(19)日本国特許庁(JP)　(12)公表特許公報(A)　(11)特許出願公表番号
特表2020-532002
(P2020-532002A)
(43)公表日　令和2年11月5日(2020.11.5)

(51)Int.Cl.	F I	テーマコード（参考）
G06F 21/64 (2013.01)	G06F 21/64	

審査請求 未請求　予備審査請求 未請求　（全 76 頁）

(21)出願番号　特願2020-511284 (P2020-511284)
(86)(22)出願日　平成30年8月24日 (2018.8.24)
(85)翻訳文提出日　令和2年2月21日 (2020.2.21)
(86)国際出願番号　PCT/IB2018/056431
(87)国際公開番号　WO2019/043537
(87)国際公開日　平成31年3月7日 (2019.3.7)
(31)優先権主張番号　1713805.8
(32)優先日　平成29年8月29日 (2017.8.29)
(33)優先権主張国・地域又は機関
英国 (GB)
(31)優先権主張番号　1713794.4
(32)優先日　平成29年8月29日 (2017.8.29)
(33)優先権主張国・地域又は機関
英国 (GB)

(71)出願人　318001991
エヌチェーン　ホールディングス　リミテッド
NCHAIN HOLDINGS LIMITED
アンティグア・バーブーダ、　セントジョンズ、　４４　チャーチ　ストリート、　フィッツジェラルド　ハウス
Fitzgerald House, 44 Church Street, St. John's, Antigua and Barbuda (AG)
(74)代理人　100107766
弁理士　伊東　忠重

最終頁に続く

(54)【発明の名称】ブロックチェーンにおけるアンロックトランザクションのインプットに対する制約

(57)【要約】
トラストレス決定性状態機械がブロックチェーンインフラストラクチャを用いて実装でき、状態機械は、1つより多くのブロックチェーントランザクションに渡り同時に実行できる。トランザクションは、Bitcoinブロックチェーン台帳の中で実行可能である。前のトランザクションのアウトプットを参照するトランザクションインプットを含むよう、アンロックトランザクションを制約するアンロックトランザクション制約が決定される。償還可能トランザクションは、量とトランザクションロックスクリプトとを含むトランザクションアウトプットを含むよう生成される。該トランザクションロックスクリプトは、アンロックトランザクション制約を含み、該量をアンロックすることは、アンロックトランザクション制約を満たすアンロックトランザクションの少なくとも1つのアンロックスクリプトの実行を条件とする。償還可能トランザクションは、ブロックチェーンネットワークのノードにおいて検証されるようにされる。

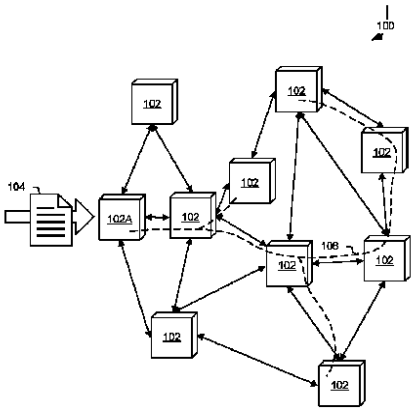


FIG. 1

【特許請求の範囲】
【請求項１】
コンピュータにより実施される方法であって、
前のトランザクションのアウトプットを参照するアンロックトランザクションインプットを含むようアンロックトランザクションを制約するアンロックトランザクション制約を決定するステップと、
償還可能トランザクションを生成するステップであって、該償還可能トランザクションは、
償還可能量を含むトランザクションアウトプットと、
前記アンロックトランザクション制約を含むトランザクションロックスクリプトと、
を含み、前記償還可能量のアンロックは、前記アンロックトランザクション制約を満たす前記アンロックトランザクションの少なくとも1つのアンロックスクリプトの実行を条件とする、ステップと、
ブロックチェーンネットワークのノードにおいて前記償還可能トランザクションを検証させるステップと、
を含むコンピュータにより実施される方法。
【請求項２】
前記アンロックトランザクション制約は、さらに、特定のハッシュ値を含むよう前記アンロックトランザクションを制約する、請求項1に記載のコンピュータにより実施される方法。
【請求項３】
前記特定のハッシュ値は、前記前のトランザクションのアウトプットを参照する識別子を符号化する、請求項2に記載のコンピュータにより実施される方法。
【請求項４】
前記アンロックトランザクション制約は、さらに、前記トランザクションロックスクリプトから複製されたスクリプト要素セットを含むよう前記アンロックトランザクションのロックスクリプトを制約する、請求項1乃至3のいずれか一項に記載のコンピュータにより実施される方法。
【請求項５】
前記償還可能トランザクションは、複数の状態を有するコントラクトを符号化する、請求項1乃至4のいずれか一項に記載のコンピュータにより実施される方法。
【請求項６】
前記アンロックトランザクションのアウトプットの償還可能価値を決定するステップ、を更に含む請求項1乃至5のいずれか一項に記載のコンピュータにより実施される方法。
【請求項７】
前記アンロックトランザクション制約は、前記前のトランザクションのアウトプットからの特定のロックスクリプト要素を含み、
前記アンロックトランザクションインプットが前記特定のロックスクリプト要素を含む結果として、前記少なくとも1つのアンロックスクリプトの実行が、前記アンロックトランザクション制約を満たす、請求項1乃至6のいずれか一項に記載のコンピュータにより実施される方法。
【請求項８】
前記特定のロックスクリプト要素は、特定のエンティティの暗号鍵を符号化する、請求項7に記載のコンピュータにより実施される方法。
【請求項９】
前記アンロックトランザクション制約は、第1アンロックトランザクション制約であり、
前記方法は、さらに、
前記アンロックトランザクションを更に制約するために、第2アンロックトランザクション制約を決定するステップと、

第2償還可能トランザクションを生成するステップであって、該第2償還可能トランザクションは、
第2量と、
前記第2アンロックトランザクション制約を含む第2ロックスクリプトと、を含み、第2量をアンロックすることは、さらに、前記第2アンロックトランザクション制約を満たす前記少なくとも1つのアンロックスクリプトの実行を条件とする、ステップと、
を含む請求項1乃至8のいずれか一項に記載のコンピュータにより実施される方法。
【請求項１０】
前記ロックスクリプトは、第1ロックスクリプトであり、
前記少なくとも1つのアンロックスクリプトは、第1アンロックスクリプト及び第2アンロックスクリプトを含み、
前記第1アンロックトランザクション制約は、前記第1ロックスクリプトの少なくとも一部を含むよう前記第1アンロックスクリプトを制約し、
前記第2アンロックトランザクション制約は、前記第2ロックスクリプトの少なくとも一部を含むよう前記第2アンロックスクリプトを制約する、
請求項9に記載のコンピュータにより実施される方法。
【請求項１１】
前記第1ロックスクリプトの前記少なくとも一部は、第1エンティティに関連付けられた暗号鍵を含み、
前記第2ロックスクリプトの前記少なくとも一部は、前記第1エンティティと異なる第2エンティティに関連付けられた暗号鍵を含む、
請求項10に記載のコンピュータにより実施される方法。
【請求項１２】
前記アンロックトランザクション制約は、さらに、アンロックトランザクションのアウトプットを制約するよう前記アンロックトランザクションを制約する、請求項1乃至11のいずれか一項に記載のコンピュータにより実施される方法。
【請求項１３】
前記アンロックトランザクション制約は、前記コントラクトと異なる別のコントラクトを符号化し、
前記償還可能量をアンロックすることは、前記他のコントラクトが前記アンロックトランザクションの前記アウトプットの中で実施されることを条件とする、請求項12に記載のコンピュータにより実施される方法。
【請求項１４】
システムであって、
プロセッサと、
前記プロセッサによる実行の結果として前記システムに請求項1乃至13のいずれか一項に記載のコンピュータにより実施される方法を実行させる実行可能命令を含むメモリと、
を含むシステム。
【請求項１５】
実行可能命令を格納した非一時的コンピュータ可読記憶媒体であって、前記実行可能命令は、コンピュータシステムのプロセッサにより実行された結果として、前記コンピュータシステムに請求項1乃至13のいずれか一項に記載のコンピュータにより実施される方法を少なくとも実行させる、非一時的コンピュータ可読記憶媒体。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、概して、コンピュータにより実施されるブロックチェーントランザクションを処理する方法に関し、より詳細には、トラストレスな（trustless）、決定性な、及び並行状態機械を含む、ブロックチェーントランザクション処理の構造内で状態機械を実装することに関する。本発明は、さらに、ブロックチェーンネットワーク上で行われる電子

的な移転に関してセキュリティを実施するために、暗号技術及び数学技術を利用する。本発明は、スマートコントラクトトランザクションを処理及び行い、かかるスマートコントラクトトランザクションを使用して状態機械を実行する方法及び装置における使用に特に適しているが、これらに限定されない。

【背景技術】

【０００２】

本明細書において、「ブロックチェーン」という用語は、コンセンサス方式のブロックチェーン及びトランザクションチェーン技術、認可・未認可台帳、共有台帳、及びそれらの変形を含む、いくつかの種類の電子的、コンピュータベースの分散台帳のいずれかを表し得る。ブロックチェーン技術の最も広く知られている用途はBitcoin台帳であるが、他のブロックチェーン実装が提案され、開発されている。便宜上及び説明の目的で、本願明細書に記載される有用な用途として「Bitcoin」の例が参照され得るが、Bitcoinは本開示に記載の技術が適用され得る多くの用途のうちのほんの1つである。しかしながら、本発明は、Bitcoinブロックチェーンとの使用に限定されず、非商業的用途を含む代替のブロックチェーン実装及びプロトコルもまた、本発明の範囲内に含まれることに留意されたい。例えば、本開示に記載される技術は、暗号通貨の交換が起こるかどうかにかかわらず、トランザクション内で符号化され得る制約に関してBitcoinと同様の制限を有するブロックチェーン実装を利用する利点を提供する。

【０００３】

ブロックチェーンとは、ピアツーピアの電子台帳であり、ブロックで構成されたコンピュータベースの非集中型の分散システムとして実装され、一方、ブロックはトランザクションやその他の情報で構成される。いくつかの例において、「ブロックチェーントランザクション」は、データ及び条件セットを含むフィールド値の構造化された集合を符号化するインプットメッセージを指し、この場合、条件セットの充足は、ブロックチェーンデータ構造に書き込まれるフィールドセットの前提条件である。ブロックチェーンシステム又はブロックチェーンネットワークは、複数のブロックチェーンノード及び動作セットを含んでもよい。ブロックチェーンノードは、動作セットの一部又はすべてを実行するように設定することができる。種々のブロックチェーンノードは、ノードオペレータによって操作されるコンピュータハードウェア、コンピュータソフトウェア、又はその両方の組み合わせとして実装されてもよく、ノードオペレータは、独立していてもよく、他のノードオペレータとは無関係であってもよい。ブロックチェーンノードは、それぞれ、ブロックチェーン台帳のコピー、又はその一部を保持することができる。動作セットには、トランザクションの作成、トランザクションの伝搬、ブロックチェーン台帳の読み取り、ブロックチェーン台帳の評価、ブロックチェーン台帳への提案された追加のための新しいブロックの生成（マイニング）、他のブロックチェーンノードとの通信、及びユーザーがブロックチェーン資産を管理するためのウォレット機能の提供が含まれる。

【０００４】

ブロックチェーン台帳は、台帳がいつ修正されるかを決定する単一のブロックチェーンノード又はエンティティが存在しないため、非集中型であってもよい。その代わりに、ブロックチェーンノードは、それぞれブロックチェーンプロトコルのルールの知識を用いてプログラムされ、ブロックチェーン台帳を検証し、他のブロックチェーンノードの動作がそれらのルールと一致することを検証することができる。「ブロックチェーン」という用語は、ブロックチェーン台帳が、各々がコンピュータメモリ内のデータ構造として表すことができ、コンピュータプロセスによって読み取り可能であり、データ送信として送信可能である一連の連鎖した（chained）ブロックを含むという事実を指すことができる。ブロックは、1つ以上のトランザクションを含み、コンピュータメモリ内のデータ構造として表現することができ、コンピュータプロセスによって読取ることができ、データ伝送として送信することができる。ブロックは、そのブロックチェーンに公式に追加される各新しいブロックが、その直前のブロックに対する不変の参照を含み、該直前のブロックがその直前のブロックに対する不変の参照を含み得る、などのように、連鎖され得る。

【０００５】
ブロックチェーンプロトコルのルールの1つは、ブロックがブロックチェーンに追加されると、それを変更することができない、すなわち、それは不変であり、ブロックチェーン台帳の唯一の変更可能なものは、新しいブロックの追加である、ということである。ブロックチェーンノードは、ルールとして、そのようにプログラムされ得るので、ブロックチェーンノードは、ブロックチェーン台帳の該ブロックチェーンノードのコピー中のブロックを修正できず、ブロックを追加するだけであり、さらに、ブロックチェーンプロトコルに準拠することを保証するために、提案したブロックに対して検証プロセスを実行した後にのみブロックを追加し得る。このようなブロックは、一旦、台帳に加えられると、不変であるため、ブロック内のトランザクションもまた、不変であり得る。

【０００６】
ブロックチェーンノードは、トランザクションを作成するときに、トランザクションの詳細を含むデータオブジェクトを作成し、典型的にはピアツーピア方式で、ブロックチェーンノードが接続できる他のブロックチェーンノードにデータオブジェクトを伝搬してよい。いくつかのブロックチェーンノードは、「マイナー」として動作し、1つ以上のトランザクションを収集し、ブロックに対応するデータ構造を形成し、ブロックに入れられたトランザクションを検証するためにいくつかの計算を行い、パズルを解き、パズルの解をデータ構造に入れ、ブロックを他のブロックチェーンノードに伝えようとする。このパズルは、トランザクションのデータ及びブロックチェーン台帳の現在状態に特有の、例えば、台帳内のブロック数及び最後に追加されたブロックからのデータのような、自明でない計算の形式であってよい。

【０００７】
パズルをトランザクションに依存させることによって、不正なブロックチェーンノードは、事前に作成されたブロックを伝搬することができなくなり得る。自明でないパズルを解くことによって、不正なブロックチェーンノードはブロックチェーンネットワークにブロックを単純に注入することができないかも知れないが、ブロックチェーンノードが努力を提供したことを証明するために重要な計算タスクを行うことを要求するかも知れない（事実上、難しい問題に対する解を示すことは「proof－of－work」である）。好ましくは、ブロックチェーンプロトコルでは、proof－of－workは容易ではないが、パズルが解決され、作業が行われたことを検証することは容易である。つまり、他のブロックチェーンノードは、必ずしも新しいブロックを台帳に追加することを提案しているブロックチェーンノードを信頼する必要はない。他のブロックチェーンノードによって検証された場合、そのブロックチェーンノードは、自身のブロックチェーン台帳のコピーの最後に新しいブロックを追加し、他のブロックチェーンノードに該新しいブロックを伝搬することができる。他のブロックチェーンノードも同様の検証を行うため、ブロックが有効であり、ブロックチェーン台帳に追加されるべきであると結論付けることもできるので、ブロックチェーンノードのブロックチェーン台帳のコピーに新しいブロックを追加することができる。ブロックチェーンノードは、提案された新しいブロックが有効でないと判断した場合、自身のブロックチェーン台帳のコピーに該新しいブロックを追加しなくてよく、伝搬しなくてよい。ブロックチェーンの妥当性はコンセンサスに基づいているので、トランザクションが妥当であることに大多数のノードが同意した場合、トランザクションは妥当であるとみなされてよい。

【０００８】
ブロックチェーンシステムは、ノード所有者及びオペレータが必ずしも他のノード所有者及びオペレータを信頼しないように動作してもよい。代わりに、プロトコルは、以前のブロックを修正する、対応するproof－of－work無しに新しいブロックを提案する、又は無効なトランザクションを含めるなど、プロトコルによって許可されていない特定の操作を実行することを計算上不可能にしてよい。特定の信頼は、他のノードによって気づかれないであろう方法でブロックチェーンプロトコルによって許可されない操作を実行することが計算上実行不可能である場合、必要とされなくてよい。

【０００９】
ブロックチェーンに書き込まれるブロックにトランザクションを含めるために、（1）マイニングブロックチェーンノードはトランザクションを検証し、（2）マイニングブロックチェーンノードはそのトランザクションを含むブロックを伝搬させようとし、（3）他のノードはブロックの妥当性と該ブロック内のトランザクションの妥当性を検証する。マイニングブロックチェーンノードは、これらのルールを念頭に置いてプログラム又は設定されるため、マイニングブロックチェーンノードが、検証に失敗したトランザクションにブロックに含める可能性は低く、このようなブロックは他のノードにより受け入れられず、マイニングノードは利益を得ることができない。いくつかのブロックチェーンシステムに対するマイニングのそのような利点の一つは、ブロックが受け入れられる場合、ブロックが「分配」トランザクションを含むことが許され、特定の量の価値が、何らかの他のエンティティからの対応する値の減少を必要とせずに、そのノードのオペレータに割り当てられることである。このようにして、成功したマイナーは、ブロックと共に生成された価値によって報酬を受けることができる。さらに、以下に説明するように、トランザクションにはトランザクション手数料が含まれる。トランザクション手数料は、マイニングブロックチェーンノードを制御するマイナーにも届くものであり、また、当該マイナーが補償を受けるためには、トランザクションが有効であることを確認することが必要である。
【００１０】
ブロックチェーンネットワークにおけるトランザクションは、トランザクション値、トランザクション時間、及び／又は他のデータ要素などの様々なデータ要素を含む。非集中型の分散台帳システムでは、台帳は公開されているため、誰もが台帳を閲覧し、トランザクションを見ることができる。ブロックチェーン台帳では、ブロックチェーンを開始し、そのジェネシス（genesis）トランザクションに何らかの価値の単位を割り当てるジェネシストランザクションが存在し得る。本明細書の例において、説明目的のために、価値の単位は暗号通貨であるが、他のバリエーションも可能である。
【００１１】
ジェネシストランザクション及び分配トランザクション以外のトランザクションは、ブロックチェーン台帳上の1つ以上の既存トランザクションの「アンロック」を伴い、このようなトランザクションがブロックチェーン台帳に追加されるとき、該トランザクションを移転することができる。各々の未移転トランザクションは、そのトランザクションの価値をアンロックするために必要な要件を公的に規定している。単純な要件は、「あなたはまず、あなたがAliceであることを証明しなければならない。次に、それをアンロックすることができる」ことである。Aliceは、次に、そのトランザクションの価値を「アンロック」する新たなトランザクションを生成することができる。ここで、Aliceの新たなトランザクションは、それがAliceから来たことを証明し、かつ、その前のトランザクションへのポインタを有する。分配トランザクションについては、トランザクション価値はあるが、分配トランザクションは前のトランザクションを「アンロック」しない。もちろん、Aliceのトランザクションがブロックチェーンネットワークに受け入れられるためには、すでに移転されたトランザクションを参照することはできず、Aliceがそれを作成したことを実際に証明しなければならない。
【００１２】
ブロックチェーンプロトコルによって、したがってブロックチェーンノードの合意によって、トランザクションは、すでに移転された前のトランザクションのアウトプットを指す場合、すなわち、台帳が前のトランザクションのアウトプットを指す有効な既存のトランザクションインプットを含む場合、有効ではない。前のUTXOで表される値を「アンロック」する新しいトランザクションインプットを侵入者（interloper）が生成するのを防ぐために、各トランザクションアウトプットには、そのようなトランザクションを生成する請求者に課された要件を表すデータが含まれる。UTXOは不変であるため、そのデータは変更できない。もちろん、移転されたトランザクションもまた不変であり得る。
【００１３】

