帰納的アプローチに基づく理想的電子現金方式のモデル化および証明支援系 Isabelle/HOL による安全性の証明

吉丸 始須雄[†1] 高橋 和子[†1]

本研究では，帰納的アプローチに基づいて理想的電子現金方式をモデル化し，その安全性について，証明支援系 Isabelle/HOL を用いて証明を与える．

理想的電子現金方式とは,「完全情報化」,「安全性」,「プライバシー」,「オフライン性」,「譲渡可能性」,「分割利用可能性」の 6 条件を満足する電子現金プロトコルである．この安全性は,「電子現金のコピー，偽造等による不正利用ができないこと」と定義されている．本研究の対象とするプロトコルは，電子現金のデータ構造として二分木を採用しており，支払いは金額に相当するノードを使うと定義される．また，同じ枝にあるノードを使用すると，額面以上の金額を使うという不正利用が生じるように設計されている．通常はユーザーの購買に関するプライバシーを銀行を含む他者が把握することはできないが，不正利用が発覚した際に，使用されたノードの情報から不正利用者を割り出すことができる仕掛けになっており，これによって安全性を保証している．

プロトコルのモデル化には，Paulson らの提案した帰納的アプローチを用いる．しかし，彼らの構築したモデルは金額のような量を含むプロトコルは扱っていない．一方，このプロトコルにおける安全性を扱うには，二分木上のノードの位置情報と電子現金の金額を一体化するようなモデル化が必要である．本研究では，これらの情報を量的なデータとしてモデルに与えた場合において本アプローチが有効であるかどうかを確かめ，その結果本アプローチにおける問題点を明らかにした．

Modeling an Ideal Electronic Cash Scheme Based on an Inductive Approach and Proving Its Security by a Proof Assistant Isabelle/HOL

Shizuo Yoshimaru[†1] and Kazuko Takahashi[†1]

We make a model of an electronic cash (e-cash) scheme based on an inductive approach and prove its security using a proof assistant Isabelle/HOL.

An ideal e-cash scheme is a protocol that satisfies the following properties: independence, security, untraceability, off-line operation, transferability and divisibility. Among these properties, we focus on security: "no overspending is allowed."

Our target protocol adopts a binary tree as a data structure of an e-cash, and a payment is defined as spending a node corresponding to the amount. It is designed so that overspending appears if a user spends a pair of nodes in the same branch. Usually the other agents including a bank cannot know a private information on payment. However, once overspending appears, the malicious user can be identified from the information of the spent nodes, which guarantees security.

We adopt an inductive approach proposed by Paulson et. al to model this protocol. However, protocols that include quantitative data such as amount of payment were not handled in their model. It is necessary to make a model that unifies the positional information of a node in a binary tree and an amount of the payment as a natural number. In this presentation, we examine the effectiveness of the inductive approach on a protocol that includes quantitative data. We discuss the problems that we found in our investigation.

1. はじめに

システムが仕様を満たしているかどうかを確認する手段として，数理的技法と呼ばれる検証手法が注目されている．これは，膨大なテストケースを使ってシステムを実際に動かすことによって安全性を保証する手法と異なり，計算機を利用することによって，代数や論理に基づいた検証を強力かつ効率的に行う．この数理的技法は，モデル検査と定理証明という 2 つの枠組みに分けられる．モデル検査とは，システムの動作を状態遷移モデルとして記述し，起こり得る全ての遷移について自動的に探索を行うことによって，システムに求められる性質が成り立つかどうかを検証するというものである[3),4)]．一方定理証明とは，システムを数学的モデルとして記述した上で，システムに求められる性質が成り立つことを，公理と推論規則に従って証明するというものである．代表的な定理証明器として Isabelle/HOL[9)], ACL2[7)], PVS[6)], Coq[5)] などがあるが，ここでは Isabelle/HOL を用いた定理証明を扱う．

Isabelle/HOL は，高階論理 HOL (Higher-Order Logic) の体系に基づき，証明すべき定理を後方推論によって半自動的に導く対話型証明支援システムである．証明は，システムが提示するサブゴールに対し，適用する推論規則をユーザが逐一指定することによって行われる．また，Isabelle/HOL は帰納的推論を得意としているため，システムをモデル化する場

