.

Overview. We pre-define a set of *market mechanisms* $\mathcal{M}$ that represent standard "functional components" of a market, such as creation (*minting*), selling of a token (*redemption*), and so on. For each pattern, we feature pre-built templatized (i.e., with formal variables) DatalogMTL rules decorated with a natural language description. Examples are in Figure 2. Then, given a white paper, we can first select the templatized rules according to the token type, and iteratively prompt the LLM with all the information required to *specialize* the rules to the white paper content.

LLM-based Rules Specialization. Algorithm 1 describes the process we developed to adapt the generic rules to the token-specific features. Based on a custom input MiCAR-compliant white paper, the corresponding set of templatized rules Σ, either e-money token or asset-referenced token, is considered. Then, for each market mechanism M of the set $\mathcal{M}$ of mechanisms (line 1), we consider the set of templatized rules R, their description D that we have pre-defined, and we extract the portion I of the white paper that contains relevant information for the mechanism (lines 2-4). Each mechanism is also decorated with a set of *keywords* $\mathcal{K}$, which describe the content of the mechanism and activate optional pre-processing steps (lines 5-7). Thus, if the mechanism deals with one of the topics defined by the keywords, a pre-processing LLM call will be performed to rewrite the information that was inserted in the white paper fields. The aim of this step is to support the LLM-translation task by simplifying, if possible, the textual input. For instance, this is the case of *temporal* information such as the specification of market opening days, which can be defined in multiple formats, e.g., the market is closed on weekends. Then, the LLM is provided R, D and I and is prompted to perform the actual specification (line 8): "*These are Datalog rules: {R} where {D}. Modify them according to this information: {I}*" Finally, the LLM generates the translated rules, and we perform post-processing checks (line 9-11), such as detection of syntax or programmatically identifiable errors. In case checks are not passed, the LLM is asked to repeat the task until they are passed.

Mechanism	E-Money Token	Asset-Referenced Token
Collateral Acceptance	$MintOrder(b,q,t), t = t_1 \rightarrow CollateralAccept(b,q,t)$	$MintOrder(b,q,t), t = t_1 \rightarrow CollateralAccept(b,q,t)$ $MintOrder(b,q,t), t = t_2 \rightarrow CollateralAccept(b,q,t)$
	t_1 can be a fiat currency or a short-term treasury bond acceptable as collateral	t_1 and t_2 are acceptable collaterals, as fiat currencies, physical assets or cryptocurrencies
Minting	$CollateralAccept(b,q,t), q \geq q_1 \rightarrow Mint(b,q)$ $CollateralAccept(b,q,t), q < q_1 \rightarrow Mint(b,0)$	$CollateralAccept(b,q,t), Price(t,p), q \geq q_1 \rightarrow Mint(b,q*p)$ $CollateralAccept(b,q,t), Price(t,p), q < q_1 \rightarrow Mint(b,0)$
	q_1 is the minimum quantity that it's possible to hold	q_1 is the minimum quantity that it's possible to hold
Redemption	$\boxminus_{[1,1]} Balance(u,n), RedeemOrder(u,q), q \leq n \rightarrow Redeem(u,q)$	$\boxminus_{[1,1]} Balance(u,n), RedeemOrder(u,q), q \leq n \rightarrow Redeem(u,q)$
	The rule describes that you need enough units in balance for redeeming	The rule describes that you need enough units in balance for redeeming
Transfer	$\boxminus_{[1,1]} Balance(s,o), Transfer(s,r,u,b), b = b_1, o \geq u \rightarrow TransferApproved(s,r,u,b)$ $\boxminus_{[1,1]} Balance(s,o), Transfer(s,r,u,b), b = b_1, o < u \rightarrow TransferDenied(s,r,u,b)$	$\boxminus_{[1,1]} Balance(s,o), Transfer(s,r,u,b), b = b_1, o \geq u \rightarrow TransferApproved(s,r,u,b)$ $\boxminus_{[1,1]} Balance(s,o), Transfer(s,r,u,b), b = b_1, o < u \rightarrow TransferDenied(s,r,u,b)$
	b_1 is a supported blockchain for the token	b_1 is a supported blockchain for the token

Figure 2: Examples of templatized rules Σ of some relevant token mechanisms, namely, the collateral acceptance terms, minting and redemption, and transfer conditions. Each mechanism is defined as a pair of DatalogMTL rules - NL description that illustrates some relevant aspects of the rules, i.e., either the role of the variables or a generic description. The number of pre-defined rules in the mechanism is arbitrary and, as we shall see, will help the translation mechanism. Possible differences between e-money tokens and asset-referenced ones might refer either to the rules or to the descriptions and depend on the different nature of the token.