上記の例では、Aliceはアンロックトランザクションを作成したかも知れない。その結果、自分だけがアンロックする能力を持っていた前のトランザクションの中の価値をBobに移転することができるようになる。つまり、今度はBobだけがアンロックする能力を持っている、それによりアンロックする、新しい未移転トランザクションが存在するようになる。Aliceが作成したアンロックトランザクションには、「誰でも自由にこのトランザクションを指し示すことができ、Bobの秘密鍵を知っていることを証明するのに十分な情報を提供できれば、その価値のすべてをアンロックすることができる」という要件に対応するデータが含まれてよい。Bobが慎重であると仮定すると、Bobは、そのトランザクションをアンロックする有効なトランザクションを作成できる唯一の人物となる。実際、Bobはその価値を所有しており、その価値をアンロックできるのはBobだけである。これは、Bobがブロックチェーンノードのオペレータを信頼したり、トランザクションを作成する能力を持つ他の相手を信頼したりする必要はないことに留意する。Bobが信頼する必要があるのは、不正な相手がブロックチェーンノードの大部分を完全に支配することができないということである。

【００１４】

特定の場合には、1つのトランザクションは、正確に1つの以前の未移転トランザクションを完全にアンロックし、1つのトランザクションは、後のトランザクションによって完全に移転されるか、又は全く移転されない。一般的なケースでは、トランザクションは、1つ以上のインプットと1つ以上のアウトプットを有し、各インプットは、前のトランザクションのアウトプットを参照し（及び、そのアウトプットは、償還可能な（redeemable）価値を有する）、各アウトプットは、償還可能な価値（将来のトランザクションのインプットによって移転/参照されるまで、移転されないままである）を有する。トランザクションのアウトプットは、そのトランザクションの償還可能単位であり、そのすべてが移転されるか全く移転されない。なお、本明細書に記載されている例では、トランザクションが「移転されている」という場合があり、トランザクションが複数のトランザクションのアウトプットを有する場合には、そのトランザクションのすべてのアウトプットよりも少ない数のアウトプットを移転させる場合を包含し、トランザクションアウトプットをアンロックすることとは対照的に、「トランザクションをアンロックする」という場合は、1つのアウトプットのみを有するトランザクションのアウトプットが移転されている場合を包含し得る。

【００１５】

トランザクションが複数のアウトプットを持つことができる場合、そのトランザクションの異なるトランザクションアウトプットは、異なる時間に移転することができる。トランザクションが作成されると、そのアウトプットは移転されていないと考えられてよい。各アウトプットは、後のトランザクションのインプットによって移転されるか、又は移転されないままである。トランザクションが、前のトランザクションのアウトプットがすでに移転された後に、前のトランザクションのアウトプットをアンロックしようとするインプットを持っている場合、ブロックチェーンノードは、それを無効なトランザクションとして拒否する。

【００１６】

一方のパーティAliceがXの値を持つUTXOを制御し、そのトランザクションアウトプットの一部分Yのみをアンロックしたい場合、Aliceは複数のアウトプットを持つ新しいトランザクション、1つはBobによってのみ移転可能な価値Yのトランザクションアウトプット、1つはAliceによってのみ移転可能な価値X－Yのトランザクションアウトプットを指定できる。事実上、元のトランザクションアウトプットは完全に移転されるが、Aliceのトランザクションに「変更を加える」新しいトランザクションアウトプットがある。

【００１７】

トランザクションへのインプットの数とそのトランザクションのアウトプットの数は、同じである必要はない。しかし、有効なトランザクションであるためには、現在のトランザクションのアウトプットで指定された価値の合計は、現在のトランザクションのインプ

ットによって移転され得る前のトランザクションのアウトプットの価値の合計を超えてはならず、いくつかの例外を除いて、場合によっては、より小さくなる。ジェネシストランザクションと分配トランザクションの場合、アウトプットの価値の合計は、インプットの価値の合計よりも大きくすることができる。又は、インプットがまったく必要ないが、通常のトランザクションの場合、そのアウトプットの価値の合計がそのインプットの価値の合計を超えると、トランザクションは無効になる。一部のトランザクションでは、アウトプットの価値の合計がインプットの価値の合計よりも小さくなる場合がある。この場合、成功したマイナーは、差額のトランザクション手数料トランザクションを加算し、次に、マイナーは精算し、トランザクションを処理する理由を生み出す。Bitcoinプロトコルでは、これはCoinbaseトランザクションであり、ブロックに追加され、Bitcoinブロックに含まれる他のすべてのトランザクションのトランザクション手数料の合計（すなわち、含まれるすべてのトランザクションのすべてのインプットとアウトプットの差の合計）に、新しいブロックを作成するための配分（distribution）を加えたものに等しい償還可能なアウトプットを持つ。

【００１８】

トランザクションの各アウトプットには、そのアウトプットの価値をアンロックするために満たさなければならない制約が含まれる。いくつかのブロックチェーンプロトコルでは、これらの制約が何であるかを定義するデータとスクリプトコマンドを指定する「ロックスクリプト」に制約が埋め込まれている。ロックスクリプトは、トランザクションアウトプット内で示される価値に対する担保として機能する。この場合、他のアクターは、トランザクションアウトプットのロックスクリプトを「アンロック」できない限り、トランザクションアウトプット内で示される価値を「アンロック」できない。

【００１９】

アンロックトランザクションへの各インプットは、前のトランザクションのアウトプットをアンロックする。アンロックトランザクションインプットの「アンロックスクリプト」は、アンロックトランザクションが前のトランザクションのアウトプットをアンロックするかどうかを決定する。したがって、有効なトランザクションは少なくとも1つのインプットを指定し、有効なアンロックトランザクションの各インプットは、前のトランザクションのアウトプット（移転されるアウトプット）へのポインタと、ロックスクリプトを「アンロック」するアンロックスクリプトを含む。対応するブロックチェーンプロトコルに従って動作するブロックチェーンノードは、トランザクションインプットを検証するために、ロックスクリプトとアンロックスクリプトを一緒に実行してよい。特定のシステムでは、スクリプトはスタックベースであり、検証ブロックチェーンノードは空のスタックで開始し、スタック上にデータオブジェクトを残し得るアンロックスクリプトを実行し、次にスタック上のデータオブジェクトを使用し得るロックスクリプトを実行する。

【００２０】

検証ブロックチェーンノードがロックスクリプトとアンロックスクリプトを組み合わせ、その組み合わせを実行する場合、実行後の結果は「TRUE（真）」又は「FALSE（偽）」のいずれかになる。場合によっては、スクリプトが完全に実行される前に、スクリプトの実行がFALSEの結果で終了することがある。たとえば、特定のスクリプトの実行には、スクリプトで何が起こるかに関係なく、有効なスクリプトの実行で常に等しい2つの値があると仮定する。そのスクリプトの実行の途中で、これらの2つの値の間で比較が行われ、それらが等しくない場合、スクリプトの実行はその比較の直後に停止し、FALSE結果を返すことができる。残りのスクリプトは実行する必要はない。

【００２１】

検証ブロックチェーンノードがロックスクリプトとアンロックスクリプトを組み合わせ、その組み合わせを実行し、その結果がTRUE（すなわち、アンロックスクリプトにはトランザクションアウトプットのアンロックに必要なすべてのものが含まれている）である場合、検証ブロックチェーンノードはトランザクションが有効であることを検証する（適切なタイムスタンプ、適切なフォーマット、移転済みのトランザクションアウトプットを指

さないなど、他の要件が満たされていると仮定する）。検証ブロックチェーンノードがトランザクションが有効であることを検証する場合、該ノードは該トランザクションを伝搬してよい。他のブロックチェーンノードは、トランザクションが有効であると結論付けるために、同じ計算を行ってもよい。このようにして、UTXOのみを指すインプットを持ち、それらのUTXOをアンロックするアンロックスクリプトを持つ有効なトランザクションが伝搬し、結局は、最終的に台帳の一部となるブロックの一部となり得る。

【００２２】

一方、不正なノードが不正なトランザクションを伝搬しようとすると、他のノードはそれが不正であると判断し、それを伝搬しなくてよい。

【００２３】

トランザクションが有効であり、ブロックチェーンに受け入れられると、その内容は変更できない。つまり、ロックスクリプトはトランザクションの作成時に確定される。しかし、アンロックスクリプトは、後のアンロックトランザクション内に含まれており、後のアンロックトランザクションが作成されるまで作成される必要はないため、必ずしもそのときに確定されるとは限らない。

【００２４】

典型的な場合、検証するアンロックスクリプトは、誰かによって作成されるのではなく、前のトランザクションのアウトプットをアンロックする権限を与えられたパーティによってのみ作成される。上記の例のように、ロックスクリプトは、「Bobの秘密鍵を知っていることを証明するのに十分な情報を提供できる場合、誰でも自由にこのトランザクションアウトプットを指し示すことができ、それによって、すべての規定値をアンロックすることができる。」であり、アンロックスクリプトは、「Bobは自分の秘密鍵によりトランザクションに署名し、ここで結果はABCCCである。」という形式であるかも知れない。その後、検証ブロックチェーンノードは、これらの2つのステートメントを処理し、TRUE又はFALSEの結論に達してよい。このプロセスは、ABCCCが有効な署名であることを検証することが容易であり、Bob（又はBobの秘密鍵を知っている他の誰か）がそのような署名を生成することが容易であり、Bobの秘密鍵を知らずに他の誰かが有効な署名を生成することが非常に困難である場合に、うまく機能する。その結果、トラストレスなシステムで価値を移転することができる。Bobがロックをアンロックしようとしているトランザクションの発信者は、Bobの秘密鍵を最初に知らなくてもブロックチェーンノードのコンセンサスによって受け入れられるであろう有効で検証可能な結果を形成することは暗号学的に困難であるため、システムを信頼する、又はBobを信頼する必要がない。

【００２５】

ブロックチェーンノードは、Bobがトランザクションに署名したことを簡単に検証し、署名がロックスクリプトの唯一の要件であることを検証できる。もちろん、他のノードが検証する場所を検証せず、他のノードが検証しない場所を検証し得る不正なノードが存在する可能性があるが、不正なノードがブロックチェーンネットワーク上の善良なノードを圧倒できない限り、不正なノードは無効なトランザクションをプッシュしたり、有効なトランザクションが伝搬又はマイニングされるのを止めることができない。

【００２６】

ノードがトランザクションインプットのアンロックスクリプトと、前のトランザクションインプットの対応するロックスクリプトを実行し、それらのそれぞれをTRUEに評価し、他の検証条件（該当する場合）が満たされている場合、そのノードに関する限り、トランザクションは有効である。次に、そのノードは、検証されたトランザクションを他のネットワークノードに伝搬し、その結果、マイナーノードは、トランザクションをブロックに含めることを選択することができる。したがって、ブロックチェーンにトランザクションを書き込むためには、（1）トランザクションを受け取るノードによって検証され、（2）ネットワーク内の他のノードに中継され（トランザクションが検証された場合のみ）、（3）マイナーによって構築された新しいブロックに追加され、（4）提案されたブロックの一部として伝搬され、（5）ノードのコンセンサスにより、その提案されたブロックが過

去のトランザクションの公開台帳への追加として受け入れられなければならない。
【００２７】
トランザクションを実際上不可逆にするために、ブロックチェーンに十分な数のブロックが追加されたとき、トランザクションは確認されたとみなされてよい。トランザクションは不変であるため、ブロックチェーンプロトコルはトランザクションの一方的な逆戻りをブロックすることができる。もちろん、もしAliceが価値XをBobに移転し、Aliceがその価値Xの返却を要求するなら、Bobが同意すれば、Aliceは価値Xを得ることができる。この場合、Alice－to－Bob－for－Xトランザクションは逆戻り又はキャンセルされないが、Bobが開始する新しいトランザクション、Bob－to－Alice－for－Xトランザクションが存在する。
【００２８】
一旦トランザクションがブロックに含まれると、そのトランザクションは不変とみなされ、ブロックがブロックチェーンにコミットされると、そのブロックは不変とみなされる。ブロックチェーンの中に枝分かれ（fork）が存在し、ブロックチェーンをロールバックする何からの能力がある、短い期間があるかも知れないが、一般に、時間が長ければ長いほど、ロールバックは起こりにくくなる。ここで、特に断らない限り、トランザクションとブロックは、ブロックチェーンに完全にコミットされた後、不変であると仮定する。
【００２９】
不変性を保証する1つの方法は、暗号技術の使用によるものである。例えば、ハッシュやデジタル署名のような暗号操作があり、それらの入力として何らかのデータシーケンスを取り込み、該入力されたデータシーケンスに何らかの方法で対応する出力データシーケンスを提供する。操作は、所与の暗号入力（例えば、トランザクション又はブロック）から生成された所与の暗号出力（例えば、ハッシュ又はデジタル署名）に対して、利用可能な計算資源を使って、同じ暗号出力をもたらす異なる暗号入力を見つけることが、計算上実行不可能であるか又は不可能であるようなものであってよい。したがって、検証器は、暗号入力が暗号出力と整合する場合、暗号出力を生成するために使用されたのは暗号入力であって、他の修正された暗号入力ではないと想定することができる。
【００３０】
ブロックチェーンネットワークでは、各ノードが互いに信頼する必要がない場合、トランザクションはそのように検証され、ブロックはそのように検証されてよく、検証不能なトランザクションとブロックは無視され、未使用となってよく、トランザクションとブロックは実質的に不変であるとみなされてよく、これは、ハッシュ又はデジタル署名がトランザクション又はブロックに正しく対応する場合、トランザクション又はブロックがその元から変更されていないという仮定の結果であり得る。
【００３１】
ブロックチェーンノードの中には、台帳全体を格納するものもあれば、未使用のトランザクションアウトプット（例えばBitcoin台帳では、UTXO）だけを格納するものもある。UTXOは、償還可能な価値に対応し、各UTXOは、その価値の「所有者」以外の者が検証可能なアンロックスクリプトを容易に生成できないロックスクリプトを有することが好ましい。もちろん、これは必要条件ではありませんが、検証可能なアンロックスクリプトが誰でも簡単に生成できるようなUTXOは、最初にそれに気づいた人だけが償還できる別のUTXOにその価値を移転するトランザクションで、すぐに移転されることが予想される。結果として、ブロックチェーンは、暗号通貨価値の制御を移転するために使用することができる。より一般的には、移転されたトランザクションアウトプット及びUTXOを含む、ブロックチェーンシステムのある参加者から別の参加者へのデジタル資産及びトランザクション記録を、公開された不変の台帳に記録することができ、これにより、デジタル資産の流れを検証し、それらのデジタル資産の二重のアンロックを防止することが容易になる。
【００３２】
実施形態において、「デジタル資産」とは、使用権に関連するバイナリデータをいう。デジタル資産の例としては、Bitcoin、ether、及びLitecoinが挙げられる。本明細書にお

いて使用される場合、「デジタル資産」とは、1つ以上のデジタル資産を指し得る。例えば、トランザクションは、複数のインプットを有してもよく、これらのインプットの各々は、異なるデジタル資産を表すことができる。制御が移転されるデジタル資産は、この例では、複数のデジタル資産の集合であってもよく、集合自体がデジタル資産である。同様に、トランザクションは、例えば、インプットの数及びアウトプットの数が異なるように、1つ以上のアウトプットを生成するために、これらの複数のインプットを細分化及び／又は結合することができる。

【００３３】

一実施形態では、暗号通貨は、トークンベースの暗号通貨であり、各トークンは、資産のシェア（例えば、企業の株式）を表し、単一のトランザクションは、複数のタイプのトークン（例えば、1つ以上の異なる企業の株式）を含む。いくつかの実施形態では、例えば、Bitcoinのように、デジタル資産は非トークン化される。その結果、例えば、ブロックチェーン内で識別されるデジタル資産の識別子は存在しないが、むしろ、デジタル資産の制御は、ブロックチェーン上に記録されることになる有効なトランザクションを生成する能力を通じて実証される。しかしながら、いくつかのブロックチェーン実装は、例えば、デジタル資産がブロックチェーンに記録された情報を使用して特に識別可能であるように、トークン化されたデジタル資産を使用することができることに注意されたい。デジタル資産は、いくつかの実施形態においては、暗号通貨として使用され得るが、実施形態においては、デジタル資産は、他のコンテキストにおいて追加的又は代替的に使用可能であることが意図される。本発明は、デジタル資産の制御に適用可能であるが、本質的に技術的であり、デジタル資産の移転を必ずしも伴わずにブロックチェーンデータ構造を利用する他のコンテキストで使用することができることに留意されたい。

【００３４】

トランザクションには、ロックスクリプトとアンロックスクリプトが含まれており、これらは計算オブジェクトを形成することができる。いったんブロックチェーンにコミットされると、トランザクションは不変になってよく、その機能は、単に暗号通貨の形でデジタル資産に対する制御の不変の移転を越えて使用される可能性がある。不変トランザクションは、価値の移転に加えて、事象の公証記録、パーティの権利義務がトランザクションに符号化されるスマートコントラクトの実装、及びブロックチェーンプロトコルに従ったスマートコントラクトの条件に従い価値を移転するなど、他のオペレーションを実行するために使用することができる。

【００３５】

Bitcoinプラットフォームでは、スクリプトはスタックベースのスクリプト言語を使用して書かれるが、代わりに他のアプローチを使用してもよい。いくつかの例において、「スタックベースのスクリプト言語」は、種々のスタックベース又はスタック指向の実行モデル及び演算をサポートするプログラミング言語を指す。スタックベースのスクリプト言語の命令を実行するとき、プロセッサ（ブロックチェーンノードの一部など）は、スタックと呼ばれる先入れ先出しのデータ構造にデータを格納する。プロセッサは、スタックの最上部に値をプッシュしたり、スタックの最上部から値をポップすることができる。スタックに対して実行される種々の操作は、結果として、スタックの最上部から1つ以上の値をプッシュ又はポップしたり、該値に対して演算を実行したり、スタック上の要素の順序を変更したりすることができる（これは、2回のポップ操作と、最初のプッシュが最初のアイテムをポップした状態での2回のプッシュと等価であり得る）。たとえば、OP_EQUAL演算はスタックから上位2つのアイテムをポップし、それらを比較し、結果（等しい場合は1、等しくない場合は0）をスタックの一番上にプッシュし得る。OP_PICKのようなスタックに対して実行される他の操作は、スタックの最上部以外の位置からアイテムを選択することを可能にし得る。本発明の実施形態のいくつかによって使用されるいくつかのスクリプト言語では、少なくとも2つのスタック、すなわち、メインスタックと代替スタックが存在し得る。スクリプト言語の一部の操作は、あるスタックの最上部から別のスタックの最上部にアイテムを移動できる。例えば、OP_TOALTSTACK演算を実行すると、プロセッ

サはメインスタックの最上部から代替スタックの最上部に値を移動させる。スタックベースのスクリプト言語は、場合によっては、厳密に後入先出（LIFO）方式での動作のみに限定されないことに留意されたい。例えば、スタックベースのスクリプト言語は、スタック内のn番目のアイテムを最上位にコピー又は移動する操作（例えば、それぞれ、BitcoinにおけるOP_PICK及びOP_ROLL）をサポートしてもよい。スタックベースのスクリプト言語で書かれたスクリプトは、ベクトル、リスト、又はスタックのような任意の適切なデータ構造を使用して実装可能な論理スタック上にプッシュ可能である。