†1 関西学院大学大学院理工学研究科
School of Science&Technology, Kwansei Gakuin University

合，帰納的モデルとして記述するのが一般的である[8),15)]．

Isabelle/HOL の開発者でもある Paulson らは，暗号プロトコルを帰納的にモデル化することにより，安全性の証明を帰納的に与えるアプローチを提案した[13)]．その後，このアプローチにより，Kerberos や TLS, SET などの複雑なプロトコルも証明され[1),2),14)]，暗号プロトコルの安全性の証明は Isabelle/HOL の得意とする分野のひとつとなった．一方，暗号プロトコルのひとつである電子現金プロトコルにおいて，安全性の証明には支払い金額などの量的な情報の考慮が不可欠である．しかし，これまでに Isabelle/HOL によって安全性が証明された暗号プロトコルは，量的な情報を考慮する必要がなかったため，電子現金プロトコルのようなプロトコルの安全性についての証明手法は確立していない．そこで，本研究では電子現金プロトコルのひとつである理想的電子現金方式[10),11)] を対象とし，金額やノードの位置情報などの量的な情報を考慮したモデル化を行うい，安全性の証明を与える．これにより，Isabelle/HOL の有効範囲を明らかにすることが，本研究の目的である．

著者らはこれまでに，理想的電子現金方式の性質のひとつである分割利用可能性について，既にデータ構造レベルでのモデル化を行った上で，Isabelle/HOL による証明を与えた[16)]．このとき，二分木と自然数の帰納スキームの相違が問題となったが，中間データとして二進数を用意することにより，この問題を解決した．今回の研究においても，二分木の帰納スキームに関する問題に直面した．本論文では，これについて議論を行う．

本論文の構成は以下の通りである．第 2 節では，理想的電子現金方式の 6 条件について触れた後，提案された二分木構造の電子現金を紹介し，更にそのプロトコルについて説明する．第 3 節ではプロトコルのモデルを与え，第 4 節で安全性の検証を行った後，第 5 節にて議論を行う．そして最後に，第 6 節で結論を述べる．

2. 理想的電子現金方式

現在世の中で使われている電子現金方式には，IC カードなどに電子情報として収納することによってオフラインで利用する方法や，クレジットカードのように後日決済する方法，小売店が銀行にリアルタイムで問い合わせることによって電子現金を利用する方法などがある．しかし，それぞれ「物理媒体に依存している」,「利用者のプライバシーが保障されない」,「通信コストや処理時間が大きくなってしまう」などの問題点が存在する．それらの問題を解決するため，岡本らによって理想的電子現金方式が提案された[10),11)]．

2.1 理想的電子現金方式の定義

理想的電子現金方式とは，いかなる物理媒体にも依存せず，情報そのものが電子現金となるような形で電子財布に格納される形態でありながら，オフラインでの利用やプライバシーの遵守などを実現する，まさに理想的な電子現金方式のことである．しかしながら，情報そのものは極めて容易に全く同一のコピーが可能であり，何の痕跡も残さずにデータを改竄することさえ可能であるため，不正利用の防止にも力を入れる必要がある．

岡本らは，理想的電子現金方式の満たすべき性質として，以下の 6 条件を定義している[11),12)]．

（1） 完全情報化
現金が完全に情報のみで自立して実現されていること（つまり，通信回線で電子現金を転送できる）．

（2） 安全性
電子現金のコピー，偽造等による不正利用ができないこと．

（3） プライバシー
利用者の購買に関するプライバシーが，小売店や電子現金発行・決済者（ここでは銀行とする）が結託しても露見しないこと．

（4） オフライン性
小売店での電子現金支払い時の処理は，オフライン処理でできること．

（5） 譲渡可能性
電子現金の他人への譲渡が可能であること．

（6） 分割利用可能性
1 回発行された電子現金を，利用合計金額が額面の金額になるまで何回でも使うことができること．

これら 6 条件のうち，前の 4 条件は理想的電子現金の本質となるものであり，後の 2 条件は電子現金の利便性を考えると当然要求されるものである．以下の小節では，6 条件を全て満たすよう提案された岡本らの方式[10)] について解説する．

2.2 理想的電子現金のデータ構造

岡本らは，分割利用可能性を効率的に実現するために，理想的電子現金のデータ構造として二分木を採用した．二分木の各ノードの該当金額は，直下の子ノードの該当金額を合計したものになっている．利用単位金額（たとえば 1 円単位など）は葉ノードの該当金額であり，その値は目的に応じて銀行が定めれば良い．たとえば，25 円単位で，100 円の電子現金を用意した場合，図 1 のような二分木となる．

なお，ノードの使用については，次のルールによって制限されている．

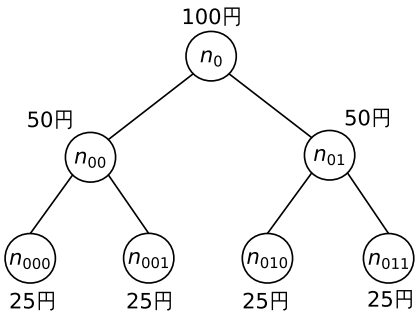

図 1 二分木構造の電子現金
Fig. 1 An Electronic Cash as a Binary Tree

Route node rule
　あるノードが一度使われた後，そのノードと連結するすべての祖先ノードおよび子孫ノードを使用してはならない．
Same node rule
　各ノードは，1 回以上使用してはならない．

