B-036

代数仕様言語 CafeOBJ と証明支援系 Coq による STS プロトコルの形式化と検証
Fomal Specification and Verification of STS protocol by CafeOBJ and Coq

原 光太朗 † **緒方 和博** ‡ **中村 正樹** † **二木 厚吉** †
Kotaro Hara Kazuhiro Ogata Masaki Nakamura Kokichi Futatsugi

1. はじめに

セキュリティプロトコルの解析と検証を行う方法としては、形式手法が有効であると考えられている．形式手法は、システムを数学的にモデル化し、形式仕様の作成および仕様の形式的検証を行う手法で、厳密性、無矛盾性の点で優れており、これまでに多くの手法が提案されている．それぞれの手法は独自の理論や論理体系に基づいており、その形式化や検証は特徴を持ったものになっているにもかかわらず、手法の提案や実例を扱う研究に比べ、その比較を扱う研究はあまり行われていない．

本稿では、セキュリティプロトコルの一つである STS プロトコルを取り上げ、安全性と信頼性を有していることを形式手法により検証する．形式手法として CafeOBJ と Coq を用いる．この具体例を通して、双方によるシステムの形式化および検証の、手法についての比較を行い、長所・短所を明らかにすることにより、形式手法に関する新たな指針を示すことを目指す．

2. 代数仕様言語 CafeOBJ

CafeOBJ[§]は、等式論理に基づく代数仕様言語で、可読性の高い形式仕様の記述を主目的とし、項書き換えによる等式推論をもとに検証を行うこともできる．

CafeOBJ による形式手法の例として、bank account 抽象機械の記述を行う．CafeOBJ による状態機械の記述は、システムに関する値の変化を表す、等式を用いて記述する．

Nat は自然数を表すソートであり、<=,<,+,-などの演算とともに輸入されるモジュール INT で既に定義されている．Account は状態空間を表すソートである．init は初期状態を表す．balance は口座の残高を表す演算で、deposit および withdraw は口座に対する預金および引き出しを表す演算である．A は状態を，N は自然数を表すとすると，balance(A) は状態 A における残高を、deposit(A,N) は状態 A で金額 N を預金した後の状態，withdraw(A,N) は状態 A で金額 N の引き出しを試みた後の状態を表す．初期状態では残高は 0 で，等式で規定する．deposit および withdraw の適用により残高がどのように変化するかを等式で規定する．R は状態の等価性を判定する演算であり、等式で規定する．

```
mod* ACCOUNT { pr(INT) *[Account]*
  op init : -> Accout
  bop balance : Account -> Nat
  bop deposit : Account Nat -> Account
  bop withdraw : Account Nat -> Account
  var N : Nat  vars A A1 A2 : Account
  eq balance(init) = 0 .
  eq balance(deposit(A,N)) = balance(A) + N .
  ceq balance(withdraw(A,N)) = balance(A) - N
                       if N <= balance(A) .
  ceq balance(withdraw(A,N)) = balance(A)
                       if balance(A) < N .
  bop _R_ : Acount Accout -> Bool
  eq (A1 R A2) = (balace(A1) == balance(A2)) .
}
```

性質の記述、検証についても考える．この抽象機械に対する操作に関して、以下のような性質がある．

inv : ある状態において引き出し操作が有効ならば、その後の預金の操作と入れ替えられる．

CafeOBJ においてこの性質は、状態空間を表すソートの CafeOBJ 変数 A を含んだ項として以下のように表される．また、検証は処理系において、帰納法を用いた証明を記述した証明譜を実行することによって行う．

```
eq inv(A) = N <= balance(A) implies
            deposit(withdraw(A,N),M) R
            withdraw(deposit(A,M),N) .
```

3. 証明支援系 Coq

Coq[¶]は、高階論理に基づく証明支援系で、帰納的定義によりシステムを記述し、そこから得られる原理等をもとに半自動的な検証を行うことを主目的としている．

Coq による形式手法の例として、CafeOBJ と同様の例の記述を行う．Coq による状態機械の記述は、状態空間を表す集合と、ある状態におけるシステムに関する値を示す述語を、帰納的に定義することにより記述する．

自然数の集合 nat および自然数に関する演算<=,<,+,-は、既に通常の意味で定義されているものとする．状態空間 Account を帰納的に定義する．Account の要素を状態と呼ぶことにする．init は初期状態を表す．a は状態を，n は自然数を表すとすると，(deposit a n) は状態 a で金額 n を預金した後の状態，(withdraw a n) は状態 a で金額 n の引き出しを試みた後の状態を表す．口座の残高は，状態と自然数を引数にとる述語 balance で表す．(balance a n) は状態 a における残高が金額 n であることを表している．R は状態の等価性を判定する演算であり、意味は CafeOBJ と同じものとし、定義は省略する．

```
Inductive Account : Set := init : Account
| deposit : Account -> nat -> Account
| withdraw : Account -> nat -> Account.

Inductive balance : Account -> nat -> Prop :=
  bal_init : (balance init 0)
| bal_depo : (a:Account;n,m:nat)
              (balance a n) ->
              (balance (deposit a m) n+m)
| bal_withT : (a:Account;n,m:nat)
               (balance a n) /\ m<=n ->
               (balance (withdraw a m) n-m)
| bal_withF : (a:Account;n,m:nat)
```

†北陸先端科学技術大学院大学, JAIST
‡NEC ソフトウェア北陸/北陸先端科学技術大学院大学,
NEC Software Hokuriku, Ltd./JAIST
§http://www.ldl.jaist.ac.jp/

¶http://pauillac.inria.fr/coq/

```
(balance a n) /\ n<m ->
(balance (withdraw a m) n).
```