【００３６】

トランザクションに含まれるスクリプトを使用して、契約の条件がスクリプトに符号化されるスマートコントラクトを実装することができる。例えば、「BobがCarolにXを支払い、Daveがトランザクションに許可を与えた場合、AliceはBobにXの半分を支払う」という形式があり、これは、（とりわけ）BobがCarolに支払った事前のトランザクションとDaveが許可を符号化した事前のトランザクションがある場合にのみ、TRUEに評価されるロックスクリプトの一部として符号化することができる。ここで使用される「事前のトランザクション」とは、ブロックチェーンにすでに追加されているトランザクションをいい、必ずしもアンロックトランザクションがアンロックするアウトプットを有する前のトランザクションとは限らない。事実上、スマートコントラクトは、結果を生成するためのインプットを定義するルールを含む機械実行可能プログラムを表すことができ、それによって、それらの結果に依存するアクションの実行を生じることができる。

【００３７】

価値の移転に加えて、トランザクションは、価値の他のオブジェクト又は占有権を移転することもある。例えば、あるトランザクションは、名目上の暗号通貨価値を移転するかも知れないが、「このトランザクションを解除することができる者は、Marley Circle 123の家及び土地の正当な所有者でもある」という主張に相当するデータを含んでいるかも知れない。占有権は、同じ趣旨で公記録では不明瞭かも知れない。例えば、トランザクションは「このトランザクションをアンロックできる者は、Central Bank of Trustに保持されている委託番号12345で委託された特定の財産の正当な所有者でもある」という主張に相当するデータを含んでいるかも知れない。ここでは、これは、ブロックチェーンを介した実世界エンティティの移転を表し及び生じさせる、トークンと呼ばれる。潜在的に機密又は秘密のアイテムは、識別可能な意味又は価値を持たないトークンによって表すことができる。したがって、トークンは、実世界のアイテムがブロックチェーンから参照されることを可能にする識別子として機能し得る。

【００３８】

いくつかの実施形態では、特定のエンティティとの相互作用は、スマートコントラクト内の特定のステップで符号化され、スマートコントラクトは、そうでなければ、自動的に自動実行され、自己強制され得る。いくつかの例では、自動実行は、UTXOの移転を可能にするために実行されるスマートコントラクトの実行を表す。このような例では、UTXOをアンロックできる「任意のエンティティ」とは、何らかの秘密の知識を証明することを要求されることなく、アンロックスクリプトを生成できるエンティティを表す。言い換えると、データのソースが暗号秘密（プライベート非対称鍵、対称鍵など）にアクセスできることを検証することなく、アンロックスクリプトが生成され得る。また、このような例では、ブロックチェーンネットワークの検証ノードがスマートコントラクトの制約に応じてアンロックトランザクションを検証した結果として、自己強制が発生する。幾つかの例では、UTXOの「アンロック」は、UTXOを参照し及び有効として実行するアンロックトランザクションアウトプットを生成することを表す。このようなトランザクションアウトプットをアンロックすることの副次的な効果は、ブロックチェーンネットワークがロックスクリプトとアンロックスクリプトを処理して、新しいトランザクション、すなわちアンロックトランザクションを検証することができることである。有効な場合、前のトランザクションのアウトプットは移転されたとみなされる。トランザクションアウトプットに特定の値を含め、誰でもそのアウトプットをアンロックできるようにすることによって、パーティが

そのようなアンロックトランザクションを作成する理由が存在し、したがって、スマートコントラクトのステップは、スマートコントラクトの参加者によってではなくても、ブロックチェーンノードを運用する他者によって実行される。

【００３９】

トランザクションの一部として記録されるロックスクリプトとアンロックスクリプトを形成するスクリプトは、不変であり得るので、ロックスクリプトは、通常、変更することができず、将来のトランザクションの参照部分は、トランザクションが確定されるときに不明である可能性があるため、参照することができない。トランザクションインプットのアンロックスクリプトは、トランザクションインプットが指す前のトランザクションのアウトプットの部分、又は前のトランザクション以外のブロックチェーン内の前のトランザクションを参照することができる。これにより、トランザクションの使用方法が制限される場合がある。

【００４０】

これらの態様の1つ以上において、ブロックチェーン技術を使用するための追加の機能及び改良された方法及びシステムを提供することが望ましい。従って、本発明によれば、添付の特許請求の範囲に定義されるシステム及び／又は方法が提供される。

【発明の概要】

【００４１】

コンピュータにより実施される方法の種々の実施形態では、方法は、

第1使用トランザクションアウトプットに対する第1制約セットを決定するステップと、

第2使用トランザクションアウトプットに対する第2制約セットを決定するステップと、

初期トランザクションを生成するステップであって、前記初期トランザクションは、

前記第1制約セット及び前記第2制約セットを含む少なくとも1つの初期ロックスクリプトと、

少なくとも1つの使用可能な価値と、を含み、前記少なくとも1つの使用可能な価値を使用することは、

少なくとも部分的に、使用トランザクションが前記第1使用トランザクションアウトプットを含むことを検証することにより、前記第1制約セットが満たされること、及び、

少なくとも部分的に、前記使用トランザクションが前記第2使用トランザクションアウトプットを含むことを検証することにより、前記第2制約セットが満たされること、を条件とする、ステップと、

ブロックチェーンネットワークのノードにおいて、前記初期トランザクションを検証させるステップと、

を含む。

【００４２】

第1使用トランザクションアウトプット内のロックスクリプトは、第2使用トランザクションアウトプット内のロックスクリプトの複製であってよい。

【００４３】

第1使用トランザクションアウトプット内のロックスクリプトは、第2使用トランザクションアウトプット内のロックスクリプトと異なってよい。

【００４４】

第1使用トランザクションアウトプット内のロックスクリプトは、少なくとも1つの初期ロックスクリプトの少なくとも一部を含んでよい。

【００４５】

前記少なくとも1つの初期ロックスクリプトの実行は、前記少なくとも1つの初期ロックスクリプトの複数の部分から前記少なくとも一部を選択してよい。

【００４６】

前記使用トランザクションのアンロックスクリプトの実行の結果、前記少なくとも1つの初期ロックスクリプトが、前記第1使用トランザクションアウトプット又は前記第2使用

トランザクションアウトプットのうちの一方に対応するデータを受信してよい。
【００４７】
データはインデックス値を有してよい。インデックス値が第1インデックス値であることを条件として、少なくとも1つの初期ロックスクリプトの実行が、第1制約セットが満たされるか否かを決定してよい。インデックス値が第2インデックス値であることを条件として、少なくとも1つの初期ロックスクリプトの実行が、第2制約セットが満たされるか否かを決定してよい。
【００４８】
データは、新しいロックスクリプトを含んでよい。データを受信した結果として、第1使用トランザクションアウトプットは、新しいロックスクリプトを含むよう制約されてよい。
【００４９】
少なくとも1つの初期ロックスクリプトは、データのソースの制約を含んでよい。
【００５０】
方法は、さらに、第1使用トランザクションアウトプットの使用可能な価値を決定するステップを含んでよい。
【００５１】
初期トランザクションは、複数の状態を有するコントラクトを符号化してよい。
【００５２】
使用トランザクションは、複数の状態に対応する複数のインプット値を含んでよい。
【００５３】
第1制約セットは、第1使用トランザクションアウトプットが第1状態を有するよう制約してよい。第2制約セットは、第2使用トランザクションアウトプットが第2状態を有するよう制約する。
【００５４】
さらに、コンピュータにより実施される方法であって、
前のトランザクションのアウトプットを参照する使用トランザクションインプットを含むよう使用トランザクションを制約する使用トランザクション制約を決定するステップと、
使用可能トランザクションを生成するステップであって、使用可能トランザクションは、
使用可能量を含む使用可能トランザクションアウトプットと、
使用トランザクション制約を含む使用可能トランザクションロックスクリプトと、を含み、使用可能量の使用は、使用トランザクション制約を満たす使用トランザクションの少なくとも1つのアンロックスクリプトの実行を条件とする、ステップと、
ブロックチェーンネットワークのノードにおいて使用可能トランザクションを検証させるステップと、を含む方法を提供することが望ましい。
【００５５】
使用トランザクション制約は、さらに、特定のハッシュ値を含むよう使用トランザクションインプットを制約してよい。
【００５６】
特定のハッシュ値は、前のトランザクションのアウトプットを参照する識別子を符号化してよい。
【００５７】
使用トランザクション制約は、さらに、使用可能トランザクションロックスクリプトから複製されたスクリプト要素セットを含むよう使用トランザクションのロックスクリプトを制約してよい。
【００５８】
使用可能トランザクションは、複数の状態を有するコントラクトを符号化してよい。
【００５９】

方法は、さらに、使用トランザクションのアウトプットの使用可能な価値を決定するステップを含んでよい。

【００６０】

使用トランザクション制約は、前のトランザクションのアウトプットからの特定のロックスクリプト要素を含んでよい。使用トランザクションインプットが特定のロックスクリプト要素を含む結果として、少なくとも1つのアンロックスクリプトの実行は、使用トランザクション制約を満たしてよい。

【００６１】

特定のロックスクリプト要素は、特定のエンティティの暗号鍵を符号化してよい。

【００６２】

使用トランザクション制約は、第1使用トランザクション制約であってよく、方法は、さらに、

使用トランザクションを更に制約するために、第2使用トランザクション制約を決定するステップと、

第2使用可能トランザクションを生成するステップであって、第2使用可能トランザクションは、

第2使用可能量と、

第2使用トランザクション制約を含む第2使用可能トランザクションロックスクリプトと、を含み、第2使用可能量を使用することは、さらに、少なくとも1つのアンロックスクリプトの実行が第2使用トランザクション制約を満たすことを条件としてよい、ステップと、を含んでよい。

【００６３】

使用可能トランザクションロックスクリプトは、第1使用可能トランザクションロックスクリプトであってよく、少なくとも1つのアンロックスクリプトは、第1アンロックスクリプト及び第2アンロックスクリプトを含んでよい。第1使用トランザクション制約は、第1使用可能トランザクションロックスクリプトの少なくとも一部を含むよう、第1アンロックスクリプトを制約してよい。第2使用トランザクション制約は、第2使用可能トランザクションロックスクリプトの少なくとも一部を含むよう、第2アンロックスクリプトを制約してよい。

【００６４】

第1使用可能トランザクションロックスクリプトの少なくとも一部は、第1エンティティに関連付けられた暗号鍵を含んでよい。第2使用可能トランザクションロックスクリプトの少なくとも一部は、第1エンティティと異なる第2エンティティに関連付けられた暗号鍵を含んでよい。

【００６５】

使用トランザクション制約は、さらに、使用トランザクションのアウトプットを制約するよう使用トランザクションを制約してよい。

【００６６】

使用トランザクション制約は、コントラクトと区別される別のコントラクトを符号化してよい。使用可能量を使用することは、他のコントラクトが使用トランザクションのアウトプットの中に実装されていることを条件としてよい。

【００６７】

さらに、コンピュータにより実施される方法であって、

ブロックチェーントランザクションを生成するステップであって、ブロックチェーントランザクションは、

第1状態であり且つ許容状態遷移セットを有する状態機械と、

使用トランザクションに、許容状態遷移セットに従い状態機械を復号化させ、使用トランザクションへのインプットに対する制限に従わせ、又は、使用トランザクションのアウトプットに対する制限に従わせる、ために実行されるべきスクリプト要素セットと、を符号化する、ステップと、

ブロックチェーントランザクションを、ブロックチェーンネットワークのノードにより検証させるステップと、を含む方法を提供することが望ましい。

【００６８】

許容状態遷移セットは、別のブロックチェーントランザクションにおいて独立に並行に実装可能な状態遷移を含んでよい。

【００６９】

スクリプト要素セットは、使用トランザクションを、使用トランザクションのアウトプットに対する制限に従わせてよい。アウトプットに対する制限は、使用トランザクションの第1アウトプットが、許容状態遷移セットに従い第1アウトプットに対応する状態を示すこと、及び、使用トランザクションの第2アウトプットが、許容状態遷移セットに従い第2アウトプットに対応する状態を示すこと、を要求してよい。

【００７０】

第1アウトプットに対応する状態及び第2アウトプットに対応する状態は同じであってよい。

【００７１】

第1アウトプットに対応する状態及び第2アウトプットに対応する状態は異なる状態であってよい。

【００７２】

スクリプト要素セットは、使用トランザクションを、使用トランザクションのインプットに対する制限に従わせてよい。インプットに対する制限は、使用トランザクションへのインプットが、別のブロックチェーントランザクション内に符号化された別の状態機械を示すことを要求してよい。

【００７３】

インプットに対する制限は、インプットが第1状態の状態機械を参照すること、及び第1状態と異なる別の状態の他の状態機械を参照することを要求してよい。

【００７４】

第1状態から第2状態への第1遷移は、許容状態遷移セットの範囲内にあってよい。他の状態から第2状態への第2遷移は、許容状態遷移セットの範囲内にあってよい。

【００７５】

スクリプト要素セットは、使用トランザクションを、使用トランザクションのインプットに対する制限に従わせてよい。インプットに対する制限は、状態機械を進展させるための条件セットを記述してよい。状態機械を進展させるための条件セットは、別の状態機械の状態に依存してよい。

【００７６】

他の状態機械は、別のブロックチェーントランザクション内に符号化されてよい。他のトランザクションは、他の状態機械を進展させるための条件セットを符号化してよい。他の状態機械を進展させるための他の条件セットは、状態機械の状態に依存してよい。

【００７７】

スクリプト要素セットは、使用トランザクションを、使用トランザクションのアウトプットに対する制限に従わせてよい。アウトプットの制限は、使用トランザクションが、スマートコントラクトの要素セットをアウトプットに組み込むことを要求してよい。

【００７８】

スマートコントラクトの要素セットは、別のブロックチェーントランザクションから取得されてよい。

【００７９】

スクリプト要素セットは、使用トランザクションを、使用トランザクションのインプットに対する制限及び使用トランザクションのアウトプットに対する制限の両方に従わせてよい。

【００８０】

さらに、コンピュータにより実施される方法であって、

ブロックチェーンネットワークのノードにおいて、第1使用トランザクションの第1使用トランザクションインプットにより使用可能なトランザクションアウトプットを有するブロックチェーントランザクションを生成するステップと、
状態機械の動作に対応する第1使用トランザクションスクリプト要素セットを含むよう、第1使用トランザクションを制約し得る、第1スクリプト要素セットを、ブロックチェーントランザクションに挿入するステップと、
ブロックチェーンネットワークにより並行に独立して処理可能な異なるブロックチェーントランザクションを用いて実装可能な状態を含む許容状態機械状態遷移に対応する、第2使用トランザクションスクリプト要素セットを制約し得る第2スクリプト要素セットを、ブロックチェーントランザクションに挿入するステップと、を含む方法を提供することが望ましい。ブロックチェーントランザクションは、第1ロックスクリプトを有する第1アウトプットを含むことができ、第1トランザクションアウトプット値は、第2ロックスクリプト及び第2トランザクションアウトプット値を有する第2アウトプットを含むことができ、第1スクリプト要素セットは、第1ロックスクリプトの部分である。

【００８１】

使用トランザクションは、アンロックスクリプトを含むことができる。アンロックスクリプトは、トランザクション検証器により、前のトランザクションのアウトプットのロックスクリプトを用いて実行されると、
所定の検証テストを満たし、前のトランザクションのアウトプットのロックスクリプトを実行し、使用トランザクションに第1スクリプト要素セットを挿入し、ここで第1スクリプト要素セットは使用トランザクションの第1アウトプットの使用トランザクションロックスクリプトの一部を形成し及び前のトランザクションロックスクリプトにより指示される要件に一致し、第1スクリプト要素セットは状態機械の動作に対応し、並びに、使用トランザクションに、第2スクリプト要素セットを挿入し、ここで、第2スクリプト要素セットは使用トランザクションロックスクリプトの一部を形成し及び前のトランザクションロックスクリプトにより指示される要件に一致し、ブロックチェーンネットワークにより並行に独立して処理可能な異なるブロックチェーントランザクションを用いて実装可能な状態を含む許容状態機械状態遷移に対応するとき、ノードに、使用トランザクションのフィールドを表す値を、トランザクション検証器がアクセス可能なメモリに少なくとも格納させる。

【００８２】

使用トランザクションは、インプットとアウトプットとを含むことができ、該インプットは前のトランザクションのアウトプットを参照する。使用トランザクションは、アンロックスクリプトを含む。アンロックスクリプトは、トランザクション検証器により前のトランザクションのアウトプットの前のトランザクションロックスクリプトを用いて実行されると、所定の検証テストを満たし、ここで、アンロックスクリプトは、ロックスクリプトを実行し、使用トランザクションに追加状態機械に対応するデータを許容状態機械状態遷移に基づく使用トランザクションインプット及び／又は使用トランザクションアウトプットとして挿入し、その結果、使用トランザクションで使用される使用トランザクションインプット及び／又は使用トランザクションアウトプットの数が状態機械遷移の数に対して十分であるとき、ノードに、使用トランザクションのフィールドを表す値を、トランザクション検証器がアクセス可能なメモリに、少なくとも格納させる。

【００８３】

許容状態機械状態遷移は、使用トランザクションについて、第1後段状態に対応する第1トランザクションアウトプットを生成し、第1トランザクションアウトプットを第1トランザクションアウトプットを参照する第1後段トランザクションのインプットにさせるのに十分な第1トランザクションアウトプットの第1トランザクションアウトプット値を、使用トランザクションに挿入し、使用トランザクションの第1トランザクションアウトプットに、第3スクリプト要素セットを挿入し、ここで、第3スクリプト要素セットは、第1後段状態を取るよう第1後段トランザクションを制約する第1ロックスクリプトの部分であり、

使用トランザクションについて、第2後段状態に対応する第2トランザクションアウトプットを生成し、使用トランザクションに、第2トランザクションアウトプットを第2トランザクションアウトプットを参照する第2後段トランザクションのインプットにさせるのに十分な第2トランザクションアウトプットの第2トランザクションアウトプット値を挿入し、第2トランザクションアウトプットに、第4スクリプト要素セットを挿入し、ここで、第4スクリプト要素セットは、第2後段状態を取るよう第2後段トランザクションを制約する第2ロックスクリプトの部分であり、並びに、第1トランザクションアウトプット又は第2トランザクションアウトプットに、第5スクリプト要素セットを挿入し、ここで、第5スクリプト要素セットは第1ロックスクリプト、又は第1後段トランザクション及び第2後段トランザクションが区別される制約を課す第2ロックスクリプトの部分である、ことにより、1つの初期状態から2つの後段状態への分岐トランザクションを含み得る。

【００８４】

第1ロックスクリプトは、第1状態を有するよう第1後段トランザクションを制約できる。第2ロックスクリプトは、第2状態を有するよう第2後段トランザクションを制約できる。使用トランザクションは、3個以上の状態の分岐に対応する3個以上のトランザクションアウトプット値を含むことができる。使用トランザクションは、3個以上の状態の融合に対応する3個以上のトランザクションインプット値を含むことができる。第1ロックスクリプト及び第2ロックスクリプトは、第1後段トランザクション及び第2後段トランザクションが同じ状態を有するよう制約できる。

【００８５】

通常、使用トランザクションは、N個のトランザクションインプット値と、N個のトランザクションアウトプット値と、を含んでよく、Nは1、2、3、又はそれ以上である。

【００８６】

許容状態機械状態遷移は、使用トランザクションに、第1前段状態の第1トランザクションインプットを挿入し、使用トランザクションに、第2前段状態の第2トランザクションインプットを挿入し、使用トランザクションに、状態遷移マトリクスに従い第1前段状態及び第2前段状態からの許容遷移である融合状態を挿入することにより、2つの初期状態から1つの後段状態への融合（merging）トランザクションを含むことができる。