このルールに従うと，図 1 の電子現金からノード n_{00} を使用した後で，使用できるノードは n_{01}, n_{010}, n_{011} の 3 つのみとなる．もちろん，残りを全て使用できるわけではなく，n_{01} を使用する場合は子ノードの 2 つが，n_{010} か n_{011} あるいはその両方を使用する場合は親ノードである n_{01} が，それぞれ使用できなくなる．つまり，上記のルールに従うことにより，額面の 100 円以上の使用ができないようになっており，更に分割利用可能性も実現している．

理想的電子現金方式では，安全性やプライバシーを満足するため，支払い時に使用するノード全てについて何らかの処理を行う必要がある．たとえば，利用単位金額をリストとして並べたような単純な構造の電子現金を仮定した場合，処理にかかる時間は単純に金額に比例することになる．一方，このように二分木構造を用いた場合，選択したノードに対してだけ処理を行えば良く，処理時間は格段に短縮される．また，額面金額を n とすると，二分木の深さはおよそ $\log_2 n$ となるため，ノード選択の効率も良い．以上の理由により，二分木構造の電子現金は，理想的電子現金方式を実現する上で最も理に適っていると言える．

2.3 理想的電子現金方式のプロトコル

理想的電子現金方式では，銀行 B，利用者 U，小売店 V という三者間での通信を規定している．B は利用許可証 L の発行と電子現金 C の発行，および決済を担当し，U と V は電子現金を利用して売買を行うものとする．

プロトコルは，大きく 4 つのフェーズに分かれている．U が B から利用許可証の発行を受けるフェーズ，U が B から電子現金の発行を受けるフェーズ，U と V の間で発生する支払いのフェーズ，そして V が B に対して行う決済のフェーズである．ここでポイントとなるのが，支払いに充てるノードを $n_{0j_1\ldots j_t}$ としたとき[*1]，U と V の間でやり取りされる $\Gamma_{j_1\ldots j_t}$ と $\Lambda_{j_1\ldots j_t}$ である．$\Gamma_{j_1\ldots j_t}$ は Route node rule に違反するノードの使用情報から不正利用者を特定するための値であり，$\Lambda_{j_1\ldots j_t}$ は Same node rule に違反するノードの使用情報から不正利用者を特定できるように設定された値である．この添字の $0j_i\ j_1\ldots j_t$ はノードの位置情報を表しており，先頭の 0 がルートノード，そこから順に j_i $(i=1,\ldots,t)$ の値が 0 なら左の子ノードへ，1 なら右の子ノードへと進み，最終的に辿り着いたノードが該当のノードとなっている（図 1 参照）．このプロトコルは数論に基づいて設計されており，決済のフェーズにおいて不正利用が発覚すると，そこに使用されている合成数 N から，その素因数 P, Q の値を $\Gamma_{j_1\ldots j_t}$ と $\Lambda_{j_1\ldots j_t}$ を用いて割り出せるようになっている．利用許可証の発行のフェーズにおいて，この P と Q から導いた値を U の識別情報として B に登録しているため，B は P と Q を割り出すことにより，不正利用者を特定することができる．

3. モデル化

プロトコルのモデル化は，Paulson の帰納的アプローチ[13] に基づく．ここでは，Bella らによる SET プロトコルのモデル[1] を参考に，理想的電子現金プロトコルのモデル化を行った．

まず，プロトコルに登場するエージェントを以下のように *agent* 型として定義する．

```
datatype
  agent = Bank            — 銀行
        | Customer nat    — 利用者
        | Shop nat        — 小売店
```

Customer と *Shop* は自然数 *nat* 型を引数に取ることにより，自然数への全単射となる

⋆1 通常，支払いに充てるノードは複数あるが，複数ノードによる支払い処理は，単一ノードによる支払い処理を並列に実行するだけで良い．ここでは単一ノードに対する支払い処理のみを示す．

ため，無限に存在する．

次に定義する *msg* 型は，エージェントによって送受信されるメッセージを表す．

```
datatype
    msg = Agent  agent          — エージェントの名前
        | Number nat            — タイムスタンプなどの推測可能な数値
        | Nonce  nat            — 推測不可能な数値
        | Pan    nat            — アカウント番号
        | Key    key            — 暗号鍵
        | Hash   msg            — ハッシュ化
        | MPair  msg msg        — メッセージの結合
        | Crypt  key msg        — 暗号化
        | NPos   "bool list"    — ノードの位置情報
```

