SHOULD BE REPLACED ON REQUIRED TITLE PAGE

Instruction

1. Open needed docx template (folder "title"/<your department or bach if bachelor student>.docx).

2. Put Thesis topic, supervisor's and your name in appropriate places on both English and Russian languages.

3. Put current year (last row).

4. Convert it to "title.pdf," replace the existing one in the root folder.

Contents

List of Tables

List of Figures

Abstract

abstract . . .

Chapter 1

Introduction

I Spacing & Type

A. Creating a Subsection

1) Creating a Subsubsection:

2) Creating a Subsubsection:

3) Creating a Subsubsection:

a) This is a heading level below subsubsection: And this is a quote:

> Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all

letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

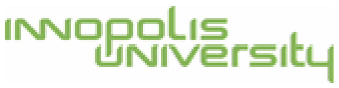

Fig. 1. One kernel at $\mathrm{x_s}$ (dotted kernel) or two kernels at $\mathrm{x_i}$ and $\mathrm{x_j}$ (left and right) lead to the same summed estimate at $\mathrm{x_s}$. This shows a figure consisting of different types of lines. Elements of the figure described in the caption should be set in italics, in parentheses, as shown in this sample caption.

This is a table:

TABLE I
This Is a Table Example

A	B	C
a1	b1	c1
a2	b2	c2
a3	b3	c3
a4	b4	c4

The package “upgreek” allows us to use non-italicized lower-case greek letters. See for yourself: $\upbeta$, $\boldsymbol{\upbeta}$, β, $\boldsymbol{\beta}$. Next is a numbered equation:

$$\|\mathbf{X}\|_{2,1} = \underbrace{\sum_{\mathrm{j}=1}^{\mathrm{n}} \mathrm{f_j}(\mathbf{X})}_{\text{convex}} = \sum_{\mathrm{j}=1}^{\mathrm{n}} \|\mathbf{X}_{\cdot,\mathrm{j}}\|_2 \tag{1.1}$$

The reference to equation (1.1) is clickable.

II Theorems, Corollaries, Lemmas, Proofs, Remarks, Definitions,and Examples

Theorem 1. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Proof. I'm a (very short) proof. □

Lemma 1. I'm a lemma.

Corollary 1. I include a reference to Thm. 1.

Proposition 1. I'm a proposition.

Remark. I'm a remark.

Definition 1. I'm a definition. I'm a definition. I'm a definition. I'm a definition. I'm a definition. I'm a definition. I'm a definition. I'm a definition. I'm a definition. I'm a definition. I'm a definition.

Example. I'm an example.

III Section with linebreaks in the name

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Chapter 2

Literature Review

This chapter introduces the reader to the Ethereum ecosystem and provides related literature. Section 2.1 describes the Ethereum Virtual Machine (EVM) and its gas consumption mechanism. Section 2.2 describes existing EVM languages. Section 2.3 gives a brief overview of compiler construction. In section 2.4 the Elixir language is introduced. To finalize, section 2.5 concludes this chapter.

I Ethereum Virtual Machine

Ethereum is a decentralized system, and at its core, the Ethereum Virtual Machine (EVM) is an essential component of the Ethereum blockchain. The EVM serves as the computational heart of the Ethereum network, enabling the execution of smart contracts and providing the foundation for a wide array of decentralized applications.

In the context of smart contract execution, it's important to mention the concept of gas limit. Gas limit is a critical aspect of Ethereum's execution model. Each operation and computation on the Ethereum Virtual Machine consumes a certain amount of gas, which is a measure of computational work. The gas limit is a cap set by the user or the entity initiating a contract execution, and it represents the maximum amount of gas they are willing to spend for that operation.

Gas limit, the EVM's stack-based architecture, and the wide set of available operations allow it to efficiently and securely for miners execute these smart contracts, granting it the status of a Turing-complete machine [1]. This level of computational flexibility empowers developers to create sophisticated programs.