【００８７】

許容状態機械状態遷移は、第1トランザクションインプットに、第1前段状態を有する第1の前のトランザクションへの第1参照を挿入し、第2トランザクションインプットに、第2前段状態を有する第2の前のトランザクションへの第2参照を挿入し、及び、使用トランザクションに、融合状態の第1トランザクションアウトプットを挿入することにより、第1トランザクションインプット及び／又は第2トランザクションインプットにハードコードすることができる。ここで、状態遷移マトリクスは、第1の前のトランザクション及び第2の前のトランザクションに利用可能である。

【００８８】

許容状態機械状態遷移は、使用トランザクションに、第1前段状態の第1トランザクションインプットを挿入し、使用トランザクションに、第2前段状態の第2トランザクションインプットを挿入し、第1トランザクションインプットに、第1前段状態を有する第1の前のトランザクションへの第1参照を挿入し、第2トランザクションインプットに、第2前段状態を有する第2の前のトランザクションへの第2参照を挿入し、使用トランザクションに、第1後段状態の第1トランザクションアウトプットを挿入し、第1トランザクションアウトプットに、第3スクリプト要素セットを挿入し、ここで、第3スクリプト要素セットは第1後段状態を取るよう第1後段トランザクションを制約する第1ロックスクリプトの部分であり、使用トランザクションに、第2後段状態の第2トランザクションアウトプットを挿入し、第2トランザクションアウトプットに、第4スクリプト要素セットを挿入し、ここで、第4スクリプト要素セットは、第2後段状態を取るよう第2後段トランザクションを制約する第2ロックスクリプトの部分である、ことにより、初期状態の数Nから、N個の後続状態への並列動作を含むことができる。Nは2以上である。

【００８９】
状態機械の動作は、ブロックチェーンを用いて実装されるスマートコントラクトのために符号化してよい。
【００９０】
トランザクションは、任意の制約により分割され得る。方法は、ブロックチェーンネットワークのノードにおいて、第1トランザクションを生成するステップであって、第1トランザクションは、第1使用可能価値を有する第1アウトプットと、第1ロックスクリプトとを含み、第1トランザクションは、第2使用可能価値を有する第2アウトプットと、第2ロックスクリプトとを含む、ステップと、第1ロックスクリプトに、第1の選択されたトランザクションアウトプットに対する第1制約セットを含めるステップであって、第1制約セットは、第1の選択されたトランザクションアウトプットが第1使用可能価値を使用するために検証されるべきである場合、第1アンロックスクリプトにより満たされる、ステップと、第2ロックスクリプトに、第2の選択されたトランザクションアウトプットに対する第2制約セットを含めるステップであって、第2制約セットは、第2の選択されたトランザクションアウトプットが、第2使用可能価値を使用するために検証されるべきである場合、第2アンロックスクリプトにより満たされ、第1制約セット又は第2制約セットは、使用トランザクションアウトプットとして、第1の選択されたトランザクションアウトプット又は第2の選択されたトランザクションアウトプットを有する使用トランザクションのロックスクリプトに対して制約を課す、ステップと、を含み得る。
【００９１】
第1の選択されたトランザクションアウトプットは、第1使用トランザクションのアウトプットになることができ、第2の選択されたトランザクションアウトプットは、第1使用トランザクションと異なる第2使用トランザクションのアウトプットになることができる。第1制約セットは、第2制約セットと異なり、第1トランザクションのトランザクションインプットに対する制約と異なり得る。第1制約セット及び第2制約セットは、それぞれ、共通制約セットを含むことができる。使用トランザクションのロックスクリプトに対する制約は、使用トランザクションが有効であるために、使用トランザクションのロックスクリプトが、使用トランザクションの少なくとも1つのフィールドへの参照を含むことを要求し得る。ロックスクリプトに対する別の制約は、使用トランザクションが有効であるために、使用トランザクションのロックスクリプトが、第1ロックスクリプトの一部のコピー及び／又は第2ロックスクリプトの一部のコピーを含むことを要求する。
【００９２】
第1の選択されたトランザクションアウトプットは、第1使用トランザクションのアウトプットであり得る。第2の選択されたトランザクションアウトプットは、第2使用トランザクションのアウトプットであり得る。第1アンロックスクリプトは、第1トランザクションのフィールド、第1の選択されたトランザクションのフィールド、及び第2使用トランザクションのフィールドを含み得る。第1ロックスクリプトは、第1トランザクションの抽出したフィールドの第1の比較、及び第1使用トランザクションの抽出したフィールドの第2の比較を含み得る。
【００９３】
第1制約セット及び第2制約セットは、それぞれ、共通制約セットを含み得る。それにより、2つの制約された使用トランザクションアウトプットロックスクリプトは、共通制約セットにより制約される。第1制約セット及び／又は第2制約セットは、第1ロックスクリプトの一部のハッシュ、及び／又は第2ロックスクリプトの一部のハッシュ、及び／又は第1ロックスクリプトの一部のサイズ及び／又は第2ロックスクリプトの一部のサイズに対する制約、を含み得る。
【００９４】
第1制約セット及び／又は第2制約セットは、インプットデータへの任意の参照を含むことができる。第1トランザクションの複数のアウトプットの各々は、複数のアウトプットのうちのアウトプットに適用される異なる制約により制約され得る。

【００９５】
トランザクションは、コンピュータにより実施される方法によるような、任意の制約との相互依存を融合し有し得る。該方法は、ブロックチェーンネットワークのノードにおいて、第1使用可能トランザクションを生成するステップであって、第1使用可能トランザクションは、第1使用可能価値を有する第1使用可能アウトプットと、第1ロックスクリプトとを含む、ステップと、第1ロックスクリプトに、使用トランザクションに対する第1制約セットを含めるステップであって、第1制約セットは、使用トランザクションが第1使用可能アウトプットを使用するために有効であるべき場合、使用トランザクションの第1アンロックスクリプトにより満たされるものである、ステップと、第1制約セットに、使用トランザクションの第1使用トランザクションアウトプットが第1スマートコントラクトスクリプト命令セットを含むための第1使用トランザクションアウトプットロックスクリプトを含むことを要求する第1使用トランザクション制約を含めるステップと、第1制約セットに、使用トランザクションが第2スマートコントラクトスクリプト命令セットを含むことを要求し、及び使用トランザクションが第2使用可能トランザクションアウトプットを参照する使用トランザクションインプットを含むことを要求する、第2使用トランザクション制約を含めるステップと、を含む。
【００９６】
第1スマートコントラクトスクリプト命令セットは、第1ロックスクリプトから複製されたスクリプト命令、及び／又は第1ロックスクリプト内に実装されたスマートコントラクトと異なる新しいスマートコントラクトに対応するスクリプト命令を含むことができる。
【００９７】
第1制約セットは、第1使用トランザクションアウトプットロックスクリプトが、第2使用可能トランザクションアウトプットへの参照を含むこと、及び第2スマートコントラクトスクリプト命令セットが、第1使用可能トランザクションのアウトプットへの参照を含むこと、であってよい。
【００９８】
方法は、第1制約セットの中で、使用トランザクションが第1使用可能トランザクションを参照する第1使用トランザクションインプットを含むことを要求する第1並列制約と、使用トランザクションが第2使用可能トランザクションアウトプットを参照する第2使用トランザクションインプットを含むことを要求する第2並列制約と、第1使用トランザクションアウトプットロックスクリプトが第1スマートコントラクトスクリプト命令セットの中に少なくとも部分的に第1ロックスクリプトにより決定された少なくとも第1スクリプト命令セットを含むことを要求する第3並列制約と、第2使用トランザクションアウトプットロックスクリプトが少なくとも部分的に第2使用可能トランザクションアウトプットのロックスクリプトにより決定された第2スマートコントラクトスクリプト命令セットを含むことを要求する第4並列制約と、を提供するステップを含んでよい。
【００９９】
第1制約セットは、使用トランザクションが第1使用可能トランザクションを参照する第1使用トランザクションインプットを含むことを要求する第1並列制約と、使用トランザクションが第2使用可能トランザクションアウトプットを参照する第2使用トランザクションインプットを含むことを要求する第2並列制約と、第1使用トランザクションアウトプットロックスクリプトが第1スマートコントラクトスクリプト命令セットに第1ロックスクリプトから複製された少なくとも第1スクリプト命令セットを含むことを要求する第3並列制約と、第2使用トランザクションアウトプットロックスクリプトが第2使用可能トランザクションアウトプットのロックスクリプトから複製された第2スマートコントラクトスクリプト命令セットを含むことを要求する第4並列制約と、を含んでよい。
【０１００】
さらに、コンピュータにより実施される方法であって、
第1アンロックトランザクションアウトプットに対する第1制約セットを決定するステップと、

第2アンロックトランザクションアウトプットに対する第2制約セットを決定するステップと、
初期トランザクションを生成するステップであって、前記初期トランザクションは、
前記第1制約セット及び前記第2制約セットを含む少なくとも1つの初期ロックスクリプトと、
少なくとも1つの償還可能な価値と、を含み、前記少なくとも1つの償還可能な価値をアンロックすることは、
少なくとも部分的に、アンロックトランザクションが前記第1アンロックトランザクションアウトプットを含むことを検証することにより、前記第1制約セットが満たされること、及び、
少なくとも部分的に、前記アンロックトランザクションが前記第2アンロックトランザクションアウトプットを含むことを検証することにより、前記第2制約セットが満たされること、を条件とする、ステップと、
ブロックチェーンネットワークのノードにおいて、前記初期トランザクションを検証させるステップと、
を含むコンピュータにより実施される方法を提供することが望ましい。

【０１０１】
第1アンロックトランザクションアウトプット内のロックスクリプトは、第2アンロックトランザクションアウトプット内のロックスクリプトの複製であってよい。

【０１０２】
第1アンロックトランザクションアウトプット内のロックスクリプトは、第2アンロックトランザクションアウトプット内のロックスクリプトと異なってよい。

【０１０３】
第1アンロックトランザクションアウトプット内のロックスクリプトは、少なくとも1つの初期ロックスクリプトの少なくとも一部を含んでよい。

【０１０４】
前記少なくとも1つの初期ロックスクリプトの実行は、前記少なくとも1つの初期ロックスクリプトの複数の部分から前記少なくとも一部を選択してよい。

【０１０５】
前記アンロックトランザクションのアンロックスクリプトの実行の結果、前記少なくとも1つの初期ロックスクリプトが、前記第1アンロックトランザクションアウトプット又は前記第2アンロックトランザクションアウトプットのうちの一方に対応するデータを受信してよい。

【０１０６】
データはインデックス値を有してよい。インデックス値が第1インデックス値であることを条件として、少なくとも1つの初期ロックスクリプトの実行が、第1制約セットが満たされるか否かを決定してよい。インデックス値が第2インデックス値であることを条件として、少なくとも1つの初期ロックスクリプトの実行が、第2制約セットが満たされるか否かを決定してよい。

【０１０７】
データは、新しいロックスクリプトを含んでよい。データを受信した結果として、第1アンロックトランザクションアウトプットは、新しいロックスクリプトを含むよう制約されてよい。

【０１０８】
少なくとも1つの初期ロックスクリプトは、データのソースの制約を含んでよい。

【０１０９】
方法は、さらに、第1アンロックトランザクションアウトプットの償還可能な価値を決定するステップを含んでよい。

【０１１０】
初期トランザクションは、複数の状態を有するコントラクトを符号化してよい。

【０１１１】
アンロックトランザクションは、複数の状態に対応する複数のインプット値を含んでよい。
【０１１２】
第1制約セットは、第1アンロックトランザクションアウトプットが第1状態を有するよう制約してよい。第2制約セットは、第2アンロックトランザクションアウトプットが第2状態を有するよう制約する。
【０１１３】
さらに、コンピュータにより実施される方法であって、
前のトランザクションのアウトプットを参照するアンロックトランザクションインプットを含むようアンロックトランザクションを制約するアンロックトランザクション制約を決定するステップと、
償還可能トランザクションを生成するステップであって、償還可能トランザクションは、
償還可能量を含む償還可能トランザクションアウトプットと、
アンロックトランザクション制約を含む償還可能トランザクションロックスクリプトと、を含み、償還可能量のアンロックは、アンロックトランザクション制約を満たすアンロックトランザクションの少なくとも1つのアンロックスクリプトの実行を条件とする、ステップと、
ブロックチェーンネットワークのノードにおいて償還可能トランザクションを検証させるステップと、を含む方法を提供することが望ましい。
【０１１４】
アンロックトランザクション制約は、さらに、特定のハッシュ値を含むようアンロックトランザクションインプットを制約してよい。
【０１１５】
特定のハッシュ値は、前のトランザクションのアウトプットを参照する識別子を符号化してよい。
【０１１６】
アンロックトランザクション制約は、さらに、償還可能トランザクションロックスクリプトから複製されたスクリプト要素セットを含むようアンロックトランザクションのロックスクリプトを制約してよい。
【０１１７】
償還可能トランザクションは、複数の状態を有するコントラクトを符号化してよい。
【０１１８】
方法は、さらに、アンロックトランザクションのアウトプットの償還可能な価値を決定するステップを含んでよい。
【０１１９】
アンロックトランザクション制約は、前のトランザクションのアウトプットからの特定のロックスクリプト要素を含んでよい。アンロックトランザクションインプットが特定のロックスクリプト要素を含む結果として、少なくとも1つのアンロックスクリプトの実行は、アンロックトランザクション制約を満たしてよい。
【０１２０】
特定のロックスクリプト要素は、特定のエンティティの暗号鍵を符号化してよい。
【０１２１】
アンロックトランザクション制約は、第1アンロックトランザクション制約であってよく、方法は、さらに、
アンロックトランザクションを更に制約するために、第2アンロックトランザクション制約を決定するステップと、
第2償還可能トランザクションを生成するステップであって、第2償還可能トランザクションは、

第2償還可能量と、
第2アンロックトランザクション制約を含む第2償還可能トランザクションロックスクリプトと、を含み、第2償還可能量をアンロックすることは、さらに、少なくとも1つのアンロックスクリプトの実行が第2アンロックトランザクション制約を満たすことを条件としてよい、ステップと、を含んでよい。
【０１２２】
償還可能トランザクションロックスクリプトは、第1償還可能トランザクションロックスクリプトであってよく、少なくとも1つのアンロックスクリプトは、第1アンロックスクリプト及び第2アンロックスクリプトを含んでよい。第1アンロックトランザクション制約は、第1償還可能トランザクションロックスクリプトの少なくとも一部を含むよう、第1アンロックスクリプトを制約してよい。第2アンロックトランザクション制約は、第2償還可能トランザクションロックスクリプトの少なくとも一部を含むよう、第2アンロックスクリプトを制約してよい。
【０１２３】
第1償還可能トランザクションロックスクリプトの少なくとも一部は、第1エンティティに関連付けられた暗号鍵を含んでよい。第2償還可能トランザクションロックスクリプトの少なくとも一部は、第1エンティティと異なる第2エンティティに関連付けられた暗号鍵を含んでよい。
【０１２４】
アンロックトランザクション制約は、さらに、アンロックトランザクションのアウトプットを制約するようアンロックトランザクションを制約してよい。
【０１２５】
アンロックトランザクション制約は、コントラクトと区別される別のコントラクトを符号化してよい。償還可能量をアンロックすることは、他のコントラクトがアンロックトランザクションのアウトプットの中に実装されていることを条件としてよい。
【０１２６】
さらに、コンピュータにより実施される方法であって、
ブロックチェーントランザクションを生成するステップであって、ブロックチェーントランザクションは、
第1状態であり且つ許容状態遷移セットを有する状態機械と、
アンロックトランザクションに、許容状態遷移セットに従い状態機械を復号化させ、アンロックトランザクションへのインプットに対する制限に従わせ、又は、アンロックトランザクションのアウトプットに対する制限に従わせる、ために実行されるべきスクリプト要素セットと、を符号化する、ステップと、
ブロックチェーントランザクションを、ブロックチェーンネットワークのノードにより検証させるステップと、を含む方法を提供することが望ましい。
【０１２７】
許容状態遷移セットは、別のブロックチェーントランザクションにおいて独立に並行に実装可能な状態遷移を含んでよい。
【０１２８】
スクリプト要素セットは、アンロックトランザクションを、アンロックトランザクションのアウトプットに対する制限に従わせてよい。アウトプットに対する制限は、アンロックトランザクションの第1アウトプットが、許容状態遷移セットに従い第1アウトプットに対応する状態を示すこと、及び、アンロックトランザクションの第2アウトプットが、許容状態遷移セットに従い第2アウトプットに対応する状態を示すこと、を要求してよい。
【０１２９】
第1アウトプットに対応する状態及び第2アウトプットに対応する状態は同じであってよい。
【０１３０】
第1アウトプットに対応する状態及び第2アウトプットに対応する状態は異なる状態であ

ってよい。
【０１３１】
スクリプト要素セットは、アンロックトランザクションを、アンロックトランザクションのインプットに対する制限に従わせてよい。インプットに対する制限は、アンロックトランザクションへのインプットが、別のブロックチェーントランザクション内に符号化された別の状態機械を示すことを要求してよい。
【０１３２】
インプットに対する制限は、インプットが第1状態の状態機械を参照すること、及び第1状態と異なる別の状態の他の状態機械を参照することを要求してよい。
【０１３３】
第1状態から第2状態への第1遷移は、許容状態遷移セットの範囲内にあってよい。他の状態から第2状態への第2遷移は、許容状態遷移セットの範囲内にあってよい。
【０１３４】
スクリプト要素セットは、アンロックトランザクションを、アンロックトランザクションのインプットに対する制限に従わせてよい。インプットに対する制限は、状態機械を発展させるための条件セットを記述してよい。状態機械を進展させるための条件セットは、別の状態機械の状態に依存してよい。
【０１３５】
他の状態機械は、別のブロックチェーントランザクション内に符号化されてよい。他のトランザクションは、他の状態機械を進展させるための条件セットを符号化してよい。他の状態機械を進展させるための他の条件セットは、状態機械の状態に依存してよい。
【０１３６】
スクリプト要素セットは、アンロックトランザクションを、アンロックトランザクションのアウトプットに対する制限に従わせてよい。アウトプットの制限は、アンロックトランザクションが、スマートコントラクトの要素セットをアウトプットに組み込むことを要求してよい。
【０１３７】
スマートコントラクトの要素セットは、別のブロックチェーントランザクションから取得されてよい。
【０１３８】
スクリプト要素セットは、アンロックトランザクションを、アンロックトランザクションのインプットに対する制限及びアンロックトランザクションのアウトプットに対する制限の両方に従わせてよい。
【０１３９】
さらに、コンピュータにより実施される方法であって、
ブロックチェーンネットワークのノードにおいて、第1アンロックトランザクションの第1アンロックトランザクションインプットにより移転可能なトランザクションアウトプットを有するブロックチェーントランザクションを生成するステップと、
状態機械の動作に対応する第1アンロックトランザクションスクリプト要素セットを含むよう、第1アンロックトランザクションを制約し得る、第1スクリプト要素セットを、ブロックチェーントランザクションに挿入するステップと、
ブロックチェーンネットワークにより並行に独立して処理可能な異なるブロックチェーントランザクションを用いて実装可能な状態を含む許容状態機械状態遷移に対応する、第2アンロックトランザクションスクリプト要素セットを制約し得る第2スクリプト要素セットを、ブロックチェーントランザクションに挿入するステップと、を含む方法を提供することが望ましい。ブロックチェーントランザクションは、第1ロックスクリプトを有する第1アウトプットを含むことができ、第1トランザクションアウトプット値は、第2ロックスクリプト及び第2トランザクションアウトプット値を有する第2アウトプットを含むことができ、第1スクリプト要素セットは、第1ロックスクリプトの部分であってよい。
【０１４０】