Algorithm 1 Rules Specialization.

for $\forall M \in \mathcal{M}$ **do** ▷ For each Token's Market Mechanism
  $R \leftarrow \Sigma.GetRules(M)$ ▷ Get generalized DatalogMTL rules
  $D \leftarrow \Sigma.GetDescription(M)$ ▷ Get the description for the mechanism variables
  $I \leftarrow WhitePaper.GetRelevantFields(M)$ ▷ Query the Textual Information from the White Paper
  if $M.Keywords \in \mathcal{K}$ **then** ▷ Mechanism deals with a topic requiring pre-processing
    $I \leftarrow LLMRewrite(I)$ ▷ LLM is asked to rewrite the information
  end if
  $R^* \leftarrow LLMPrompt(R, D, I)$ ▷ LLM is prompted to adapt the rules R according to I
  while $R^*.checks$ **not passed do**
    $R^* \leftarrow LLMPrompt(R, D, I)$ ▷ LLM is asked to repeat the task
  end while
end for

For our preliminary experiments and tests, we employed a pre-trained LLaMA 3 70B as our reference LLM. The model was prompted with system content providing a generic DatalogMTL adaptation example. We opted not to perform fine-tuning since (i) few-shot learning demonstrated promising results in our initial trials, and (ii) there is a current lack of large-scale datasets of DatalogMTL rules.

Example 1 (X-Coin - Supported Blockchain). *Consider the market mechanism $M = Transfer$ for a hypothetical e-money token named X-Coin. It defines the conditions required by the white paper for the approval of a transfer of tokens between a Sender s and a Receiver r of u units on the blockchain b. For instance, suppose the only main conditions for approving a transfer are (i) having enough tokens in the sender's balance, and (ii) that the transfer occurs on a supported blockchain. Figure 3 illustrates the pipeline in action for the Transfer mechanism. In this example, the variable b_1 (refer to Figure 2) has been instantiated and the model has also created additional rules to account for the multiple blockchains, i.e., inserting OR conditions. To achieve this, the input has been automatically pre-processed: since the mechanism referred to the supported blockchain technology, a step to separate each blockchain type has been activated. Then, post-processing checks have controlled the syntax in the output, such as the presence of all initial predicates and the absence of additional undesired predicates.*

DatalogMTL Market Formalization. By collecting all the rules that have been generated or modified in Algorithm 1, we can create a DatalogMTL program that formalizes the token's specific market that, if extensional facts are injected, is executable in a DatalogMTL engine. Full examples of our approach over

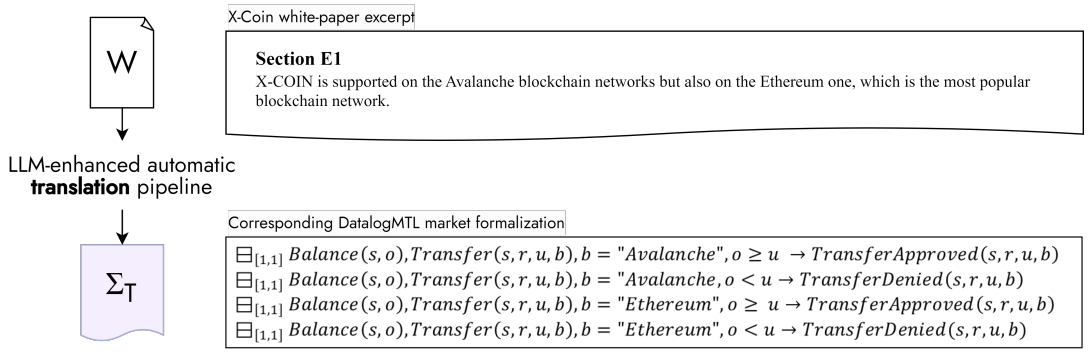

Figure 3: The LLM-enhanced translation pipeline in action for the *Transfer* mechanism for the X-Coin white paper. The pipeline takes as input the E1 field of the X-Coin white paper, since we know, from MiCAR, that we can find in that field the information about the blockchain. In output, we have the DatalogMTL rules that have been specialized according to the specific white paper content.

existing crypto activities, converting natural language white papers into a DatalogMTL formalization, can be found in our repository [12]. Such programs are executable if the required extensional facts are provided and replicate the conditions laid out in the white papers.

Discussion and Limitations. We developed the pipeline with the goal of producing DatalogMTL programs that are executable in any reasoner engine that supports the MTL extension of Datalog. Thus, engines like the MeTeoR [13] and Vadalog [14] systems can execute the generated programs. However, in all cases, the extensional database to practically run the program should be manually (or semi-automatically) generated, in the scenario you aim to reproduce or test the market for particular use cases, which is our goal for future work. In particular, any interested user might inject a market scenario and run the corresponding DatalogMTL program to see what the consequences of such a scenario would be.