Key の引数である *key* 型は，自然数 *nat* 型の別名宣言となっている．*NPos* は引数に真偽値 *bool* 型のリストを取ることにより，ノード $n_{0j_1\ldots j_t}$ の位置情報を表すものとして，本研究で新たに定義した識別子である．また，*MPair* について，以降は *MPair X Y* を {|*X*, *Y*|} と略記し，{|*X*, {|*Y*, *Z*|}|} を {|*X*, *Y*, *Z*|} と略記する．

プロトコルにおけるイベントは，次のように *event* 型として定義されている．

```
datatype
  event = Says  agent agent msg  — Says A B X で A が B に X を送信
        | Gets  agent       msg  — Gets A X で A が X を受信
        | Notes agent       msg  — Notes A X で A が X を保管
```

プロトコルは *event* 型のリストの集合として定義され，起り得る全てのイベント列がその集合に含まれる．ここでは，理想的電子現金プロトコルを集合 *iecp* として定義する．

```
inductive_set
  iecp :: "event list set"
where
  Nil:  — イベントが何も起っていない状態
  "[] ∈ iecp"

| Reception:  — B に送信された X を B が受信
  "⟦ evsr ∈ iecp;  Says A B X ∈ set evsr ⟧
   ⟹ Gets B X # evsr ∈ iecp"
```

[] は空リスト，*evsr* は *Reception* におけるイベント列を表すリストである．また，*set* はリストを集合として扱う関数であり，*x # xs* はリスト *xs* の先頭に要素 *x* を追加する操作である．ここでは，次に起こるイベントを逐一イベント列に追加することによって，プロトコル *iecp* を再帰的に定義している．

なお，この定義には，上記の *Nil* と *Reception* に加え，利用許可証の発行処理をモデル化した *OPstart*, *OP1*, *OP2*，電子現金の発行処理をモデル化した *WPstart*, *WP1*, *WP2*，電子現金の支払いをモデル化した *CA*, *DR1*, *DR2*, *DR3*, *DR4*，決済における不正利用者の特定をモデル化した *DORn*, *DOSn* が続く．ここでは，電子現金の支払いのフェーズのみを紹介する．

3.1 電子現金の支払いのモデル化

ここではまず，電子現金の正当性を検査すると共に，通信相手を確定し，どのノードを使用するのかについて合意しておく．検査に合格すれば，利用者 U と小売店 V は互いに電子現金の情報 *CoinInfo* を保管する．

なお，ノードの位置情報は *NPos bs* として指定する．*bs* は真偽値のリストであり，ノード $n_{0j_1\ldots j_t}$ の j_i $(i = 1, \ldots, t)$ の 0, 1 がそのまま *True*, *False* に対応している．また，電子現金の金額は 2^l 円であり，深さ l の二分木を生成するため，位置情報を表すリストの長さは l 以下でなくてはならない．

```
| CA: — 電子現金の正当性の検査
  "⟦ evsCA ∈ iecp;  V = Shop j;
    Gets U C ∈ set evsCA;
    C = Crypt (priCK w) (Hash {|Key N, Nonce b|});
    Gets U L ∈ set evsCA;
    L = Crypt (priSK Bank) (Key N);
    Notes U (Key (invKey N)) ∈ set evsCA;
    Notes U {|Number WP, Nonce b|} ∈ set evsCA;
    TransWP = {|Agent Bank, Agent U, Number w|};
    Notes Bank {|Number WP, TransWP|} ∈ set evsCA;
    w = 2^l;  length bs ≤ l;
    Number DR ∉ used evsCA;
    CoinInfo = {|L, Key N, C, Nonce b, Number w|};
    TransDR = {|Agent V, Agent U, CoinInfo|} ⟧
```

```
    ⟹ Notes U {|Number DR, NPos bs, TransDR|}
        # Notes V {|Number DR, NPos bs, TransDR|}
        # evsCA ∈ iecp"
```

続いて，$\Gamma_{j_1 \ldots j_t}$ と $\Lambda_{j_1 \ldots j_t}$ をモデル化する．$\Gamma_{j_1 \ldots j_t}$ はノードの位置情報の違いによって，$\Lambda_{j_1 \ldots j_t}$ は e の違いによってそれぞれ N の素因数を割り出せるように設計されており，それらを生成できるのは N の素因数 P, Q を知る U のみである．それらを踏まえて，次のように *Gamma* と *Lambda* を記述する．

```
  Gamma = Crypt P {|NPos bs, Hash {|C, Key N|}|}
  Lambda = Crypt P {|e, Hash {|C, NPos bs, Key N|}|}
```