アンロックトランザクションは、アンロックスクリプトを含むことができる。アンロックスクリプトは、トランザクション検証器により、前のトランザクションのアウトプットのロックスクリプトを用いて実行されると、
所定の検証テストを満たし、
前のトランザクションのアウトプットのロックスクリプトを実行し、アンロックトランザクションに第1スクリプト要素セットを挿入し、ここで第1スクリプト要素セットはアンロックトランザクションの第1アウトプットのアンロックトランザクションロックスクリプトの一部を形成し及び前のトランザクションロックスクリプトにより指示される要件に一致してよく、第1スクリプト要素セットは状態機械の動作に対応し、並びに、アンロックトランザクションに、第2スクリプト要素セットを挿入し、ここで、第2スクリプト要素セットはアンロックトランザクションロックスクリプトの一部を形成し及び前のトランザクションロックスクリプトにより指示される要件に一致し、ブロックチェーンネットワークにより並行に独立して処理可能な異なるブロックチェーントランザクションを用いて実装可能な状態を含む許容状態機械状態遷移に対応するとき、ノードに、アンロックトランザクションのフィールドを表す値を、トランザクション検証器がアクセス可能なメモリに少なくとも格納させてよい。

【０１４１】

アンロックトランザクションは、インプットとアウトプットとを含むことができ、該インプットは前のトランザクションのアウトプットを参照する。アンロックトランザクションは、アンロックスクリプトを含む。アンロックスクリプトは、トランザクション検証器により前のトランザクションのアウトプットの前のトランザクションロックスクリプトを用いて実行されると、所定の検証テストを満たし、ここで、アンロックスクリプトは、ロックスクリプトを実行し、アンロックトランザクションに追加状態機械に対応するデータを許容状態機械状態遷移に基づくアンロックトランザクションインプット及び／又はアンロックトランザクションアウトプットとして挿入し、その結果、アンロックトランザクションで使用されるアンロックトランザクションインプット及び／又はアンロックトランザクションアウトプットの数が状態機械遷移の数に対して十分であるとき、ノードに、アンロックトランザクションのフィールドを表す値を、トランザクション検証器がアクセス可能なメモリに、少なくとも格納させる。

【０１４２】

許容状態機械状態遷移は、アンロックトランザクションについて、第1後段状態に対応する第1トランザクションアウトプットを生成し、第1トランザクションアウトプットを第1トランザクションアウトプットを参照する第1後段トランザクションのインプットにさせるのに十分な第1トランザクションアウトプットの第1トランザクションアウトプット値を、アンロックトランザクションに挿入し、アンロックトランザクションの第1トランザクションアウトプットに、第3スクリプト要素セットを挿入し、ここで、第3スクリプト要素セットは、第1後段状態を取るよう第1後段トランザクションを制約する第1ロックスクリプトの部分であってよく、アンロックトランザクションについて、第2後続状態に対応する第2トランザクションアウトプットを生成し、アンロックトランザクションに、第2トランザクションアウトプットを第2トランザクションアウトプットを参照する第2後段トランザクションのインプットにさせるのに十分な第2トランザクションアウトプットの第2トランザクションアウトプット値を挿入し、第2トランザクションアウトプットに、第4スクリプト要素セットを挿入し、ここで、第4スクリプト要素セットは、第2後段状態を取るよう第2後段トランザクションを制約する第2ロックスクリプトの部分であってよく、並びに、第1トランザクションアウトプット又は第2トランザクションアウトプットに、第5スクリプト要素セットを挿入し、ここで、第5スクリプト要素セットは第1ロックスクリプト、又は第1後段トランザクション及び第2後段トランザクションが区別される制約を課す第2ロックスクリプトの部分であってよい、ことにより、1つの初期状態から2つの後続状態への分岐トランザクションを含み得る。

【０１４３】

第1ロックスクリプトは、第1状態を有するよう第1後段トランザクションを制約できる。第2ロックスクリプトは、第2状態を有するよう第2後段トランザクションを制約できる。アンロックトランザクションは、3個以上の状態の分岐に対応する3個以上のトランザクションアウトプット値を含むことができる。アンロックトランザクションは、3個以上の状態の合併に対応する3個以上のトランザクションインプット値を含むことができる。第1ロックスクリプト及び第2ロックスクリプトは、第1後段トランザクション及び第2後段トランザクションが同じ状態を有するよう制約できる。

【０１４４】

通常、アンロックトランザクションは、N個のトランザクションインプット値と、N個のトランザクションアウトプット値と、を含んでよく、Nは1、2、3、又はそれ以上である。

【０１４５】

許容状態機械状態遷移は、アンロックトランザクションに、第1前段状態の第1トランザクションインプットを挿入し、アンロックトランザクションに、第2前段状態の第2トランザクションインプットを挿入し、アンロックトランザクションに、状態遷移マトリクスに従い第1前段状態及び第2前段状態からの許容遷移である融合状態を挿入することにより、2つの初期状態から1つの後段状態への融合（merging）トランザクションを含むことができる。

【０１４６】

許容状態機械状態遷移は、第1トランザクションインプットに、第1前段状態を有する第1の前のトランザクションへの第1参照を挿入し、第2トランザクションインプットに、第2前段状態を有する第2の前のトランザクションへの第2参照を挿入し、及び、アンロックトランザクションに、融合状態の第1トランザクションアウトプットを挿入することにより、第1トランザクションインプット及び／又は第2トランザクションインプットにハードコードすることができる。ここで、状態遷移マトリクスは、第1の前のトランザクション及び第2の前のトランザクションに利用可能である。

【０１４７】

許容状態機械状態遷移は、アンロックトランザクションに、第1前段状態の第1トランザクションインプットを挿入し、アンロックトランザクションに、第2前段状態の第2トランザクションインプットを挿入し、第1トランザクションインプットに、第1前段状態を有する第1の前のトランザクションへの第1参照を挿入し、第2トランザクションインプットに、第2前段状態を有する第2の前のトランザクションへの第2参照を挿入し、アンロックトランザクションに、第1後段状態の第1トランザクションアウトプットを挿入し、第1トランザクションアウトプットに、第3スクリプト要素セットを挿入し、ここで、第3スクリプト要素セットは第1後段状態を取るよう第1後段トランザクションを制約する第1ロックスクリプトの部分であり、アンロックトランザクションに、第2後段状態の第2トランザクションアウトプットを挿入し、第2トランザクションアウトプットに、第4スクリプト要素セットを挿入し、ここで、第4スクリプト要素セットは、第2後段状態を取るよう第2後段トランザクションを制約する第2ロックスクリプトの部分である、ことにより、初期状態の数Nから、N個の後続状態への並列動作を含むことができる。Nは2以上である。

【０１４８】

状態機械の動作は、ブロックチェーンを用いて実装されるスマートコントラクトのために符号化してよい。

【０１４９】

トランザクションは、任意の制約により分割され得る。方法は、ブロックチェーンネットワークのノードにおいて、第1トランザクションを生成するステップであって、第1トランザクションは、第1償還可能価値を有する第1アウトプットと、第1ロックスクリプトとを含み、第1トランザクションは、第2償還可能価値を有する第2アウトプットと、第2ロックスクリプトとを含む、ステップと、第1ロックスクリプトに、第1の選択されたトランザクションアウトプットに対する第1制約セットを含めるステップであって、第1制約セットは、第1の選択されたトランザクションアウトプットが第1償還可能価値をアンロックする

ために有効であるべきである場合、第1アンロックスクリプトにより満たされる、ステップと、第2ロックスクリプトに、第2の選択されたトランザクションアウトプットに対する第2制約セットを含めるステップであって、第2制約セットは、第2の選択されたトランザクションアウトプットが、第2償還可能価値を使用するために有効であるべきである場合、第2アンロックスクリプトにより満たされ、第1制約セット又は第2制約セットは、アンロックトランザクションのアウトプットとして、第1の選択されたトランザクションアウトプット又は第2の選択されたトランザクションアウトプットを有するアンロックトランザクションのロックスクリプトに対して制約を課す、ステップと、を含み得る。

【０１５０】

第1の選択されたトランザクションアウトプットは、第1アンロックトランザクションのアウトプットになることができ、第2の選択されたトランザクションアウトプットは、第1アンロックトランザクションと異なる第2アンロックトランザクションのアウトプットになることができる。第1制約セットは、第2制約セットと異なり、第1トランザクションのトランザクションインプットに対する制約と異なり得る。第1制約セット及び第2制約セットは、それぞれ、共通の制約セットを含むことができる。アンロックトランザクションのロックスクリプトに対する制約は、アンロックトランザクションが有効であるために、アンロックトランザクションのロックスクリプトが、アンロックトランザクションの少なくとも1つのフィールドへの参照を含むことを要求し得る。ロックスクリプトに対する別の制約は、アンロックトランザクションが有効であるために、アンロックトランザクションのロックスクリプトが、第1ロックスクリプトの一部のコピー及び／又は第2ロックスクリプトの一部のコピーを含むことを要求する。

【０１５１】

第1の選択されたトランザクションアウトプットは、第1アンロックトランザクションのアウトプットであり得る。第2の選択されたトランザクションアウトプットは、第2アンロックトランザクションのアウトプットであり得る。第1アンロックスクリプトは、第1トランザクションのフィールド、第1の選択されたトランザクションのフィールド、及び第2アンロックトランザクションのフィールドを含み得る。第1ロックスクリプトは、第1トランザクションの抽出したフィールドの第1の比較、及び第1アンロックトランザクションの抽出したフィールドの第2の比較を含み得る。

【０１５２】

第1制約セット及び第2制約セットは、それぞれ、共通制約セットを含み得る。それにより、2つの制約されたアンロックトランザクションのアウトプットのロックスクリプトは、共通制約セットにより制約されてよい。第1制約セット及び／又は第2制約セットは、第1ロックスクリプトの一部のハッシュ、及び／又は第2ロックスクリプトの一部のハッシュ、及び／又は第1ロックスクリプトの一部のサイズ及び／又は第2ロックスクリプトの一部のサイズに対する制約、を含み得る。

【０１５３】

第1制約セット及び／又は第2制約セットは、インプットデータへの任意の参照を含むことができる。第1トランザクションの複数のアウトプットの各々は、複数のアウトプットのうちのアウトプットに適用される異なる制約により制約され得る。

【０１５４】

トランザクションは、コンピュータにより実施される方法によるような、任意の制約との相互依存を融合し有し得る。該方法は、ブロックチェーンネットワークのノードにおいて、第1償還可能トランザクションを生成するステップであって、第1償還可能トランザクションは、第1償還可能価値を有する第1償還可能アウトプットと、第1ロックスクリプトとを含む、ステップと、第1ロックスクリプトに、アンロックトランザクションに対する第1制約セットを含めるステップであって、第1制約セットは、アンロックトランザクションが第1償還可能アウトプットをアンロックするために有効であるべき場合、アンロックトランザクションの第1アンロックスクリプトにより満たされるものである、ステップと、第1制約セットに、アンロックトランザクションの第1アンロックトランザクションアウ

トプットが第1スマートコントラクトスクリプト命令セットを含むための第1アンロックトランザクションのアウトプットのロックスクリプトを含むことを要求する第1アンロックトランザクション制約を含めるステップと、第1制約セットに、アンロックトランザクションが第2スマートコントラクトスクリプト命令セットを含むことを要求し、及びアンロックトランザクションが第2償還可能トランザクションアウトプットを参照するアンロックトランザクションインプットを含むことを要求する、第2アンロックトランザクション制約を含めるステップと、を含む。

【０１５５】

第1スマートコントラクトスクリプト命令セットは、第1ロックスクリプトから複製されたスクリプト命令、及び／又は第1ロックスクリプト内に実装されたスマートコントラクトと異なる新しいスマートコントラクトに対応するスクリプト命令を含むことができる。

【０１５６】

第1制約セットは、第1アンロックトランザクションのアウトプットのロックスクリプトが、第2償還可能トランザクションアウトプットへの参照を含むこと、及び第2スマートコントラクトスクリプト命令セットが、第1償還可能トランザクションのアウトプットへの参照を含むこと、であってよい。

【０１５７】

方法は、第1制約セットの中で、アンロックトランザクションが第1償還可能トランザクションを参照する第1アンロックトランザクションインプットを含むことを要求する第1並列制約と、アンロックトランザクションが第2償還可能トランザクションアウトプットを参照する第2アンロックトランザクションインプットを含むことを要求する第2並列制約と、第1アンロックトランザクションのアウトプットのロックスクリプトが第1スマートコントラクトスクリプト命令セットの中に少なくとも部分的に第1ロックスクリプトにより決定された少なくとも第1スクリプト命令セットを含むことを要求する第3並列制約と、第2アンロックトランザクションのアウトプットのロックスクリプトが少なくとも部分的に第2償還可能トランザクションアウトプットのロックスクリプトにより決定された第2スマートコントラクトスクリプト命令セットを含むことを要求する第4並列制約と、を提供するステップを含んでよい。

【０１５８】

第1制約セットは、アンロックトランザクションが第1償還可能トランザクションを参照する第1アンロックトランザクションインプットを含むことを要求する第1並列制約と、アンロックトランザクションが第2償還可能トランザクションアウトプットを参照する第2アンロックトランザクションインプットを含むことを要求する第2並列制約と、第1アンロックトランザクションのアウトプットのロックスクリプトが第1スマートコントラクトスクリプト命令セットに第1ロックスクリプトから複製された少なくとも第1スクリプト命令セットを含むことを要求する第3並列制約と、第2アンロックトランザクションのアウトプットのロックスクリプトが第2償還可能トランザクションアウトプットのロックスクリプトから複製された第2スマートコントラクトスクリプト命令セットを含むことを要求する第4並列制約と、を含んでよい。

【０１５９】

また、システムであって、プロセッサと、前記プロセッサによる実行の結果として前記システムに上述の方法のいずれかを実行させる実行可能命令を含むメモリと、を含むシステムを提供することが望ましい。

【０１６０】

また、実行可能命令を格納した非一時的コンピュータ可読記憶媒体であって、前記実行可能命令は、コンピュータシステムのプロセッサにより実行された結果として、前記コンピュータシステムに上述の方法のいずれかを少なくとも実行させる、非一時的コンピュータ可読記憶媒体を提供することが望ましい。

【図面の簡単な説明】

【０１６１】

本発明のこれら及び他の態様は、本明細書に記載した実施形態から明らかになり、説明されるであろう。次に、本発明の実施形態を、単なる例示として、添付の図面を参照して説明する:
【図１】様々な実施形態を実装可能なブロックチェーン環境を示す。
【図２】図1のブロックチェーン環境で使用され得るブロックチェーンノードの例を示す。
【図３】図2のブロックチェーンノードによって使用されるブロックチェーン台帳に記憶さ得るトランザクションの例を示す。
【図４】Bitcoinブロックチェーン環境に特有のブロックチェーントランザクションの例を示す。
【図５】署名を生成するためのOP_GENSIGスクリプトを実装するための例示的なスクリプトを示す。
【図６】アンロックトランザクションのインプットに対応する前のトランザクションを注入するためのOP_PREVTXINJECTIONスクリプトを実装するための例示的なスクリプトを示す。
【図７】署名されているインプットに対応する前のトランザクションを注入するためのOP_SELFTXINJECTIONスクリプトを実装するための例示的なスクリプトを示す。
【図８】ロックスクリプトに直列アンロックトランザクションを注入するためのOP_SPENDINGTXINJECTIONスクリプトを実装するための例示的なスクリプトを示す。
【図９】一組のトランザクションの例を示し、一方は、移転されるべき価値を有するトランザクションであり、他方は、その価値のアンロックを表すトランザクションである。
【図１０】複数トランザクションの例を示し、トランザクションは、他のトランザクションからの複数のインプット及び将来のトランザクションによって移転可能な複数のアウトプットを有する。
【図１１】一実施形態による直列化されたトランザクションフィールドのセットから署名を生成する一例を示す。
【図１２】アンロックトランザクションの一部のコピーを含むアンロックスクリプトを含む、アンロックトランザクションの例を示す。
【図１３】アンロックトランザクション及び前のトランザクションを示す。
【図１４】アンロックトランザクションのロックスクリプトに制約を課すロックスクリプトの例のフローチャートである。
【図１５】図14のステップに対応するより具体的なステップのいくつかの例を示す。
【図１６】アンロックスクリプトを検証する例を示す。
【図１７】インプットに制約セットを課すロックスクリプトの例を示す。
【図１８】アンロックトランザクションのインプットに制約を課すロックスクリプトの例のフローチャートである。
【図１９】インプット制約の例を示す。
【図２０】図18に示されるプロセスの文脈における制約の例を示す。
【図２１】スクリプト制約のインターロックを示す。
【図２２】ブロックチェーントランザクションを使用して実装された状態機械の例を示す。
【図２３】状態機械論理がブロックチェーントランザクションにおいてどのように符号化され得るかの例を示す。
【図２４】一実施形態による、信頼性のない決定性状態機械の例を示す。
【図２５】様々な実施形態によるトラストレスな決定性状態機械のためのプロセスの一例を示すフローチャートである。
【図２６】特定の特徴を有する状態遷移マトリクスを使用する状態機械を示す。
【図２７】状態機械を実装するために使用可能なトランザクションの例を示す。
【図２８】ブロックチェーントランザクションを使用したスマートコントラクトの一部の枝分かれの例を示す。

【図２９】枝分かれ処理のための擬似コードシーケンスの例である。
【図３０】ブロックチェーントランザクションを使用したスマートコントラクトからの新しいスマートコントラクトの作成の例を示す。
【図３１】アンロックトランザクションにおいてスマートコントラクトを強制するための疑似コードシーケンスの例である。
【図３２】ブロックチェーントランザクションを使用するスマートコントラクトのための融合操作の例を示す。
【図３３】弱い依存関係を有する融合処理の一例としての擬似コードシーケンスである。
【図３４】強い依存関係を有する融合処理の一例としての擬似コードシーケンスである。
【図３５】ブロックチェーントランザクションを使用するスマートコントラクトに対する並列処理動作の例を示す。
【図３６】並列トランザクションバリアプロセスの例としての擬似コードシーケンスである。
【図３７】スマートコントラクトの部分が、並行パスにおけるブロックチェーントランザクションを使用して、コントラクトバリアに適合してどのように実行され得るかの例を示す。
【発明を実施するための形態】
【０１６２】
まず、図1を参照すると、本開示の一実施形態によるブロックチェーンと関連付けられたブロックチェーンネットワーク100の例を示す。本実施形態では、ブロックチェーンネットワーク100は、ブロックチェーンノードのオペレータ間で全体的又は部分的に合意されたブロックチェーンプロトコルに従った動作を実行するソフトウェア及び／又はハードウェアのインスタンスを各々実行するピアツーピア分散電子装置として実装され得るブロックチェーンノードを含む。いくつかの例では、これらの分散電子装置は、図1のノード102のように、単に「ノード」と呼ばれる。ブロックチェーンプロトコルの一例がBitcoinプロトコルである。
【０１６３】
ノード102は、コンピュータリソースサービスプロバイダの分散システム内の複数の計算装置による、又は任意の適切な電子クライアント装置による、任意の適切な計算装置（例えば、データセンター内のサーバ、デスクトップコンピュータ、ラップトップコンピュータ、タブレットコンピュータ、スマートフォンなどのクライアント計算装置）を備えることができる。ノード102は、データメッセージ又は提案されたトランザクションを表すオブジェクト、例えばトランザクション104を受信するためのインプットを有することができる。ノード102は、それらが保持する情報、例えば、トランザクションの状態の理解について問い合わせることができる。
【０１６４】
図1に示すように、いくつかのノード102は、ノード102のうちの1つ以上の他のノードに通信的に結合されてもよい。どのノード102がどの他のノードと通信することができるかに関して、これらの詳細は、中央で決定される必要はない。ブロックチェーンネットワーク100内のアクティブノードは、1つ以上の他のノード102と通信できる。その結果、メッセージがブロックチェーンプロトコルが転送されるべきであると示すメッセージであると仮定すると、1つのノード102から他のノード102に渡されるメッセージが、ブロックチェーンネットワーク100全体（又は、その重要な部分）に伝搬することができる。そのようなメッセージの1つは、ノード102Aの1つによる提案されたトランザクションの公表であり、これは、パス106のようなパスに沿って伝搬する。そのようなメッセージは、ブロックチェーンに含めるために提案された新しいブロックの公表であり得る。
【０１６５】
図2は、図1のブロックチェーンネットワーク内のノードとしてブロックチェーン環境で使用され得るブロックチェーンノード202の一例を示す。図示のように、ブロックチェーンノード202は、プロセッサ204、プログラムメモリ206、ブロックチェーンルール及びプ