Finally, while the *semantic correctness* of the resulting programs can be evaluated using rule-based checks or domain-specific constraints, which involve detecting rule-specific features, assessing the *content correctness* is more challenging. Specifically, verifying that the translated rules, including potential errors introduced by the LLM, accurately replicate the mechanisms of a specific token remains difficult, particularly in an automated way. In fact, although employing human evaluators seems like a straightforward solution, automatic checks and correctness experiments would require a gold standard, i.e., the true sets of market rules, which is so far unavailable. Nevertheless, our pipeline is strictly guided, with the presence of specific regulations and pre-defined rule templates that mitigate the issue of creating unreliable DatalogMTL representations of crypto-markets.

4. Conclusion and Future Work

In this work, we presented the first approach that, by employing the enormous power of pre-trained LLMs in a controlled fashion, tries to model crypto activity markets in a tractable and inherently transparent formal language, i.e., DatalogMTL. Preliminary results show that we can automatically generate executable DatalogMTL programs starting from white papers in natural language by carefully instructing the LLM to perform the translation task and by leveraging the semi-structured nature of the MiCAR. In future works, we will expand on this approach and industrialize it on a larger scale, allowing stakeholders to get a DatalogMTL formalization of any MiCAR-compliant token. We will also test whether such programs fully replicate the markets by simulating their evolution and comparing it with the real market they aim to represent.

Acknowledgments

This work has been funded by the Vienna Science and Technology Fund (WWTF) [10.47379/VRG18013, 10.47379/NXT22018, 10.47379/ICT2201]. The views expressed in the paper are those of the authors and do not necessarily reflect those of the Bank of Italy.

References

[1] M. Nissl, E. Sallinger, Modelling Smart Contracts with DatalogMTL., in: EDBT/ICDT Workshops, volume 3135, 2022.

[2] T. Baldazzi, L. Bellomarini, S. Ceri, A. Colombo, A. Gentili, E. Sallinger, Fine-tuning large enterprise language models via ontological reasoning, in: RuleML+RR, Springer, 2023, pp. 86–94.

[3] L. Bellomarini, M. Favorito, E. Laurenza, M. Nissl, E. Sallinger, Towards FATEful Smart Contracts, in: "6th Distributed Ledger Technology Workshop, May 14–15, 2024, Turin, Italy", 2024.

[4] EU, Regulation - 2023/1114 - en - eur-lex, https://shorturl.at/QAyRM, 2023.

[5] F. Barbàra, E. A. Napoli, V. Gatteschi, C. Schifanella, Automatic Smart Contract Generation Through LLMs: When The Stochastic Parrot Fails, in: "6th Distributed Ledger Technology Workshop", 2024.

[6] R. Karanjai, L. Xu, W. Shi, SolMover: Smart Contract Code Translation Based on Concepts, Association for Computing Machinery, New York, NY, USA, 2024, p. 112–121.

[7] N. Petrović, I. Al-Azzoni, Model-Driven Smart Contract Generation Leveraging ChatGPT, in: Advances in Systems Engineering, Springer Nature Switzerland, 2023, pp. 387–396.

[8] S. Brandt, E. G. Kalayci, V. Ryzhikov, G. Xiao, M. Zakharyaschev, Querying Log Data with Metric Temporal Logic, J. Artif. Int. Res. 62 (2018) 829–877. doi:10.1613/jair.1.11229.

[9] P. Walega, M. Zawidzki, B. C. Grau, Reasoning Techniques in DatalogMTL, in: Proceedings of Datalog 2.0, 2022.

[10] A. Colombo, L. Bellomarini, S. Ceri, E. Laurenza, Smart Derivative Contracts in DatalogMTL, in: Proceedings of the EDBT Conference, volume 26, 2023, p. 773 – 781.

[11] C. Camassa, Legal NLP Meets MiCAR: Advancing the Analysis of Crypto White Papers, in: Proceedings of the Natural Legal Language Processing Workshop 2023, 2023, pp. 138–148.

[12] A. Colombo, MiCAR White Paper - DatalogMTL Translation Examples, https://bit.ly/MiCARTranslations, 2024. (Accessed on 14/08/2024).

[13] D. Wang, P. Hu, P. A. Wałęga, B. C. Grau, Meteor: Practical reasoning in datalog with metric temporal operators, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 36, 2022, pp. 5906–5913.

[14] L. Bellomarini, L. Blasi, M. Nissl, E. Sallinger, The Temporal Vadalog System, in: Rules and Reasoning: 6th International Joint Conference on Rules and Reasoning, RuleML+RR 2022, Berlin, Germany, September 26–28, 2022, Proceedings, Springer-Verlag, 2022, p. 130–145.