この *Gamma* と *Lambda* を用いて，支払い処理を以下のようにモデル化した．

```
 | DR1:  — U が Γ_{j1...jt} を V に送信
   "⟦ evsDR1 ∈ iecp;
     Gamma = Crypt (invKey N) {|NPos bs, Hash {|C, Key N|}|};
     Notes U {|Number DR, NPos bs, TransDR|} ∈ set evsDR1;
     CoinInfo = {|L, Key N, C, Nonce b, Number w|};
     TransDR = {|Agent V, Agent U, CoinInfo|} ⟧
    ⟹ Says U V Gamma
        # evsDR1 ∈ iecp"

 | DR2:  — V が Γ_{j1...jt} の正当性を検査して e を U に送信
   "⟦ evsDR2 ∈ iecp;
     Gets V Gamma ∈ set evsDR2;
     Gamma = Crypt (invKey N) {|NPos bs, Hash {|C, Key N|}|};
     Number T ∉ used evsDR2;  Nonce e' ∉ used evsDR2;
     e = Hash {|Pan (pan V), Number T, Nonce e'|};
     Notes V {|Number DR, NPos bs, TransDR|} ∈ set evsDR2;
     CoinInfo = {|L, Key N, C, Nonce b, Number w|};
     TransDR = {|Agent V, Agent U, CoinInfo|} ⟧
    ⟹ Says V U {|Number DR, Pan (pan V), Number T, Nonce e'|}
        # Notes V {|Number DR, e|}
        # evsDR2 ∈ iecp"

 | DR3:  — U が Λ_{j1...jt} を V に送信
   "⟦ evsDR3 ∈ iecp;
     Gets U {|Number DR, Pan IDV, Number T, Nonce e'|} ∈ set evsDR3;
     e = Hash {|Pan IDV, Number T, Nonce e'|};
     Lambda = Crypt (invKey N) {|e, Hash {|C, NPos bs, Key N|}|};
     Notes U {|Number DR, NPos bs, TransDR|} ∈ set evsDR3;
     CoinInfo = {|L, Key N, C, Nonce b, Number w|};
     TransDR = {|Agent V, Agent U, CoinInfo|} ⟧
    ⟹ Says U V Lambda
        # evsDR3 ∈ iecp"

 | DR4:  — V が Λ_{j1...jt} の正当性を検査して通信記録を銀行 B に送信
   "⟦ evsDR4 ∈ iecp;
     Gets V Lambda ∈ set evsDR4;
     Lambda = Crypt (invKey N) {|e, Hash {|C, NPos bs, Key N|}|};
     Gets V Gamma ∈ set evsDR4;
     Gamma = Crypt (invKey N) {|NPos bs, Hash {|C, Key N|}|};
     Notes V {|Number DR, NPos bs, TransDR|} ∈ set evsDR4;
     CoinInfo = {|L, Key N, C, Nonce b, Number w|};
     TransDR = {|Agent V, Agent U, CoinInfo|} ⟧
    ⟹ Says V Bank (Crypt (priSK V) {|CoinInfo, NPos bs, Gamma, Lambda|})
        # evsDR4 ∈ iecp"
```

V は最後に通信記録として，{|*CoinInfo*, *NPos bs*, *Gamma*, *Lambda*|} に自分の署名を付けて *Bank* に送信することで，決済としている．それぞれ，*CoinInfo* は電子現金の情報，*NPos bs* は使用したノードの位置情報，*Gamma* と *Lambda* は不正利用者の特定に必要な情報であり，通信記録として過不足ないものと解釈する．

4. 検　　証

理想的電子現金方式における安全性の条件のひとつに「額面の金額を越えた支払いが行

われないこと」が挙げられている[10]．このプロトコルでは，額面の金額を越えた支払いを行った者がいた場合，銀行がその不正利用者を必ず特定できるよう設計することで，上記の条件を満足するものとしている．本モデルにおいて，Route node rule か Same node rule のどちらかに違反した決済が発生すると，銀行が不正利用者の *U* の *Key P* を入手し，*U* のアカウント番号 *Pan (pan U)* を特定する．ここでは，額面の金額を越えた支払いが行われた場合，Route node rule か Same node rule のどちらかに違反した決済が発生していることを示す．

theorem *overspending:*

```
"⟦ evs ∈ iecp;
   ∃bs Gamma Lambda.
     {|CoinInfo, NPos bs, Gamma, Lambda|} ∈ knows Bank evs;
   CoinInfo = {|L, Key N, C, Nonce b, Number (2^l)|};
   payment_amount l (payment_node CoinInfo evs) > 2^l ⟧
 ⟹ ∃ev1 ∈ set evs. ∃ev2 ∈ set evs.
      check_overspending CoinInfo ev1 ev2"
```

knows A evs はイベント列 *evs* におけるエージェント *A* の知識の集合，つまり *A* が *Says*, *Gets*, *Notes* を行ったメッセージ全てを含む集合である．また，*payment_amount l bss* は 2^l 円の電子現金を表す二分木に対し，リスト *bss* 中に列挙したノードの金額を合計した値を返す関数，*payment_node CoinInfo evs* はイベント列 *evs* 中の *CoinInfo* による支払いに使用されたノードを全て列挙する関数である．つまり上記の前提部は，「*Bank* が受け取った *CoinInfo* について，支払いの合計金額が 2^l 円を越えた場合」となっている．