ロトコル詳細のための記憶装置208、ピアのリスト210、アプリケーション変数のための記憶装置212、通信インタフェース214、及びブロックチェーン台帳のための記憶装置216を含む。ブロックチェーン台帳216は、トランザクション230（1）～（4）を有するブロック222のような、トランザクションに対応するデータを含むブロック221、222、223のような、ブロックチェーン内のブロックに対応するデータを含む。ブロック223を除いて、ブロックは、後続のブロックがブロックが不変となったときに該ブロックに基づいて計算される値に依存し、不変ブロックに対するいかなる修正も、他のブロックチェーンノードによって無効ブロックとして容易に認識され得るという点で、暗号学的に不変であり得る。ブロック223は、変更可能である可能性があり、ブロックチェーンに未だコミットされていないトランザクションの記憶を表す、開いた箱として示されている。

【０１６６】

ブロックチェーンノード202は、有線又は無線通信の1つ以上として実装することができる通信インタフェース214を介して通信する。ブロックチェーン台帳216は、ブロックチェーンネットワークのブロックチェーン台帳の完全なコピー又はその一部であってもよい。ブロックチェーンノードの中には、未移転トランザクションのみを維持するものもあれば、台帳全体を維持するものもある。このようにして、各ノードが独自のコピーを持つことができるので、台帳は分散型台帳である。ノードは、好ましくは、プロトコルの規則に従って自身のコピーのみを変更し、それらの規則に完全に従うすべてのノードについては、彼らのコピーは、時には、ブロック及びトランザクションのためのある伝搬時間を除き、他のノードと同じであるべきである。ブロックチェーンノードは、受信したブロックとそれらのブロック内のトランザクションを検証する機能を含む必要がある。ブロックチェーンプロトコルの規則では、ブロックチェーンノードは、トランザクションのブロックが無効であると判断した場合、ブロック又はトランザクションを他のノードに伝搬しない。この規則では、有効であり、有効であることが確認されたブロックとトランザクションは、ブロックチェーンネットワークを伝搬してよいが、無効なブロックとトランザクションは伝搬しない。

【０１６７】

ブロックチェーンネットワーク内のノードのいくつかは、新しいブロックをブロックチェーンにコミットできるように、トランザクションを収集し、トランザクションのブロックを作成する操作を実行するマイナーノードであってよい。マイナーノードは、（不正なノードが容易に不適切なブロックを生成することを避けるために）複雑であり、（不正なノードが事前に複雑な操作を実行できないように）含まれるトランザクションのデータに依存する何らかの操作を実行することが期待されてよい。これらのタスクのパフォーマンスは、他のノードによって容易に検証される。これらの他のノードは、マイナーノードの作業を検証し、検証するとブロックチェーンの彼らのコピーにブロックを受け入れ、それを伝搬する。これにより、新しいブロックがブロックチェーンの分散台帳にコミットされる。いくつかの例では、ブロックはトランザクションのグループであり、前のブロックのタイムスタンプと「フィンガープリント」（ハッシュなど）でマークされることが多い。このようにして、各ブロックは前のブロックにリンクされ、ブロックチェーンのブロックをリンクする「チェーン」が生成される。実施形態において、有効なブロックは、ブロックチェーンネットワーク内のノードのコンセンサスによってブロックチェーンに追加されてもよい。また、いくつかの例では、ブロックチェーンは、検証されたブロックのリストを含む。

【０１６８】

一実施形態では、ノードの少なくともいくつかは、本開示に記載されるように、トランザクションを検証する検証ノードとして動作する。いくつかの例では、ブロックチェーンは、デジタル資産の形で価値を証明し、ブロックチェーンのトランザクションは、（ブロックチェーンを開始する）ジェネシストランザクションから、あるいは、マイナーノードオペレータが参加する理由として、新たに「マイニング」されたデジタル資産を生成する分配トランザクションから、後続のトランザクションへのチェーンを提供する。チェーン

の1つの部分の終わりは、未移転トランザクション、又は、その一部であり、ブロックチェーン上の各トランザクションは、1つ以上のアウトプットを指定する。これらのアウトプットの一部は、移転されてよく、また、移転されていないものもある。ここで、アウトプットは、そのアウトプットの値の仕様と、チェーンの該当部分を延ばす有効なトランザクション（すなわち、指定された要件を少なくとも満たすもの）を挿入することによって該アウトプットを「アンロック」するために必要とされる要件とを含む。本明細書で使用されるように、未使用トランザクションアウトプットは、UTXOと称される場合がある。UTXOをアンロックすることは、UTXOを使用することとしても知られている。

【０１６９】

UTXOをアンロックするために必要な要件は、UTXOのロックスクリプト内で指定することができる。有効なアンロックトランザクションは、（ｉ）トランザクションのインプットとして1つ以上の前のUTXOを指定し、指定はUTXOへのポインタによるものであってよく、（ｉｉ）それぞれのUTXOに対するロックスクリプトの要件を満たし、（ｉｉｉ）これらのインプットによってポイントされたすべてのUTXOの値の合計がトランザクションのすべてのアウトプットの値の合計以下になるようなインプットを有することができる。ロックスクリプトの要件の充足を表すデータは、アンロックスクリプトと呼ばれる。検証ノードは、アンロックスクリプトに続いてロックスクリプトを処理することができ、その処理のアウトプットは、無効な結果又はアンロックトランザクションの検証のいずれかである。

【０１７０】

あるノードがアンロックスクリプトとロックスクリプトを実行し、その結果が他のノードでの結果となり得る検証である場合、そのプロセスはステートレスプロセスである。したがって、まだ移転されていないブロックチェーン上の前のトランザクションのアウトプットのロックスクリプトを満たすアンロックトランザクションは、伝搬し、最終的にブロックチェーンにコミットされてよい。コミットされるアンロックトランザクションを作成したノードは、新しいトランザクションのアウトプットの値をアンロックするために必要な要件を、自分自身のロックスクリプトで指定し得る。このプロセスは、スクリプトがスクリプトの外側の参照を参照する必要がなく、いくつかの小さな例外を除いて、それらのスクリプトの外側の変数や他の状態に依存しないことから、ステートレスとみなすことができる。その結果、アンロックスクリプトとロックスクリプトを分離して評価することができる。

【０１７１】

他のバリエーションでは、ブロックチェーンシステムは、スクリプト評価プロセスに外部データを含めることを可能にする機能を含んでもよい。このような場合、外部データは、アンロックスクリプト及びロックスクリプトが外部データを使用して評価されるときに、評価が評価を実行するノード間で一貫しているような方法で制御及び／又はセキュアにすることができる。

【０１７２】

図3は、図2のブロックチェーンノードによって使用されるブロックチェーン台帳に記憶されるトランザクション300の例を示す。同様の機能を持つ他のバリエーションも可能である。トランザクションのデータ要素又はフィールドは、図3に示されるようなものであってもよく、本開示に記載されている以外の追加のフィールドを含んでもよい。示されるように、トランザクション300のブロックチェーンプロトコルバージョンを示す値を有するブロックチェーンバージョン（Blockchain Version）フィールド302がある。#vinフィールド304は、トランザクション300に存在するトランザクションインプット（以下に説明する）の数を示す。他のフィールドが存在し、図示されていないが、各トランザクションインプット（ここでは例示としてVin[y]310として示されている）に対して、前のトランザクションのトランザクションID(TxID)311、前のトランザクション（トランザクションインプット310に適合するトランザクションアウトプットを提供するトランザクション）のアウトプットのうちの1つへのポインタ312を含むフィールドセットが存在してもよく、ここで、TxID311及びポインタ312は共に、前のトランザクションのアウトプットを参照す

るポインタ313を形成する。本明細書中で使用されるように、現在（current又はpresent）のトランザクションに関する文脈で使用される場合、「前のトランザクション」という用語は、現在のトランザクションによって参照される（及び「移転される」）トランザクションアウトプットを有する特定の前のトランザクションを指すことがある。例において、現在のトランザクションは「アンロックトランザクション」と呼ばれることがある。

【０１７３】

いくつかのブロックチェーン実装では、固有のTxID値を割り当てるための集中化されたメカニズムがなく、代わりに、トランザクション自体の内容のハッシュを生成することによって、トランザクションのための固有のTxIDを生成するための分散化されたメカニズムがある。有効なトランザクションは、他の有効なトランザクションと全く同じコンテンツを持つことはできないので、各有効なトランザクションは（ハッシュ衝突の天文学的に低い確率を除いて）TxIDに対して固有のハッシュを有し得る。しかしながら、ここでは、各トランザクションが固有のトランザクションIDを持つと仮定する。ハッシュの性質のために、ひとたびTxIDがトランザクションのコンテンツから生成されると、そのコンテンツのいずれも変更することができず、TxIDがそのトランザクションに対して有効のままである。

【０１７４】

図3に示すように、トランザクションインプットVin[y]310のためのフィールドセットは、次に続くスクリプトの長さを示すUnlocking_Script_Lengthフィールド314と、ポインタ313によって示されるトランザクションアウトプットの対応するロックスクリプトを「アンロック」するvin[y]310のためのアンロックスクリプトを含むUnlocking_Scriptフィールド315と、トランザクション300を制約するために使用され得るSequence#フィールド316とを含む。

【０１７５】

図3は、1つのトランザクションインプット及び1つのトランザクションアウトプットのみを明示的に示しているが、各々、1つ以上が可能である。トランザクションインプットに続いて、トランザクション300内に存在するトランザクションアウトプットの数（以下にも説明する）を示すことができる#voutフィールド320が存在してもよい。各トランザクションアウトプット（ここでは、例示的なVout[x]330として示される）に対して、このトランザクションアウトプットVout[x]330によって提供されるトランザクション値を示すOutput Value（アウトプット値）フィールド332、続くスクリプトの長さを示すLocking_Script_Lengthフィールド334、及びこのトランザクションアウトプットVout[x]330のためのロックスクリプトを含むLocking_Scriptフィールド336を含む一連のフィールドが存在してもよい。このように、このトランザクションアウトプットのトランザクション値は、アンロックスクリプトとロックスクリプトを使用して検証を行うときに、ブロックチェーンノードがTRUEとして検証できるアンロックスクリプトを持つトランザクションインプットを持つアンロックトランザクションを作成できる任意の者により「移転」することができる。トランザクションアウトプットフィールドの後に、トランザクション300が指定された将来の時間の前又は指定された将来のブロックの前にアクティブでないように制約するLock Timeフィールド338のような他のフィールドが続いてよい。アンロックトランザクションの各トランザクションインプットが前のトランザクションのアウトプットの対応するトランザクションアウトプットを指し、前のトランザクションのアウトプットがトランザクション値を含む場合、トランザクションインプットは、そのトランザクション値を示すフィールドを含む必要はない。

【０１７６】

図4は、Bitcoinブロックチェーン環境に特有のブロックチェーントランザクション400の例を示す。この例では、ブロックチェーントランザクション400は、示されたデータ要素又はフィールドを含み、本開示において示された、又は説明されたフィールド以外の追加フィールドを含み得る。トランザクション400は、トランザクション400のバージョンを示す値を有するnVersionフィールド402を含む。#vinフィールド404は、トランザクション

400内に存在するトランザクションインプットの数を示すことができる。トランザクションインプットVin[y]410のような各トランザクションインプットに対して、前のトランザクションを参照するhash（ハッシュ）411、前のトランザクションの特定のアウトプットを示すポインタn412、後に続くアンロックスクリプトの長さを示すscriptSigLenフィールド414、そのトランザクションインプットVin[y]410のためのアンロックスクリプトを含むscriptSigフィールド415、及びトランザクション400を制限するために使用され得るnSequenceフィールド416を含む一連のフィールドが存在し得る。

【０１７７】

トランザクションインプットに続いて、トランザクションアウトプットがトランザクション400内に存在する数を示す#voutフィールド420が存在し、各トランザクションアウトプット（ここでは、例示的なVout[x]430として示されている）に対して、このトランザクションアウトプットVout[x]430によって提供されるトランザクション値を示すnValueフィールド432、続くロックスクリプトの長さを示すscriptPubKeyLenフィールド434、及びこのトランザクションアウトプットVout[x]430のためのロックスクリプトを含むscriptPubKeyフィールド436を含む一連のフィールドが存在してもよい。トランザクション440はまた、トランザクション400が指定された将来の時間の前又は指定された将来のブロックの前にアクティブでないように制約し得るnLockTimeフィールド438と共に示されてもよい。

【０１７８】

OPコード

多くのBitcoinブロックチェーンノードで使用されるもののようなスクリプト言語の例では、ロックスクリプトやアンロックスクリプトのようなスクリプト命令としてスクリプト内に現れるオペレーションコードのセットがあり得る。本開示では、種々のスクリプトOPコード及びキーワードが、種々の操作を実行するために参照される。しかしながら、他のブロックチェーン技術は異なる命令セットを実装することができると考えられ、従って、本開示に記載されたオペコードは、オペコードによって実行される操作を例示するものであると考えるべきである。

【０１７９】

本開示の特定の実施形態は、記述された命令セットを実施するためのスクリプトシステム又は他のシステムが、単一のスクリプトにおいて200を超える命令（例えば、200を超えるOPコード）を許容するという仮定の下で動作する。同様に、本開示の特定の実施形態は、さらに、本開示で言及されたオペコードによって提供される機能が、オペコード、スクリプト/命令セットを実行するシステムに存在し、かつ可能であると仮定する。本開示で言及したオペレーションコードの例は、以下を含む:

OP_ADD　スタックから上位2つのアイテムをポップし、それらを加算して結果をスタックにプッシュする。

【０１８０】

OP_BIGMOD　スタックから上位2つのアイテムをポップし、2番目のアイテムを1番目のアイテムで割って残余をスタックにプッシュする。

【０１８１】

OP_BIGMODADD　スタックから上位3つのアイテムをポップし、1番目と2番目のアイテムをモジュロ加算し、3番目のアイテムをモジュロし、結果をスタックにプッシュする。

【０１８２】

OP_BIGMODINVERSE　スタックから上位2つのアイテムをポップし、1番目のアイテムを法とする2番目のアイテムのモジュロ負指数演算を実行して、結果をスタックにプッシュする。

【０１８３】

OP_BIGMODMUL　スタックから上位3つのアイテムをポップし、1番目と2番目のアイテムをモジュロ乗算し、3番目のアイテムをモジュロし、結果をスタックにプッシュする。

【０１８４】

OP_CAT　スタックから上位2つのアイテムをポップし、2つのアイテムを連結し、結果を

スタックにプッシュする。
【０１８５】
OP_CHECKSIG　スタックから2つのアイテムをポップし、1番目のアイテムを公開鍵として、2番目のアイテムを署名として使用し、ポップした署名が有効な署名であるかどうかをチェックし、有効な場合は、スタックに1（TRUE）をプッシュし、その他の場合は、スタックに0（FALSE）をプッシュする。
【０１８６】
OP_CHECKSIGVERIFY　OP_CHECKSIGと同じ機能を持つが、OP_VERIFYが後に実行される。
【０１８７】
OP_DIV　スタックから2つのアイテムをポップし、2番目のアイテムを1番目のアイテムで分割し、結果をスタックにプッシュする。
【０１８８】
OP_DUP　スタック上の1番上のアイテムのコピーをスタックにプッシュすることで、一番上のアイテムを複製する。
【０１８９】
OP_ELSE　先行するOP_IF又はOP_NOTIF又はOP_ELSEが実行されなかった場合、これらのステートメントが実行される。先行するOP_IF又はOP_NOTIF又はOP_ELSEが実行された場合、直前のOP_IF,OP_NOTIF,OP_ELSEが実行された場合、これらのステートメントは実行されない。
【０１９０】
OP_ENDIF　if/elseブロックを終了する。
【０１９１】
OP_EQUAL　スタックの最初の2つのアイテムを読み込み、2つのアイテムが完全に等しい場合にスタックに1を、その他の場合に0をプッシュする。
【０１９２】
OP_EQUALVERIFY　OP_EQUALと同じであるが、後にOP_VERIFYを実行する。
【０１９３】
OP_FROMALTSTACK　インプットをメインスタックの一番上に置き、代替スタックからそれを削除する。
【０１９４】
OP_HASH256　最初はSHA－256、次にRIPEMD－160により、インプットが2回ハッシュされる。
【０１９５】
OP_IF　一番上のスタック値がFalseでない場合、ステートメントが実行され、一番上のスタック値が削除される。
【０１９６】
OP_MUL　スタックの上位2つのアイテムを乗算する。
【０１９７】
OP_NOTIF　一番上のスタック値がFalseの場合、ステートメントが実行され、一番上のスタック値が削除される。
【０１９８】
OP_OVER　スタックの2番目のアイテムをコピーし、スタックの一番上にプッシュする。
【０１９９】
OP_ROLL　スタックの深さがn個のアイテムが一番上に移動される。
【０２００】
OP_SUBSTR　文字列のセクションを返す。
【０２０１】
OP_SWAP　スタックの上位2つのアイテムを交換する。
【０２０２】
OP_TOALTSTACK　インプットを代替スタックの一番上に置き、メインスタックから削除

する。

【０２０３】

OP_VERIFY　一番上のスタック値が真でない場合、トランザクションを無効としてマークする。

【０２０４】

上記のOPコードのいくつかは、ブロックチェーンノードによってサポートされる基本OPコードであるかも知れないが、他のものは、説明を明確にするために、ここではOPコードとして参照されるが、典型的には、複数の基本OPコードで構成される小さなスクリプトとして実装される。例えば、OP_BIGMODはOPコードとして参照されるが、スタックの上位2つのアイテムを除算して余りを返す機能を実行するために、一連のOPコードを使って実装されるかも知れない。OP_BIGMODADD,OP_BIGMODINVERSE,OP_BIGMODMULは、同様のOPコードのシーケンスで簡単に表現され得る。