II Existing EVM languages

Official Ethereum documentation [2] enumerates only four languages that compile to EVM: Solidity, Vyper, Yul, and Fe. However, other attempts to create a language for EVM are known [3].

Among the available languages for EVM, Solidity stands out as the most widely adopted and utilized one [4], [5]. This dominance is further emphasized by data presented in [6], which reveals that there are approximately 800 times more Solidity files on Github compared to Vyper files. Consequently, the majority of existing smart contracts on the Ethereum blockchain are written in Solidity. On one hand, the concentration of development and resources around a single language like Solidity can be viewed positively. It fosters a strong sense of community cohesion and ensures that the majority of developers are focused on refining and improving a single language, potentially enhancing its robustness and feature set. However, this dominance also brings significant challenges. The high prevalence of Solidity in the EVM ecosystem means that if a vulnerability or security flaw arises in the Solidity language, it affects a substantial portion of the smart contracts on the network. This centralized risk can be a double-edged sword, as a single point of failure in Solidity could have far-reaching consequences for the entire Ethereum blockchain. Furthermore, the Ethereum community is not without its criticisms of Solidity. Community members have voiced their concerns and challenges related to the language [5]. These concerns range from the language's complexity to its lack of certain features, which may hinder the development of robust and secure smart contracts. So, decentralization and availability of choice are important, especially for such a decentralized ecosystem as blockchain.

A. Solidity

Solidity is a high-level, object-oriented programming, statically typed language designed specifically for the EVM. It has syntax similar to C++, Python, and Javascript [7]. It supports multiple inheritance, complex user-defined types, and libraries as well as the number of other features [8].

B. Vyper

Vyper contrasts significantly with Solidity, it is based on Python and extends it to suit smart contract development. Vyper aims for security and simplicity in terms of compiler implementation itself and code readability to be auditable more easily. For instance, it lacks a list of features that are considered harmful for code readability, and that makes the language more error-prone [9].

III Compiler construction

The process of compiler construction is a crucial bridge between high-level programming languages and the machine code that EVM understands. This process can be divided into three or four main stages, each playing a pivotal role in transforming human-readable code into executable instructions that can be processed by the EVM. Here is a general overview of these stages:

A. Tokenization

Tokenization is the initial step in the compilation process. During this stage, the source code is broken down into smaller units called tokens. These tokens are fundamental building blocks, including keywords, identifiers, operators, and constants. Tokenization simplifies the process of analyzing and parsing the code by providing a structured representation of the code's elements.

Tokenization facilitates syntactical and lexical analysis, ensuring that the code adheres to the language's grammar and can be properly understood.

B. Parsing

Parsing follows tokenization and focuses on the syntax of the source code. During parsing, the compiler checks the arrangement and structure of tokens to create a structured representation, often in the form of a syntax tree or abstract syntax tree (AST). This representation helps ensure that the code conforms to the language's grammar rules and can be further processed.

C. Intermediate Representation (Optional)

While not always a mandatory stage, the use of an intermediate representation can greatly enhance the efficiency and capabilities of a compiler. In the context of EVM, various intermediate representations like Yul and Yul+ are used for optimizing code. These representations help in enhancing code quality, optimizing performance, and preparing the code for final code generation.

Additionally, a language like Simplicity is employed for formal proofs. This stage is particularly important in scenarios where verification, security, and performance optimizations are key concerns.

D. Code Generation

The final stage of the compilation process involves code generation. During this stage, the compiler generates the actual machine code that the EVM can execute. The generated code is specific to the EVM's hardware architecture and represents a translation of the original high-level code into instructions that the EVM can understand and execute.

The code generation stage is crucial in ensuring that the compiled code is efficient and correctly reflects the intended behavior of the high-level source code.

In conclusion, compiler construction is fundamental to the Ethereum ecosystem, enabling the translation of high-level languages into machine code that can be executed on the EVM. The stages of tokenization, parsing, optional intermediate representation, and code generation are essential components of this process. The flexibility to compile existing languages like Vyper further enriches the Ethereum development landscape, making it accessible to a wider range of developers and promoting the platform's growth and adoption.