また，結論部の *check_overspending* は補助関数である．イベント *ev1* と *ev2* が *CoinInfo* に対応する決済であった場合に，2 つのイベントが Route node rule か Same node rule のどちらかに違反すること，つまり使用された 2 つのノードが同じ枝のものかどうかを検査するものである．つまり上記の結論部は，「Route node rule か Same node rule のどちらかに違反するような決済が存在する」ことを表している．*check_overspending* の定義は次の通り．

definition

```
check_overspending :: "msg ⇒ event ⇒ event ⇒ bool"
```

where

```
"check_overspending CoinInfo ev1 ev2 ≡
   (∃X1 X2 bs1 bs2.
      X1 ≠ X2 ∧
      ev1 = Gets Bank X1 ∧
      ev2 = Gets Bank X2 ∧
      pick_NPos CoinInfo X1 = (bs1,True) ∧
      pick_NPos CoinInfo X2 = (bs2,True) ∧
      check_sublist bs1 bs2)"
```

check_sublist は，引数のリスト 2 つの要素が等しいかどうかについて先頭から順に比較していき，どちらか一方が末尾まで辿り着けば，もう一方に対して先頭からの部分リストであるとして *True* を返すという関数である．これをノードの位置情報 *bs1*, *bs2* が満たすとき，それらは同じ枝のノードであると言える．

なお，定理 *overspending* の証明のために，以下の補題を用意する．この補題は，*check_sublist* を満たす 2 つのノードが存在するための十分条件を示している．

lemma *case_of_exists_sublist:*

```
"⟦ max_list (map length bss) ≤ l;
   payment_amount l bss > 2^l ⟧
 ⟹ ex_2elem_list check_sublist bss"
```

max_list はリストの要素の最大値を求める関数，*length* はリストの長さを求める関数であり，*map f xs* は 関数 *f* をリスト *xs* の全ての要素に適用し，その結果をリストに格納して返す関数である．つまり，リスト *bss* の要素であるノードの位置情報の長さが全て l 以下であり，それらによる支払いの合計金額が 2^l 円を越えている場合，*check_sublist* を満たすような 2 つのノードが存在することを表している．

上記はノードの位置情報をリストとして表現した場合の性質であるが，これは二分木上の性質であるため，二分木として表現し，その上で証明するのが自然である．そのため，上記を二分木上の性質として記述し直し，補題の証明を試みる．

ノードを使用した回数を各ノードの属性値として設定した二分木を用意する．リスト *bss* からマッピングを行うため，未使用の 2^l 円の電子現金に対して，リスト *bss* 内の位置情報の指すノード全てに 1 ずつ加える関数 *all_node_mapping l bss* を定義した（図 2 参照）．たとえば，リストの 1 番目の要素である *[True, False, True]* はノード n_{0010} に対応しており，2 番目の要素である *[False]* はノード n_{01} に対応している．そして，このような二分木から使用金額を求める関数 *all_node_amount* を定義したところ，以下のよう

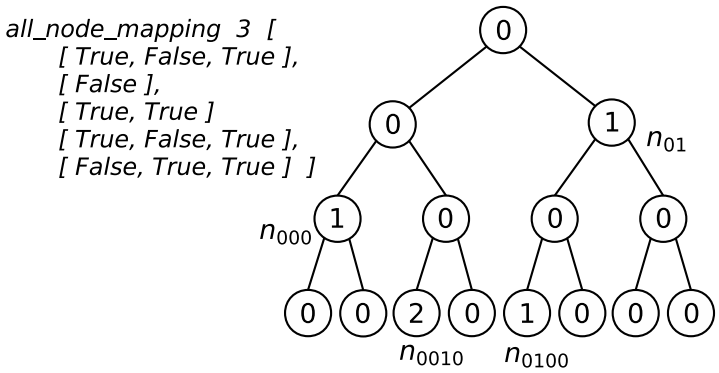

図 2 *all_node_mapping* の図示

に *payment_amount* と一致した．

lemma *payment_amount_eq_all_node_amount:*

```
"max_list (map length bss) ≤ l
 ⟹ payment_amount l bss = all_node_amount (all_node_mapping l bss)"
```

この二分木に対し，Route node rule あるいは Same node rule に違反しているかどうかを検査する関数 *check_doublespending* を定義することで，補題 *case_of_exists_sublist* を以下のように二分木の性質として記述し直し，二分木の帰納法による証明を与えた．

lemma *overspending_imp_doublespending:*

```
"all_node_amount t > 2 ^ depth t
 ⟹ check_doublespending t"
```

depth は木の深さを求める関数である．つまり，元の金額よりも使用金額が上回っている場合，Route node rule あるいは Same node rule に違反していることを示している．