【０２０５】

OP_GENSIG　本開示においては、OP_GENSIGの言及は、ここに記述されたOP_GENSIGスクリプトに対応する操作の略称として見なされるべきである。トランザクションの署名を生成するOP_GENSIGスクリプトは、また、スタックにプッシュされるか使用され得る。OP_GENSIGスクリプトは、始めに、メイン（後入、先出）スタックにプッシュされたインプットが、順に<SIGHASH　Type>（トランザクションのどのフィールドがハッシュで使用されるべきかを示す）、メッセージ値<m>（トランザクションの使用されたトランザクションフィールドの直列（serialized）セットのダブルSHA256）、秘密鍵<a>（メッセージ値<m>のハッシュに使用される）、及び数字<k>（秘密鍵をマスク/保護するために使用される乱数又は擬似乱数；「マスク番号」（mask　number）であると仮定する。

【０２０６】

図5は、OP_GENSIGスクリプトのインプット及びアウトプットを示す。OP_GENSIGスクリプトでは、最初の行は数字<k>を代替スタックにコピーし、次に楕円曲線生成器<PubK　G>に対して<k>を乗算し、メインスタックの一番上に楕円曲線点Kを生成する。スクリプトの2行目は、Kモジュロnのx座標からrを計算し、rのコピーを代替スタックにプッシュする。スクリプトの3行目は、$s=k^{-1}(m+r\times a) \bmod n$を決定する。最後に、スクリプトの4行目はrとsをDER形式で符号化し、<SIGHASH　Type>に連結する。

【０２０７】

署名ハッシュのタイプは、バイト符号化値であるSIGHASH　Typeで示すことができる。SIGHASH　Type値は、直列化（例えば正規化）及びハッシュ化される前にトランザクションから抽出されるフィールドの集合を指定する。いくつかの例では、SIGHASH　Type値はSIGHASH_ALL、SIGHASH_NONE、SIGHASH_SINGLE、SIGHASH_ANYONECANPAY、又はこれらの2つ以上の組み合わせであることができる。一実施形態では、タイプSIGHASH_ALLは、インプットスクリプトを除き、トランザクション内のすべてのフィールドが直列化に含まれる（従って、ハッシュ化され、署名される）ことを示す。一実施形態では、タイプSIGHASH_NONEは、他者がトランザクションを更新することができるように、アウトプットが署名される必要がないことを示す。一実施形態では、タイプSIGHASH_SINGLEは、インプットは署名されるが、シーケンス番号は空白であることを示す。したがって、他者は、同じハッシュを使用するトランザクションの新しいバージョンを作成することができるが（ハッシュされるフィールドは変更されないと仮定する）、署名されるアウトプットはインプットと同じ位置にあるアウトプットのみである。

【０２０８】

一実施形態では、タイプSIGHASH_ANYONECANPAYは、他のタイプと組み合わされ、SIGHASH_ANYONECANPAYを含むインプットが署名されているが、他のインプットが署名される必要はないことを示す。SIGHASH　Type値は、SIGHASH　Typeを示す数値によって表すことができる。例えば、いくつかの実装では、SIGHASH_ALLは1の値を持つバイト（例えば、バイナリ"00000001"）で表され、SIGHASH_NONEは2の値を持つバイト（例えば、バイナリ"00000010"）で表され、SIGHASH_SINGLEは3の値を持つバイト（例えば、バイナリ"000000011"）

で表され、SIGHASH_ANYONECANPAYは80の値を持つバイト（例えば、バイナリ"01010000"）で表される。いくつかの実装では、SIGHASH Type値の組み合わせは、それぞれのバイト値のバイナリ加算によって実行される。いくつかの例では、SIGHASH Typeによって決定されたトランザクションフィールドの集合は、直列化される対応するトランザクションのSIGHASH Type値によって決定されるサブセットを表す。例えば、SIGHASH_ANYONECANPAYのSIGHASH Typeは、1つのトランザクションインプットだけが署名に含まれる。

【０２０９】

アンロックスクリプトにSIGHASHを含めることを要求することによって、ロックスクリプトは、アンロックスクリプトによって提供された直列化されたアンロックトランザクションフィールドのセットが実際にSIGHASHのソースであることを検証することができ、次に、OPコードOP_CHECKSIGを使用して、アンロックスクリプトによって提供されたSIGHASH Type値がアンロックトランザクションの署名ハッシュと一致することを検証することができる。ロックスクリプトは、トランザクション検証器によって実行されると、そのときトランザクション検証器はOP_CHECKSIG操作を実行し、実際のアンロックトランザクションの署名をスタック上に置くことができるため、これを実行することができる。

【０２１０】

OP_DERENCODE 本開示においては、OP_DERENCODEの言及は、スタックから上位2つのアイテムをポップし、それらをDER形式で符号化し、結果をスタックにプッシュすることに対応する操作の略称として見なされるべきである。

【０２１１】

OP_ECPMULT 本開示においては、OP_ECPMULTの言及は、スタックから2つのアイテムをポップし、これら2つのアイテムの楕円曲線点乗算（楕円曲線スカラ乗算とも呼ばれる）を実行し、結果をスタックにプッシュすることに対応する操作のための略称として見なされるべきである。

【０２１２】

OP_ECPX 本開示において、OP_ECPXの言及は、スタックから2つのアイテムをポップし、1番目のアイテムをモジュラスnとして、2番目のアイテムを楕円曲線点Kとして使用し、Kのx座標modulo nを計算し、結果をスタックにプッシュすることに対応する操作の略称として見なされるべきである。

【０２１３】

OP_PREVTXINJECTION 本開示において、OP_PREVTXINJECTIONの言及は、アンロックトランザクションのインプットXに対応する前のトランザクションの注入に対応する操作のための略称として見なされるべきである。図6は、OP_PREVTXINJECTIONスクリプトのインプット及びアウトプットを示す。

【０２１４】

OP_SELFTXINJECTION 本開示において、OP_SELFTXINJECTIONの言及は、署名されているインプットに対応する前のトランザクションの注入に対応する操作のための略称として見なされるべきである。図7は、OP_SELFTXINJECTIONスクリプトのインプット及びアウトプットを示す。

【０２１５】

OP_SPENDINGTXINJECTION 本開示においては、OP_SPENDINGTXINJECTIONの言及は、ロックスクリプトに、直列化されたアンロックトランザクションを注入することに対応する操作のための略称とみなされるべきである。SIGHASH Type値とSIGHASH Typeに従って決定されたアンロックトランザクションフィールドセットを提供するエンティティは、トランザクションアウトプットをアンロックすることができる。これは便利な機能である。図8は、OP_SPENDINGTXINJECTIONスクリプトのインプット及びアウトプットを示す。

【０２１６】

OP_EXTRACTTXID 本開示においては、OP_EXTRACTTXIDの言及は、トランザクションID及びトランザクションID位置をインプットとして取得し、抽出したトランザクションIDをアウトプットすることに対応する操作のための略称として見なされるべきである。OP_EXTRA

CTTXIDへのインプットは、直列化された（seriailzed）トランザクション及びインプットインデックスを示す値Xであり、アウトプットは、直列化されたトランザクションにおけるそのインデックスにあるトランザクションインプットに関連付けられたトランザクション識別子である。このスクリプトは、トランザクションIDを抽出するために、直列化されたトランザクションを解析する必要があってよい。なぜなら、直列化されたトランザクションは、直列化されたトランザクション内の他のフィールドによって示される長さの分散を持つ可変長のフィールドを含んでいるからである。

【０２１７】

図9は、一対のトランザクションの例を示す。トランザクションを検証するブロックチェーンノード902は、トランザクション検証器904を有してもよい。トランザクション検証器904は、そのインプットが有効であるかどうかに応じて、TRUE又はFALSEの結果906をアウトプットするステートレススタックベースのトランザクション検証器であってもよい。他の変形では、トランザクション検証器は、必ずしもステートレスではなく、かつ/又は外部の信頼できるデータ又は状態を参照することができる。ここでの教示は、スタックベースのトランザクション検証器を使用しないブロックチェーンシステムにも適用することができる。

【０２１８】

トランザクション検証器904へのインプットは、前のトランザクション（Previous Tx）912のアウトプット（移転されるトランザクションアウトプット）からのロックスクリプト910と、アンロックトランザクション（Unlocking Tx）922のインプット（アンロックを行うトランザクションインプット）のアンロックスクリプト920とであってよい。示されていないが、各トランザクションは他のフィールドを含み得る。より一般的な場合では、各トランザクションは、複数のインプット及び／又は複数のアウトプットを有することができ、その場合、ロックスクリプト910は、前のトランザクション912の複数のトランザクションアウトプットのうちの1つに対するロックスクリプトであってもよく、アンロックスクリプト920は、アンロックトランザクション922の複数のトランザクションインプットのうちの1つに対するアンロックスクリプトであってもよい。

【０２１９】

ステートレスなスタックベースのトランザクション検証器では、ロックスクリプト910とアンロックスクリプト920は、いくつかの少数の例外を除いて、単独で処理され、これらのスクリプトの外部の変数又は他の状態に依存しない。結果として、ロックスクリプト910は、ロックスクリプト910自体の外部の前のトランザクションのフィールドを参照しなくてよい。ロックスクリプト910はまた、前のトランザクション912、したがってロックスクリプト910が作成され、不変とされたときには、ロックトランザクション922が存在しなかったので、アンロックトランザクション922のフィールドを参照することはできない。

【０２２０】

また、アンロックスクリプト920は、ステートレスであるために、アンロックトランザクション922のフィールドに対して直接操作することもできない。ステートレス品質は、適切にプログラムされたブロックチェーンノードが、同じロックスクリプト910及び同じアンロックスクリプト920の場合に、同じ結果906に到達することを保証するという点で有用であり得る。これに対する1つの例外は、スクリプトが、トランザクション全体から生成された署名に対して、1つのスクリプトの署名をチェックすることができることである。一実施形態において、スクリプトは、この目的のためにOP_CHECKSIGを使用することができる。

【０２２１】

スタックベースの操作では、トランザクション検証器は、スクリプト内の各OPコードを順に処理することによって、アンロックスクリプトを実行する。いくつかのOPコードは、結果として処理され、いくつかは結果としてデータがスタックにプッシュされるか、又はスタックからポップされ、又はいくつかの組み合わせになる。アンロックスクリプトの実行が完了すると、スタックにデータが残っている可能性がある。アンロックスクリプトの

実行にスクリプトエラー又は実行を終了するOPコードがなければ、トランザクション検証器はロックスクリプトを実行し、スタックに存在するデータを使用することができる。ロックスクリプトとアンロックスクリプトが同じOPコードセットを使用して書かれ得る場合、これは実質的に、アンロックスクリプトと後続のロックスクリプトとの連結を含む単一のスクリプトを実行することに類似している。しかし、アンロックスクリプトのOPコードの実行終了時及びロックスクリプトのOPコードの実行前に、スクリプトエラーのチェックを行うと、不正なアンロックスクリプトによって引き起こされる問題を防ぐことができる。

【０２２２】

スクリプトの実行が完了すると、スタックの一番上に残る結果が検証の結果となる。TRUEがスタックの一番上にある場合、これはアンロックスクリプトが実際にロックスクリプトをアンロックすることの検証と考えられ得る。この点に関して、アンロックスクリプトは、ロックスクリプトの要件のすべてを満たす。

【０２２３】

図10は、複数トランザクション及び複数のインプット及びアウトプットの例を示す。図示されているように、前のトランザクション（Previous Tx1）1002は、2つのトランザクションアウトプット、すなわち、ロックスクリプト1006を有するVout[0]1004及びロックスクリプト1016を有するVout[1]1014、及び、図示されていない他のフィールドの中でも、前のトランザクション1002へのトランザクションインプットのような、それらの対応するトランザクションアウトプット値を有する。アンロックトランザクション（Unlocking Tx1）1020は、2つのトランザクションインプット、Vin[0]1024（アンロックスクリプト1026及び対応する前のトランザクションのアウトプットへのポインタ1028を有する）及びVin[1]1034（アンロックスクリプト1036及び対応する前のトランザクションのアウトプットへのポインタ1038を有する）で示される。この例では、Vin[0]に対する対応する前のトランザクションのアウトプットは、前のトランザクション（Previous Tx1）1002のVout[0]であり、一方、Vin[1]に対する対応する前のトランザクションのアウトプットは、別のトランザクション（Previous Tx2）（図示せず）の何らかのアウトプットである。

【０２２４】

アンロックトランザクション1020を検証するために、トランザクション検証器1044を備えたブロックチェーンノード1042は、そのすべてのインプットが有効であるかどうかに応じて、TRUE又はFALSEの結果1046をアウトプットする。この場合、複数のインプットがトランザクション1020のアンロックのために存在してもよいので、トランザクションインプットのアンロックスクリプトと、インプットのポインタが指す対応するトランザクションアウトプットのロックスクリプトとを処理することによって、各インプットが検証される。

【０２２５】

複数のアウトプットと複数のインプットを持つことができるため、1つのトランザクションの価値は、複数の他のトランザクションによって移転でき、新しいトランザクションで償還可能な価値は、1つより多くの前のトランザクションから来る可能性がある。上記で説明したように、ロックスクリプトは、署名及び他の暗号技術を介して、アンロックスクリプト内にあるべきものに制約を課すことができる。

【０２２６】

そのような制約の1つは、アンロックスクリプトが何らかの任意のデータを含まなければならないことであってよい。制約の中には、「弱い制約」やより一般的な制約、例えば、アンロックスクリプトが4バイトの長さを持つデータを含んでいなければならないという制約、又は、バイトを特定の値に制約せずに特定数のバイトを含む制約など、があってよい。データがハッシュされるときに、データが特定のハッシュ値を生成しなければならないということは、より強力な、又はより特定の制約であり得る。この後者の制約は、データが特定の長さであり、該データに固有の値であることを典型的に必要とする。なぜなら、同じハッシュを生じる異なる値を誰かが決定することはほとんど不可能であるからで

ある。制約は、アンロックスクリプトがそのデータをスタックにプッシュするか、又はそのデータのハッシュをスタックにプッシュし、ロックスクリプトが必要なハッシュをスタックにプッシュし、次にOP_EQUALを実行し、次にOP_VERIFY　OPコードを実行することによって、スクリプトに実装され得る。それらのOPコードは、もしそれらの2つのプッシュされたアイテムが一致した場合にTureを返し、それらが一致しなかった場合にFalseを返す。より一般的なケースでは、特定のデータが存在することを要求する制約や、特定の特性を持つデータが存在することを要求する制約を課すことができる。

【０２２７】

弱い制約の例は、アンロックスクリプトが、必ずしもそれらのフィールドが特定の値であることを必要とせずに、直列化されたアンロックトランザクションフィールドのセットを構成する可能性のある特定のデータを含まなければならないことである。このようにして、ロックスクリプトは、アンロックスクリプトが来た場所からのアンロックトランザクションのフィールドを含むように、ロックスクリプトのアウトプットをアンロックするアンロックスクリプトのスクリプトを制約することができる。これにより、ロックスクリプトは、アンロックスクリプトがスタックに残すアンロックトランザクションのフィールドに対して、効果的に操作を実行することができる。弱い制約の別の例は、アンロックスクリプトが、指定された場所から得られる可能性のある特定のデータを含まなければならないが、データの値に制約がない場合である。これにより、データの値がロックスクリプトが不変になった後まで決定されないとしても、いくつかの弱い制約を課すことができる。

【０２２８】

制約の操作と制約のデータには、「スマートコントラクト」の要素が含まれる場合がある。スマートコントラクトとは、誰が何をすることができるのか、彼らがいつすることができるのか、そして誰が、いつ、何が支払われるのか、といった契約の条件を定義するものである。条件が、アンロックスクリプト又はアンロックトランザクションの他の部に対する、弱い又は強い制約として表現され得る場合、スマートコントラクトの条件は、ブロックチェーンプロトコルによって施行され得る。したがって、上述の例のように、例えば、Bobだけがロックされたトランザクションアウトプットの値をアンロックできる場合（Bobだけが作成できるアンロックスクリプトだけがそのトランザクションアウトプットをアンロックするため）、一方のパーティによって許可された場合にのみ、一定の時間が経過した後で、他方のトランザクションアウトプットが使用された後にのみ、トランザクションアウトプットが使用できる。

【０２２９】

図8は、OP_SPENDINGTXINJECTION、すなわち、直列化されたアンロックトランザクションフィールドのセットの注入を引き起こすための制約を符号化するスクリプトのためのOPコードの略称を示す。ロックスクリプトは、OPコードOP_GENSIG（結果としてBitcoin署名を生成する）と、後続のOPコードOP_CHECKSIG（結果として、署名がアンロックトランザクションの有効な署名としてチェックされる）と、を含み得る。OP_GENSIGが有効な署名を生成するためには、特定インプット要素として、SIGHASHタイプ要素、直列化されたアンロックトランザクションフィールドのセット（SIGHASHタイプ要素に対応する特定のセット）、秘密鍵、マスク番号（例えば、秘密鍵をマスク/保護するために使用される乱数又は擬似乱数）を持たなければならない。ロックスクリプトにハッシュOPコード、秘密鍵、及び乱数を含めることにより、これは、SIGHASHタイプ及び直列化されたアンロックトランザクションフィールドのセットを提供するように、アンロックスクリプトを効果的に制約する。

【０２３０】

図11は、直列化されたトランザクションフィールドのセットから署名を生成するための手順を示す。これは、トランザクションを作成しているブロックチェーンノードや、トランザクションの他のジェネレータによって行われる場合がある。図11に示すように、直列化されたトランザクション1110（すなわち、特定のフォーマットの一連のデータオブジェクトで表されるトランザクション）は、トランザクションのフィールド値のセットを含む

。直列化されたトランザクション1110の署名者は、SIGHASH Type1112を選択し、署名者の秘密鍵を提供する。ブロックチェーンクライアント/ノードは、SIGHASH Type1112から、直列化されたトランザクション1110のどのフィールド又は修正されたフィールドが使用されるべきかを決定することができる。フィールドの一部は、ハッシュが生成される前に変更されてよい（ゼロに設定することを除く）。図11の例では、SIGHASH TypeはSIGHASH_NONE|SIGHASH_ANYONECANPAYに等しく設定され、従って、選択されるフィールドは、修正された直列化トランザクション1114によって示されるフィールドである。

【０２３１】

ブロックチェーンクライアント/ノードは、修正された直列化トランザクション1114のハッシュを生成し（例えば、ダブルSHA－256ハッシュを実行する）、その結果、メッセージm1116をもたらす。直列化トランザクション1110を処理するブロックチェーンクライアント/ノードは、秘密鍵をマスク/保護するために、典型的には乱数又は擬似乱数である数k1118を選択する。kという数字は、本開示の時には「マスク番号」として参照される。次に、ブロックチェーンクライアント/ノードは、SIGHASH Type1112、メッセージm1116、署名者の秘密鍵1122、及び数字k1118からの、一組の署名インプット1120から、署名1124（この例ではDER符号化EC署名）を生成する。次に、署名1124及びSIGHASH Type1112をアンロックスクリプトで使用することができる。