IV The Elixir language

Elixir is a modern programming language, which is working on the top of Erlang VM. It has immutable variables that will make smart contracts' code more clear and straightforward. Elixir's metaprogramming capabilities enable the creation of domain-specific languages (DSLs). This is valuable for writing smart contracts, as it allows developers to express contract logic in a way that is highly readable and aligned with the problem domain. Metaprogramming can simplify contract development and make it more accessible to a wider range of developers. According to O'Grady [10] Elixir is a top 50 language. Moreover, it's more popular than Solidity. Therefore using Elixir will expand Ethereum community.

To sum up, we chose Elixir as a language for writing compiler, and also as a base for Elixireum language.

V Conclusion

In conclusion, two problems were highlighted in this chapter. Firstly, the dominance of Solidity brings serious downsides to the Ethereum ecosystem. Secondly, the absence of a functional paradigm for smart contract development does not allow the usage of a functional paradigm strong pros for smart contract development. So, we decided to develop the Elixirium - a functional programming language based on Elixir for smart contract development.

Chapter 3

Methodology

...

Referencing other chapters 2, 3, 4, 5 and 6

TABLE II
Simulation Parameters

A	B
Parameter	Value
Number of vehicles	$\mid \mathcal{V} \mid$
Number of RSUs	$\mid \mathcal{U} \mid$
RSU coverage radius	150 m
V2V communication radius	30 m
Smart vehicle antenna height	1.5 m
RSU antenna height	25 m
Smart vehicle maximum speed	$\mathrm{v}_{\mathrm{max}}$ m/s
Smart vehicle minimum speed	$\mathrm{v}_{\mathrm{min}}$ m/s
Common smart vehicle cache capacities	$[50, 100, 150, 200, 250]$ mb

A	B
Common RSU cache capacities	$[5000, 1000, 1500, 2000, 2500]$ mb
Common backhaul rates	$[75, 100, 150]$ mb/s

Fig. 2. One kernel at $\mathrm{x_s}$ (dotted kernel) or two kernels at $\mathrm{x_i}$ and $\mathrm{x_j}$ (left and right) lead to the same summed estimate at $\mathrm{x_s}$. This shows a figure consisting of different types of lines. Elements of the figure described in the caption should be set in italics, in parentheses, as shown in this sample caption.

. . .

Chapter 4

Implementation

TABLE III
Simulation Parameters

A	B
Parameter	Value
Number of vehicles	$\mid \mathcal{V} \mid$
Number of RSUs	$\mid \mathcal{U} \mid$
RSU coverage radius	150 m
V2V communication radius	30 m
Smart vehicle antenna height	1.5 m
RSU antenna height	25 m
Smart vehicle maximum speed	$\mathrm{v}_{\mathrm{max}}$ m/s
Smart vehicle minimum speed	$\mathrm{v}_{\mathrm{min}}$ m/s
Common smart vehicle cache capacities	$[50, 100, 150, 200, 250]$ mb
Common RSU cache capacities	$[5000, 1000, 1500, 2000, 2500]$ mb
Common backhaul rates	$[75, 100, 150]$ mb/s

Fig. 3. One kernel at x_s (dotted kernel) or two kernels at x_i and x_j (left and right) lead to the same summed estimate at x_s. This shows a figure consisting of different types of lines. Elements of the figure described in the caption should be set in italics, in parentheses, as shown in this sample caption.

. . .