また，補題 *case_of_exists_sublist* と補題 *overspending_imp_doublespending* を繋ぐ補題として，補題 *payment_amount_eq_all_node_amount* の他に次の補題が必要である．

lemma *doublespending_imp_exists_sublist:*

```
"⟦ max_list (map length bss) ≤ l;
   check_doublespending (all_node_mapping l bss) ⟧
 ⟹ ex_2elem_list check_sublist bss"
```

この補題は，リスト *bss* に存在するノード全てについて二分木上にマッピングしたとき，Route node rule あるいは Same node rule に違反していれば，*bss* 内に *check_sublist* を満たす 2 つのノードが存在することを示す．これにより，補題 *case_of_exists_sublist* が証明され，これを用いて定理 *overspending* を証明する．

5. 議　　論

理想的電子現金方式の安全性の検証において，2 つの問題点が浮上した．ひとつは，二分木上で *check_doublespending* が満たされたとき，リスト上に *check_sublist* を満たすような 2 つの要素が存在することを表す補題 *doublespending_imp_exists_sublist*．もうひとつは，定理 *overspending* の証明の末に残る *X1* ≠ *X2* というサブゴールである．

check_doublespending と *check_sublist* との関係を表す補題の証明が困難なのは，ノードのリスト *bss* が時系列に依存していることが原因だと思われる．イベント列 *evs* からノードの位置情報リスト *bss* を生成する際，*bss* の要素の順序はイベント列の順序，すなわち時系列に依存する．一方，二分木は根ノードを中心とする再帰構造となっており，時系列とは全く異なる．このように全く異なる再帰構造を持つデータ間に帰納法を適用するのは無理である．我々は，前研究では二分木と自然数という異なる再帰構造を持つデータ間の性質に対して，適切な中間データを用意することにより，帰納法を適用することに成功した[16]．しかし，本研究ではそのような中間データを発見できなかった．

たとえば，図 2 の二分木に対し，帰納法を適用することにより，リストとの対応関係を論じようと試みる．まず，根ノード n_0 の属性値は 0 であり，ノードは使用されていないため，リストとの対応関係はないと言える．続いて，左右の部分木について同様に属性値を参照する．ここで，右の部分木に着目すると，根ノード n_{01} の属性値は 1 であるため，リストに該当要素 *[False]* が存在するはずである．しかし，帰納法において，これがリストの先頭要素でなければ失敗する．ここでは *[False]* はリストの 2 番目の要素であるため，帰納法では判定できない．

一般に，二分木上の操作列と，その結果得られる二分木との対応関係を証明する場合にも，同様の問題が生じる．この証明では，ある性質を常に満たす二分木上でノードの挿入や削除を定義したとき，その操作がどのような順序で行われても，性質は保たれていることを証明することになる．この証明も，上記の時系列の問題のため，困難であることが予想される．

一方，*X1* ≠ *X2* というサブゴールについては，メッセージを区別できないことが問題となる．定理 *overspending* において，前提部にも結論部にもメッセージ *X1* および *X2* は

出現しておらず，証明の過程において，前提部に ∃*X1*, ∃*X2* という形で登場する．

実際には，定理 *overspending* の証明の過程で，次のようなサブゴールが提示される．

```
⋀ bs1 bs2.
  ⟦ evs ∈ iecp;
                              （中略）
    bs1 ∈ set (payment_node CoinInfo evs);
    bs2 ∈ set (payment_node CoinInfo evs);
    check_sublist bs1 bs2 ⟧
   ⟹ ∃ev1∈set evs. ∃ev2∈set evs. check_overspending CoinInfo ev1 ev2
```

これに対し，次のような補題を *bs1*, *bs2* についてそれぞれ適用する．

```
lemma pick_NPos_in_bss:
  "bs ∈ set (payment_node CoinInfo evs)
   ⟹ ∃X ∈ knows Bank evs. pick_NPos CoinInfo X = (bs, True)"
```

その結果生成されるサブゴールは以下のとおり．

```
⋀ bs1 bs2.
  ⟦ evs ∈ iecp;
                              （中略）
    bs1 ∈ set (payment_node CoinInfo evs);
    bs2 ∈ set (payment_node CoinInfo evs);
    check_sublist bs1 bs2;
    ∃X1∈knows Bank evs. pick_NPos CoinInfo X1 = (bs1, True);
    ∃X2∈knows Bank evs. pick_NPos CoinInfo X2 = (bs2, True) ⟧
   ⟹ ∃ev1∈set evs. ∃ev2∈set evs. check_overspending CoinInfo ev1 ev2
```

