日本語スマートコントラクト記述の実現に向けて

渡邉 遥輔[1,a)] 松本 彩花[2,b)] 多田 拓[1,c)] 倉光 君郎[2,d)]

概要：近年注目を集める暗号通貨を支えるブロックチェーン技術の利用法としてスマートコントラクトがある．スマートコントラクトは従来の契約をブロックチェーン上で運用することでブロックチェーンの恩恵を受けられるものだ．しかしスマートコントラクトは従来の契約と異なり，その契約内容がプログラミング言語で書かれているという特徴がある．これは契約の締結時に必要な契約内容への同意を困難にし，スマートコントラクトで従来の契約を置き換える際の障害となる．そこで本研究ではこの課題に対し日本語でスマートコントラクトが記述可能なプログラミング言語 Nico を提案する．本言語はスマートコントラクトの内容を自然な日本語で記述可能とすることで多くのスマートコントラクト利用者が契約内容を確認できることを目指す．

キーワード：Ethereum，スマートコントラクト，Solidity，日本語プログラミング言語

1. はじめに

近年 Bitcoin や Ethereum などの暗号通貨が注目を集めている．これらはブロックチェーン技術を利用した分散台帳システムのうえに成り立っており，その性質として対改竄性の高さや取引記録の参照透明性の高さが挙げられる．このシステムは通貨の取引に限らず様々な取引，いわゆる契約に適用可能であると期待されており，いくつかの実証実験や試験導入も行われている．[1] このように分散台帳システム上で表される契約をスマートコントラクトといい，契約の自動履行が可能であったり記録や参照が容易であったりと様々な利点がある．

先述の Ethereum は Ethereum ネットワーク内で流注する暗号通貨であると同時にスマートコントラクト開発プラットフォームでもある．[2] Ethereum ネットワークに Solidity[3] というプログラミング言語で記述された契約（コントラクト）をデプロイすることでネットワーク参加者はその契約を履行することができるようになる．

ここで契約とは契約内容についての契約提示者と契約者間での合意形成と考えられ，契約者は契約内容を確認する必要がある．従来の契約は契約内容が自然言語で書かれているため契約者は契約内容を確認できるものとみなせる．しかしスマートコントラクトの場合は契約内容がプログラミング言語で記述されているため契約者が契約内容を直接確認することは難しく，現実的な対応として契約提示者にコントラクト内容を自然言語で記述したものを別途用意してもらう必要がある．しかしこの場合，契約者が合意形成する対象は自然言語で書かれた方の契約書となってしまい，これでは従来の契約方法と変わらない．つまり既存の契

1 横浜国立大学大学院理工学府
2 日本女子大学理学部数物科学科
a) watanabe-yosuke-wj@ynu.jp
b) matty1980315@gmail.com
c) tada-taku-jp@ynu.jp
d) kuramitsuk@fc.jwu.ac.jp

約を完全にブロックチェーン上に実装するためには契約者がスマートコントラクトを直接読めるようにする必要があると考える．そこで本研究では自然言語，特に今回は日本での利用を考え日本語によるスマートコントラクトの記述が可能な日本語プログラミング言語 Nico を提案する．本言語はスマートコントラクトのプログラムを比較的自然な日本語で記述することが可能であり，プログラミングの知識がなくともコードの内容を理解できるような文法として設計した．

本稿では Nico の設計と現在の開発成果について述べる．

2. Nico

Nico は日本語によるスマートコントラクトプログラムの記述が可能なプログラミング言語であり，その最大の目的はプログラミングの知識がない一般ユーザでも契約を表すコードの内容を直接把握することができるようにすることである．Nico は契約内容が記述される NicoContract ファイル (拡張子 nc) と，nc ファイル内に出現する単語に対する型が記述される NicoDictionary ファイル (拡張子 nd) で構成され，これらはコンパイルにより Solidity コードへと変換される．

2.1 言語デザイン

2.1.1 NicoContract

NicoContract では Solidity における状態変数定義と関数定義を記述する．Nico では Solidity でスマートコントラクトを記述する際に最低限必要になるであろうと考えられる構文を比較的自然な日本語の構文に置き換えて定義した．ここで，Solidity には複数ある記述方法のなかで推奨されない記述方法や使用の際に注意しなければならない構文が存在し，これらの要点は BestPractice としてまとめられている [4]．またこれらに起因するセキュリティバグに対する解析ツール [5] も存在するなどコントラクトの安全性に対する関心は高い．本言語では定義する構文を選択する際，推奨されない記法や注意の必要な構文を重点的に取り払うことでコントラクトの安全性についても考慮している．また Solidity は静的型付け言語であるが，型の存在はコントラクトコードを読む際にプログラミングに精通していないものを混乱させると考え，NicoContract からは型の表記を完全に取り払った．

2.1.2 NicoDictionary

NicoDictionary では Solidity における struct の定義と NicoContract 内に現れる単語と型の対応を記述する。また，この中で定義された単語をトークンとして構文解析器が補完されるため従来のプログラミング言語のように単語をスペースで区切らずとも構文解析を行うことができる．

2.2 トランスコンパイラ

本言語は Solidity ソースコードへのトランスコンパイラとして実装される．コンパイラ内部ではまず NicoDictionary をパースし，得られたトークンを用いて NicoContract のパーサを補完する．また Solidity コードの生成部においては Solidity の BestPractice に則ったコード補完を行うことで安全性を考慮したコードが生成できるような設計を考えている．コンパイラは現在開発中であり我々の Git レポジトリから入手することが可能である．
https://github.com/Caterpie-poke/nico

2.3 Nico のコード例

付録に Nico で簡単な仮想通貨を実装したコードを載せる．

3. まとめ

本稿では日本語によるスマートコントラクトの記述が可能であるプログラミング言語 Nico の設計について報告した．スマートコントラクトで従来の契約を置き換えるのであれば一般の利用者が内容を読める，確認できるという点は必要になるだろう．しかし何が読めるべきであるか，どのような機能が本言語に必要なのかという点を決定するために実契約の調査，分析が必要になるだろう．今後の展望として，既存の契約をスマートコントラクトとして実装するために必要な性質，機能を本言語に追加していきたい．

参考文献

[1] 東京海上日動火災保険株式会社，株式会社 NTT データ: 外航貨物保険の保険金請求へのブロックチェーン技術適用に向けた実証実験の完了，http://www.tokiomarine-nichido.co.jp/company/release/pdf/181101_01.pdf，2018.

[2] Ethereum Foundation. https://www.ethereum.org/，2018.

[3] Solidity Documentation. https://solidity.readthedocs.io/en/latest/，2018.

[4] Ethereum Smart Contract Best Practices. https://consensys.github.io/smart-contract-best-practices/，2018.

[5] Loi Luu, Duc-Hiep, Hrishi Olickel, Prateek Saxena, Aquinas Hobor: Making Smart Contracts Smarter. Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security，2016.

付　　録

ソースコード **1**　simple_coin.nc

```
「簡易通貨に関する契約」

第0条　記録される項目
    総発行量
    ～の残高

第1条　契約開始
    総発行量を10000とする
    あなたの残高を総発行量とする

第2条　総発行量の取得
    出力：総発行量

第3条　残高の取得
    入力：対象者
    出力：対象者の残高

第4条　送金
    入力：対象者、送金額
    要件
        対象者NOT=@0x0
        あなたの残高>=送金額
    あなたの残高を送金額だけ減らす
    対象者の残高を送金額だけ増やす
```

ソースコード **2**　simple_coin.nd

```
総発行量 : int
～の残高 : address -> int
対象者 : address
送金額 : int
```