性質 inv は状態空間を表すデータ型の変数宣言を含んだスキーマとして以下のように表される．また、検証は処理系において、帰納法を使うなどの証明の操作を示すコマンドを入力することによって行う．

```
Theorem inv : (a:Account)(n,m,l:nat)
(balance a l) /\ l>=n ->
(deposit (withdraw a n) m)R
(withdraw (deposit a m) n).
```

4. STS プロトコル

STS(station-to-station) プロトコルは、1992 年に W. Diffie, P.van Oorschot, M.Wiener らによって、共通鍵暗号方式を用いた認証鍵交換プロトコルとして提案されたプロトコル [1] で、Diffie-Hellman 鍵交換プロトコルに存在する脆弱性への対抗を主目的としている．具体的には、通信を行う双方が、デジタル署名と公開鍵証明書を使用することにより、お互いを認証することで目的を実現している．

通信を行う各主体は、共通に信頼できる認証局 T から自分の名前と公開鍵を含んだ証明書を受け取れる、と仮定すると、STS プロトコルの一つのセッションは次の 3 ステップで記述できる．ここで信頼できる認証局とは、主体の名と公開鍵の組を、間違いなく認識できる認証局とする．

1. $A \rightarrow B:\ p, EX_A$
2. $B \rightarrow A:\ EX_B, Cert_B, \{\{EX_A, EX_B\}_{s_B}\}_{K_{AB}}$
3. $A \rightarrow B:\ Cert_A, \{\{EX_A, EX_B\}_{s_A}\}_{K_{AB}}$

$$Cert_X = \{X, p_X, p\}_{s_T}$$

p は D.H. パラメータ (十分大きな素数 q とそれ以下の整数 g の組) を表し、EX_X は X(秘密値 x を保有) の公開値 ($g^x \ mod\ q$) を表す．K_{XY} は X と Y(秘密値 y を保有) との間の共通鍵 ($g^{xy} \ mod\ q$) を表し、これにより暗号化した文が $\{\cdots\}_{K_{XY}}$ である．また、$\{\cdots\}_{s_X}$ は X により署名された文を表し、p_X, $Cert_X$ はそれぞれ X の公開鍵、公開鍵証明書を表す．

鍵交換後、各主体は交換した鍵で暗号化を行うことにより、メッセージを送信する．そのメッセージを暗号通信文と呼ぶことにする．

5. STS プロトコルの検証

CafeOBJ と Coq の双方で、STS プロトコルを状態機械としてモデル化した後、次の二つの表明が成り立つことをそれぞれで検証する．各々の場合でのモデル、表明の記述、および証明の詳細は割愛する．

表明 1 侵入者は、他人に向けられた暗号通信文を解読できない (安全性)

表明 2 プロトコルの正式な参加者は正しく認証鍵を交換する (信頼性)

6. CafeOBJ と Coq の比較

以上の検証を通して、形式手法における CafeOBJ と Coq の特徴について考える．

• 形式化に関して

CafeOBJ では、自然言語ほどではないものの、システムの記述を、オブジェクトの意味を直接捉える方程式の集合によって記述するという、直感的な形で行うことができ、仕様として可読性の高い形式化を行うことができる．しかしそのため、多少冗長になる部分もある．一方 Coq では、システムを集合論と述語論理に基づいた構造化された論理体系として捉えるため、記述は型制約などの制限が多く、構文論的な記号の記述となり、直感的な形式化を行うことはできないが、その分記述量を抑えることができる．STS プロトコルの記述においては性質の記述を含めて、CafeOBJ では 21k バイト、Coq では 16k バイトであった．

• 処理系における検証に関して

CafeOBJ では、事実を表す等式集合と簡約を行う項といったいわば証明そのものを記述した証明譜の実行を行うのに対し、Coq では、タクティクと呼ばれるコマンドの系列という一連の証明構築操作を記述したスクリプトの実行を行う、といった特徴がある．前者は可読性、編集容易性の面では優れているが、等式の追加などの証明譜の変更が、検証にどの程度影響を与えるのかわかりづらく、証明の一部の書き換えと処理系での実行を何度も繰り返して行う必要がある．後者は、対話的に証明を進められることや証明の再利用の面では優れているが、可読性や編集の自由度が低く、また証明の構造も把握しにくい．

• 定理等の証明に関して

CafeOBJ では、検証の基本機構が一方向の等式推論であるため、検証系が複雑になり過ぎない．その一方、場合分けが検証を行う者の経験に依存するという問題がある．Coq では、場合分けや帰納法の仮定等は定義により自動生成される．その一方証明を行うには、適切な仮定やコマンドの選択等に、ある程度の経験が必要となる．

7. まとめと今後の課題

セキュリティプロトコルの一つである STS プロトコルの振る舞いを、CafeOBJ および Coq で記述した．また、それらの記述が安全性と信頼性を示す性質を満たしているかどうかの検証を、それぞれで証明することにより行った．さらに具体例を通して形式化および検証を行ったことで、それぞれの手法の特徴を捉えることができた．

今後の課題としては、残りの補題の証明やさらなる考察を行うことなどが挙げられる．また、SSH 等の現在利用されているようなプロトコルを扱うのも今後の課題であろう．

参考文献

[1] W. Diffie, P. van Oorschot and M. Wiener, *Authentication and authenticated key exchanges*, Designs, Codes and Cryptography, 2, 107-125, 1992

[2] 緒方和博, 二木厚吉, 書き換えによるセキュリティプロトコルの帰納的検証, コンピュータソフトウェア, vol.20, No3, pp.54-72, 2003.