この *X1* と *X2* が区別されるべきデータであるが，これらは独立して出現するため，同一のメッセージであっても許されてしまう．この問題は，同一のリストや集合から 2 個以上の要素を取り出す際に起こるものであり，帰納的な証明方法はうまく適用できない．

一般に，帰納的アプローチにおいて，メッセージ同士の比較をする必要はない．なぜなら，モデルに量的なデータを含まないため，そのメッセージが存在するかどうかさえ分かれば良いからである．しかし，本研究で扱った理想的電子現金プロトコルの場合，安全性を論じる上で，ノード情報と金額といった量的なデータをモデルに含むことは不可欠であった．そのような量的なデータを比較する際に，比較対象のデータを含む複数のメッセージが同一のものであっては意味がないため，まずメッセージ自体を区別する必要がある．この「区別」という概念を解決しない限り，量的なデータを考慮するプロトコルについて，帰納的アプローチを適用することは難しいだろう．

6. おわりに

本研究では，証明支援系 Isabelle/HOL を用いることにより，理想的電子現金方式についてプロトコルレベルにおけるモデル化を行い，安全性についての検証を試みた．第 5 節で述べた 2 つの問題点は未解決となったが，この問題を解析することにより，Isabelle/HOL におけるプロトコルの検証に広く適用されている帰納的アプローチについて，量的なデータを扱う際の問題点を明確化した．なお，今回の証明のために，およそ 50 個の補題を用意した．

今後の展望としては，量的なデータを扱うプロトコルのモデル化のアプローチについて模索するつもりである．また，未だ手つかずであるプライバシーの検証や，通信を傍受するスパイをエージェントとして追加し，スパイによる電子現金の偽造や複製など，スパイを絡めた安全性の検証についても検討するつもりである．

参考文献

1) G.Bella, F.Massacci, and L.C. Paulson. Verifying the SET purchase protocols. In *Journal of Automated Reasoning*, Vol.36, pp. 5–37, 2006.
2) G. Bella and L. C. Paulson. Kerberos version IV: inductive analysis of the secrecy goals. In J.J. Quisquater, editor, *5th European Symposium on Research in Computer Security*, Vol. 1485 of *Lecture Notes in Computer Science*, pp. 361–375, 1998.
3) B.Berard, M.Bidoit, A.Finkel, F.Laroussinie, A.Petit, L.Petrucci, P.Schnoebelen, and P.McKenzie. *Systems and Software Verification: Model-Checking Techniques and Tools*. Springer, 2001.
4) E.M. Clarke, O.Grumberg, and D.Peled. *Model Checking*. The MIT Press, 2000.
5) C. Cornes, J. Courant, J. C. Filliatre, G. Huet, C. Murthy, C. Munoz, C. Parent, C.Symbolique, P.Manoury, P.Manoury, A.Saibi, B.Werner, and Projet Coq. *The Coq Proof Assistant - Reference Manual*, 1995.
6) J.Crow, S.Owre, J.Rushby, N.Shankar, and M.Srivas. *A Tutorial Introduction to PVS*, 1995.
7) M.Kaufmann and J.S. Moore. *ACL2-Tutorial*, 1997.
8) Y.Minamide. Verified decision procedures on context-free grammars. In *Proc. of the 20th International Conference on Theorem Proving in Higher Order Logics*, Vol.

4732 of *Lecture Notes in Computer Science*, pp. 173–188, 2007.
9) T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL A Proof Assistant for Higher-Order Logic*. Springer, 2008.
10) T. Okamoto. An efficient divisible electronic cash scheme. In *Proc. of Crypto'95*, pp. 438–451, 1995.
11) T. Okamoto and K. Ohta. Universal electronic cash. In *Proc. of Crypto'91*, pp. 324–337, 1992.
12) T. Okamoto and K. Ohta. A universal electronic cash scheme. *The transactions of the Institute of Electronics, Information and Communication Engineers*, Vol. J76-D-1, No.6, pp. 315–323, 1993.
13) L.C. Paulson. The inductive approach to verifying cryptographic protocols. *Computer Security*, pp. 85–128, 1998.
14) L.C. Paulson. Inductive analysis of the internet protocol TLS. *ACM Transactions on Computer and System Security*, pp. 332–351, 1999.
15) L.C. Paulson and K.W. Susanto. Source-level proof reconstruction for interactive theorem proving. In *Proc. of the 20th International Conference on Theorem Proving in Higher Order Logics*, Vol. 4732 of *Lecture Notes in Computer Science*, pp. 232–245, 2007.
16) K. Takahashi and S. Yoshimaru. Formalization of data conversion for inductive proof. In *Tunisia-Japan Workshop on Symbolic Computation in Software Science*, pp. 135–150, 2009.