Chapter 5

Evaluation and Discussion

TABLE IV
Simulation Parameters

A	B
Parameter	Value
Number of vehicles	$\mid \mathcal{V} \mid$
Number of RSUs	$\mid \mathcal{U} \mid$
RSU coverage radius	150 m
V2V communication radius	30 m
Smart vehicle antenna height	1.5 m
RSU antenna height	25 m
Smart vehicle maximum speed	$\mathrm{v}_{\mathrm{max}}$ m/s
Smart vehicle minimum speed	$\mathrm{v}_{\mathrm{min}}$ m/s
Common smart vehicle cache capacities	$[50, 100, 150, 200, 250]$ mb
Common RSU cache capacities	$[5000, 1000, 1500, 2000, 2500]$ mb
Common backhaul rates	$[75, 100, 150]$ mb/s

Fig. 4. One kernel at $\mathrm{x_s}$ (dotted kernel) or two kernels at $\mathrm{x_i}$ and $\mathrm{x_j}$ (left and right) lead to the same summed estimate at $\mathrm{x_s}$. This shows a figure consisting of different types of lines. Elements of the figure described in the caption should be set in italics, in parentheses, as shown in this sample caption.

...

Chapter 6

Conclusion

TABLE V
Simulation Parameters

A	B
Parameter	Value
Number of vehicles	$\mid \mathcal{V} \mid$
Number of RSUs	$\mid \mathcal{U} \mid$
RSU coverage radius	150 m
V2V communication radius	30 m
Smart vehicle antenna height	1.5 m
RSU antenna height	25 m
Smart vehicle maximum speed	$\mathrm{v}_{\mathrm{max}}$ m/s
Smart vehicle minimum speed	$\mathrm{v}_{\mathrm{min}}$ m/s
Common smart vehicle cache capacities	$[50, 100, 150, 200, 250]$ mb
Common RSU cache capacities	$[5000, 1000, 1500, 2000, 2500]$ mb
Common backhaul rates	$[75, 100, 150]$ mb/s

Fig. 5. One kernel at $\mathrm{x_s}$ (dotted kernel) or two kernels at $\mathrm{x_i}$ and $\mathrm{x_j}$ (left and right) lead to the same summed estimate at $\mathrm{x_s}$. This shows a figure consisting of different types of lines. Elements of the figure described in the caption should be set in italics, in parentheses, as shown in this sample caption.

. . .

Bibliography cited

[1] V. Buterin. "Ethereum whitepaper." (2014), [Online]. Available: https://ethereum.org/en/whitepaper (visited on 10/25/2023).

[2] "Smart contract languages." (2022), [Online]. Available: https://ethereum.org/en/developers/docs/smart-contracts/languages/ (visited on 10/22/2023).

[3] "Github: Curated list of programming languages for blockchains." (2023), [Online]. Available: https://github.com/s-tikhomirov/smart-contract-languages#ethereum (visited on 10/22/2023).

[4] D. Harz and W. Knottenbelt, "Towards safer smart contracts: A survey of languages and verification methods," 2018. arXiv: 1809.09805 [cs.CR].

[5] W. Zou, D. Lo, P. S. Kochhar, et al., "Smart contract development: Challenges and opportunities," IEEE Transactions on Software Engineering, vol. 47, no. 10, pp. 2084–2106, 2021. DOI: 10.1109/TSE.2019.2942301.

[6] "Analyzing solidity and vyper for smart contracts programming." (2023), [Online]. Available: https://blockchain.oodles.io/blog/solidity-vyper-smart-contracts-programming/ (visited on 10/25/2023).

[7] "Language influences." (2021), [Online]. Available: https : / / docs . soliditylang . org / en / v0 . 8 . 22 / language - influences . html (visited on 10/29/2023).

[8] "Solidity." (2023), [Online]. Available: https://docs.soliditylang.org/en/v0.8.22/ (visited on 10/29/2023).

[9] "Vyper." (2020), [Online]. Available: https://docs.vyperlang.org/en/stable/ (visited on 10/29/2023).

[10] S. O'Grady. "The redmonk programming language rankings: January 2023." (2023), [Online]. Available: https://redmonk.com/sogrady/2023/05/16/language-rankings-1-23/ (visited on 10/29/2023).

Appendix A

Extra Stuff

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Appendix B

Even More Extra Stuff

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.