【０２３２】

図12は、アンロックトランザクションのコピー（直列化されたアンロックトランザクションフィールドのセットなど）を含む、アンロックスクリプト1203を含む、アンロックトランザクション1200の例を示す。図12に示すように、トランザクションインプットの1つは、Vin[z]である。インプットVin[z]は、前のトランザクションTxIDであるハッシュと、前のトランザクションのアウトプット数nを持つ。これらを合わせると、Vin[z]がアンロックすべき特定の前のトランザクションの特定のアウトプットへのポインタ1202を形成する。トランザクションインプットVin[z]は、アンロックスクリプト1203を有し、これはまた、いくつかのコンポーネント又はサブスクリプト1206、1208、1210、1212を有する。アンロックスクリプト1203は、Previous TxZの直列化されたバージョン1206を含む。同じ方法で直列化されたPrevious TxZによって移転されたアウトプットを有するトランザクションであるPrevious TxZ'の直列化されたバージョン1208も示されている。示されていないが、追加の前のトランザクションを含めることができる。また、アンロックスクリプト1203は、アンロックトランザクション1200のどのフィールドが含まれたかを示すために、SIGHASH_TYPEバイト1210を含む。

【０２３３】

前のトランザクションのロックスクリプトは、アンロックスクリプトにアンロックトランザクションを含めるようにOP_GENSIGとOP_CHECKSIGを使用して、前のトランザクションのTxIDとOP_EQUALVERIFYを使用して、アンロックスクリプトにPrevious TxZの直列化バージョン1206を含めるようにする。Previous TxZのTxIDをアンロックトランザクションのフィールドから抽出し、Previous TxZを挿入することができる。Previous TxZそれ自体はPrevious TxZ'のTxIDを含んでいるので、そのTxIDを抽出することができる。以下同様である。

【０２３４】

アンロックトランザクションの直列化されたトランザクションフィールドのセットは、アンロックトランザクション自体のアンロックスクリプトを含まない。そうでなければ因果関係の問題が生じる。いくつかの実施形態、例えばBitcoinブロックチェーン環境において、トランザクションの署名ハッシュは、いくつかの所定のフィールドを除外することができる。SIGHASH Typeは、直列化されハッシュされる前に、どのフィールドの集合がトランザクションから抽出されるかを表してよい。

【０２３５】

前のトランザクションのロックスクリプトは、アンロックトランザクションが有効であるためには、アンロックトランザクションのアンロックスクリプトが、アンロックトラン

ザクションのフィールドの一部又は全部のコピーと、アンロックトランザクションがアンロックする前のトランザクションのうちの1つ以上への参照を含める必要があるという要件をアンロックトランザクションに課すことができる。もちろん、アンロックトランザクションのフィールドは、アンロックトランザクションが作成されるとき変更可能である。しかし、前のトランザクションのロックスクリプトは、アンロックトランザクションのアンロックスクリプトの一部であるアンロックトランザクションのフィールドが制約を満たさない場合、アンロックトランザクションがロックスクリプトの実行時に検証しなくてよいという点で、アンロックトランザクションのフィールドに制約を課すことができる。アンロックスクリプトの一部ではないフィールド（SIGHASH Type値によって含まれない、又はゼロに設定されないフィールドなど）は、ロックスクリプトによって制約されなくてよい。また、直列化されたアンロックトランザクションに含まれるフィールドのすべてが制約を必要とするわけではない。例えば、ロックスクリプトは、アンロックスクリプトに含まれる、直列化されたアンロックトランザクションの多くのフィールドのうちの1つを制約するだけかもしれない。

【０２３６】

前のトランザクションの異なるアウトプットは、おそらく異なるアンロックトランザクションの異なるインプットとアウトプットに対して、異なる制約セットを適用することができる。このような技術を用いて、ブロックチェーンノードは、このようなロックスクリプトを用いてトランザクションを生成することができ、このロックスクリプトは、生成されたトランザクションアウトプットが移転され、そのトランザクションアウトプットをアンロックするトランザクションが、必然的に様々なデータ要素及び状態を運ぶことができるように、さらに他のチェックを実行することができる。

【０２３７】

図12のトランザクションは、かなりの複雑な機能を有する状態機械を実装するために拡張することができる。上述のように、ロックスクリプトは、検証時に、ロックスクリプトが作成され確定された後にのみ利用可能なデータにアクセスできる。そのデータには、検証中のアンロックトランザクションのフィールドの詳細、及びロックスクリプトを含む前のトランザクションの詳細（及び、そのときにアンロックトランザクションを作成しているブロックチェーンノードに知られている他の前のトランザクションも可能である）を含めることができる。これにより、状態機械の状態がスクリプト内で表現され、状態及び状態遷移が実装され、ブロックチェーン環境のトラストレスな範囲内で発生することが可能になる。言い換えれば、状態機械は、必ずしも信頼性のないブロックチェーンノードによって検証されるトランザクションを用いて、ブロックチェーンネットワークによって処理される他のトランザクションと同様にトランザクションを処理することによって、実装することができる。いくつかの例では、「トラストレス」（trustless）は、制約が満たされる限り、どのエンティティも有効なアンロックトランザクションを作成することができる、及びそのエンティティが信頼される必要がない、というプロパティを表す。しかし、場合によっては、エンティティが必要なインプットを得るために、決定されたソースと対話する必要がある場合がある。いくつかの実施形態において、1つ以上のインプットは、直列化された前のトランザクションなど、未決定のソースから得ることができる。

【０２３８】

ブロックチェーンネットワークでは、トランザクションは、そのインプットのいくつかの値（トランザクションインプットがアンロックする前のトランザクションのトランザクションアウトプット値によって決定される）を持ち、そのアウトプットのいくつかの値を持つ。有効であるためには、アウトプットの合計はインプットの合計を超えることはできず（この説明では除外され得る分配トランザクション及びジェネシストランザクションを除く）、また、マイナーノードに取り込まれるためには、アウトプットは、マイナーノードのオペレータが参加する理由を提供するために、インプットよりも小さくされてよい。ブロックチェーンネットワークを使用してトラストレスな決定性状態機械を実装する場合、状態機械の初期状態は、状態機械の状態遷移に対応するトランザクションを含む理由を

マイナーに提供するのに十分な価値を含むトランザクションの中に存在し得る。

【０２３９】

一般的な場合には、前のトランザクションのアンロックスクリプトを状態機械を実装する際に使用することができる。その結果、アンロックトランザクションは、アンロックトランザクションの内容を制約する、特にアンロックトランザクションの種々のトランザクションアウトプットの内容及び性質を制約するロックスクリプトにより、状態機械の状態遷移を表す。従って、前のトランザクションのアウトプットのロックスクリプトは、実質的に、アンロックトランザクションの各トランザクションアウトプットがどのようになる必要があるかを指定することができる。ロックスクリプトは、例えば、アンロックトランザクションのVout[0]が特定のデータ及びスクリプト要素を含むこと、ならびにアンロックトランザクションのVout[1]が特定の他のデータ及び他のスクリプト要素を含むことを要求することによって、異なるアンロックトランザクションアウトプットの異なる特徴を要求することができる。

【０２４０】

SIGHASH Typeと特定のSIGHASH Typeが使用するアンロックトランザクションフィールドセットを提供するエンティティは、トランザクションアウトプットをアンロックできる。このようにして、いくつかのパーティは、アンロックし、ロックスクリプトから詳細を転送することによって、トランザクションの内容を伝搬する理由を有する。上述したように、いくつかの例では、トランザクションアウトプットを「アンロック」することは、トランザクションアウトプットを参照し及び有効と評価し得るアンロックトランザクションを作成することを表し、それによってトランザクションアウトプットの移転を生じ得る。

【０２４１】

上述の技術及び装置を使用して、トランザクションは、スマートコントラクトを実装するスクリプトを含み、状態機械を実装することができ、現在のトランザクションのフィールド、現在のトランザクションによって移転されるアウトプットを有する前のトランザクション、及び場合によっては他の前のトランザクションを参照することができるスクリプトを含むブロックチェーン台帳上に作成することができる。これらの機能により、トランザクションフローは、スマートコントラクトなどで使用する自己複製スクリプトを強制的に実行できる。

【０２４２】

図13は、アンロックトランザクション1320及び前のトランザクション1310を示す。この例では、アンロックトランザクションは、前のトランザクションと前のトランザクションのトランザクションアウトプットVout[x]1314とを参照するインプットVin[0]1322を有する。アンロックトランザクションが有効とみなされるためには、Vout[x]のロックスクリプトの制約が満たされなければならない。上述したように、いくつかの制約は単純であってよい。例えば、ロックスクリプトを指すアンロックトランザクションインプットのアンロックスクリプトが、ロックスクリプトに現れる公開鍵によって署名された有効な署名を提供できなければならないという単純な要件である。しかし、本明細書に記載されるように、ロックスクリプトは、より複雑な制約を提供することができる。

【０２４３】

例えば、前のトランザクション1310のVout[x]1314のロックスクリプトは、特にアンロックトランザクションのアウトプットに制約を課すことができる。この例におけるVout[x]1314のロックスクリプトは、アンロックトランザクション1320のアウトプットVout[0]1324に一連の制約（Constraint Set1）を課し、アンロックトランザクション1320のアウトプットVout[0]1326に一連の制約（Constraint Set2）を課す。特定の場合において、Constraint Set1及びConstraint Set2は、同一であり、両方とも、アウトプットVout[0]1324及びVout[1]1326のロックスクリプトが、前のトランザクション1310のVout[x]1314のロックスクリプトと同じであることを単に必要とするだけである。

【０２４４】

より一般的なケースでは、制約は、単にロックスクリプトをコピーするという要件を超

えることがある。さらに、制約セットは、アンロックトランザクションアウトプットからアウトプットへと変化し得る。これらの制約は、アンロックトランザクションが持つことができるアウトプットの数、各アウトプットの値、及びアンロックトランザクションのアウトプットのロックスクリプトの特定の内容を必要とすることであり得る。

【０２４５】

注入されたアンロックトランザクションのアウトプットに対する制約

図14は、アンロックトランザクションのロックスクリプトに制約を課す前のトランザクションのロックスクリプトの一部分1400の例のフローチャートである。この例は、スタック上にデータを残し得る、アンロックトランザクションのアウトプットのアンロックスクリプトを処理し、次に、前のトランザクションのアウトプットのロックスクリプトを処理して、この処理を実行するスタックベースの検証器によって実行されてもよい。アンロックスクリプトの処理は、スタック上にデータを残すことができるので、また、アンロックスクリプトは、例えば、アンロックトランザクションのロックスクリプトの詳細を含む、アンロックトランザクションの直列化された内容をスタック上に残すことができるので、これらの詳細は、検証時に利用可能である。

【０２４６】

プロセスのステップ1402において、検証器は、アンロックスクリプトが特定のデータを含むことを検証し、検証器は、アウトプット毎に行う必要のない他の検証を実行し得る。そのようなデータの例は、直列化されたアンロックトランザクションフィールドのセット、直列化された前のトランザクション、前のトランザクションへの参照などを含み得る。ステップ1404で、検証器は、これらの制約が満たされているかどうかをチェックし、満たされていない場合は、ステップ1406で停止し、アンロックトランザクションを無効にする。さもなければ、処理はステップ1408に進み、検証器はチェックすべきアンロックトランザクションアウトプットがあるかどうかを決定する。1回目で、検証器は、インデックスi=0を設定してよく、チェックすべきアウトプットがそれ以上ない場合は、ステップ1410で別の処理フローに移動する。

【０２４７】

アウトプットの各インデックスについて、検証器はステップ1412、1414、及び1416を実行し、次いで、iをインクリメントし（ステップ1418）、考慮すべきインデックスがなくなるまでステップ1408に戻る。別の変形では、インデックスを1つずつインクリメントし、各アウトプットを検証する代わりに、検証器はインデックスのセットを通じてループし、セットは必ずしもiのすべての値を含まず、及び／又は、必ずしも追加的順序で、それらを含み、それらをチェックしない。そのような変形では、ステップ1418は、それに応じて異なり、チェックされる特定のインデックスのセット内のアウトプットについて、インデックスiを通じてループする。

【０２４８】

ステップ1412において、検証器は、アンロックトランザクションのVout[i]のロックスクリプトを抽出する。これはスタック内容に対するOP_SUBSTR操作を使って実装され得る。次に、ステップ1414において、検証器は、前のトランザクションのアウトプットのロックスクリプトに基づいて、アンロックトランザクションのVout[i]のロックスクリプト上に何の制約があるかを決定する。ステップ1416において、検証器は、次に、これらの制約を検証する。このようにして、アンロックトランザクションのインプットによって移転される前のトランザクションのアウトプットのロックスクリプトは、そのアンロックトランザクションのアウトプットに制約を課すことができる。

【０２４９】

前のトランザクションのアウトプットのロックスクリプトは、前のトランザクションロックスクリプトがアンロックトランザクションのアウトプットを移転する方法に関するルールを指示できるように、アンロックトランザクションのアウトプットを制約する。これらの制約は、前のトランザクションのアウトプットのロックスクリプトでハードコーディングすることができ、又は、前のトランザクションのそのアウトプットを指すアンロック

トランザクションのインプットのアンロックスクリプト内に存在するデータに基づいて、制約を決定することができる。

【０２５０】

図15は、図14のステップ1402及び1414に対応するより具体的なステップのいくつかの例を示す。ステップ1402を実行する検証器の一例は、ステップ1502を実行することであり、検証器は、アンロックトランザクションのインプットにおけるアンロックスクリプト（又は他の場所）内のデータの一部が決定されたソースから来たと判断する。決定されたソースは、アンロックトランザクションの直列化又は直列化された前のトランザクションであってよい。あるいは、さらに、検証器は、ステップ1504を実行し、アンロックトランザクションデータの詳細の何らかの他の検証を実行することができる。

【０２５１】

前のトランザクションのアウトプットのロックスクリプトが、アンロックトランザクションのアウトプットのロックスクリプトに課し得る特定の制約に関して、ステップ1506があり、アンロックトランザクションのアウトプットVout[i]のロックスクリプトは、何らかの決定されたソースからのデータと等しくなければならない（「==」は等価性のテストを表す）。別の例は、ステップ1508であり、アンロックトランザクションのアウトプットVout[i]のロックスクリプトが、前のトランザクションのアウトプットのロックスクリプトにハードコードされたデータと等しくなければならない。ステップ1510は、決定されたソースからのデータに基づいて、アンロックトランザクションのアウトプットVout[i]に別の制約が課される場合をカバーし、ステップ1512は、前のトランザクションのアウトプットのロックスクリプトからのハードコードされたデータに基づいて、Vout[i]に別の制約が課される場合をカバーする。

【０２５２】

場合によっては、ソースからの又はハードコードされたデータの完全な範囲を使用する代わりに、データのハッシュを使用することができる。この場合、比較はハッシュからハッシュへ行われる。

【０２５３】

使用されるべきデータは、前のトランザクションから、例えば前のトランザクションのアウトプットのロックスクリプトにハードコードされたものからであってもよいし、前のトランザクションに先行するトランザクション（又は先行するトランザクションへの参照、アンロックトランザクションから出る2つのリンク（すなわち、前のトランザクションによってそのアウトプットの1つ以上が移転されたトランザクション）からのものであってもよい。このようにして、使用されるデータは、前のトランザクションからその分配トランザクションへのチェーン内の任意のトランザクション（例えば、Bitcoinプロトコルのcoinbaseトランザクション）から得ることができる。

【０２５４】

図16は、この一例を示す。図14のプロセス1400におけるステップ1402（検証は、アンロックスクリプトが特定のデータを有することである）について、ステップ1602、1604、1608、1610、及び1612を実行して、アンロックトランザクションのアンロックスクリプトがトランザクションチェーンの中のトランザクションへの参照を有することを検証することができる。

【０２５５】

ステップ1602で、検証器は、アンロックトランザクションのインプットからトランザクションID（TxID）を抽出する。これは、TxIDがアンロックスクリプトで利用可能であれば、行うことができる。次いで、検証器は、ステップ1604において、アンロックスクリプトが、そのTxIDの直列化された前のトランザクションのコピーも含むことを検証することができる。含まない場合、検証器はトランザクションを無効にすることができるが、含む場合、ステップ1608で、検証器は、前のトランザクションTxID'からTxIDを抽出し、次いでステップ1610で、前のトランザクションが、TxID'によって参照された直列化されたトランザクションのコピー又はそれへの参照を含むことを検証する。このチェーンは、n回継

続することができ、ここで、ステップ1612において、検証器は、TxID$^{(n-1)}$'(n－1ダッシュ）によって参照されるトランザクションが、TxIDn'(nダッシュ）によって参照される直列化されたトランザクションのコピーを含むことを検証する。

【０２５６】

図16において、ステップ1614及び1616は、TxID^{n}'によって参照されるトランザクションのコピーのためのロックスクリプトを抽出し、アンロックトランザクションのアウトプットVout[i]のロックスクリプトが、TxID^{n}'によって参照されるトランザクションの抽出されたロックスクリプトと等しいという制約を検証するステップを含む。

【０２５７】

制約は、ロックスクリプト全体ではなく、アウトプットのロックスクリプトのサブセットに適用できる。制約は、データ全体ではなく、データのサブセットから決定できる。例えば、あるサブセットは「Pay－to－Public－Key－Hash」が表れることを要求し、別のサブセットは「Pay－to－Bob」が表れることを要求することがある。

【０２５８】

アンロックトランザクションのアウトプットのロックスクリプトが、<x><y>の2つの部分で表現できるとする。ここで、<x>と<y>は、<x>と<y>に個別に制約が適用されるように、文字列連結操作を使って抽出できる。たとえば、<x>は<PubK A>に制約され、<y>はOP_CHECKSIGに制約される。このようにして、全体的なロックスクリプトは<PubK A>OP_CHECKSIGに制約される。また、アンロックスクリプト内のデータから導出された制約を使用することも可能である。

【０２５９】

これらの技術を使用して、前のトランザクションのアウトプットのロックスクリプトは、そのアウトプットのアンロックを試みるアンロックトランザクションに制約を課すことができる。特に、制約は、アンロックトランザクションが有するアウトプットのセットに対する制約であり得、制約は、異なるアウトプットに対して異なり得、また、前のトランザクションのロックスクリプトとは異なり得る。

【０２６０】

制約は、様々なアウトプットを参照するインデックスに対応することができる。ある場合には、制約は各インデックスに対して繰り返されるが、他の場合には、幾つかの構造を提供することができる。たとえば、制約のインデックスは、"if index==0: constraint A els constraint B"などの制約の集合を選択するために条件文で使用できる。

【０２６１】

一連のインデックスがハードコードされている場合、そのインデックスを使用するスクリプトの部分の結果は固定されてよい。アンロックトランザクションのアウトプットに対する制約の代わりに、又はそれに加えて、アンロックトランザクションのインプットに対する制約も存在し得る。

【０２６２】

注入されたアンロックトランザクションのインプットに対する制約

アンロックトランザクションのインプットを制約するロックスクリプトを持つことの1つの用途は、ロックスクリプトが特定のプロパティを持つトランザクションへの依存関係（例えば、制約されたインプットはそのトランザクションを参照しなければならない）を符号化することを可能にすることである。これは、アウトプットが可変であるという点で、アンロックトランザクションのアウトプットを制約することとは異なる。一方、インプットは、ブロックチェーン上の既存の不変のUTXOへの参照であってよい。言い換えれば、アウトプットは、制約に合致するように変更することができ、一方、それらの制約に合致するインプットを見つけなければならない。

【０２６３】

アンロックトランザクションのインプットに対する制約は、（1）インプットの参照（すなわち、トランザクションTxIDフィールド、及び任意でアウトプットポインタフィールド）に直接設定される制約と、（2）インプットの中の参照が指す前のトランザクション

のフィールドに対する制約、の2つの一般的なタイプに分類される。

【０２６４】

図17は、前のトランザクション1710とアンロックトランザクション1720を示し、前のトランザクション1710は、インプットVin[y]1712とアウトプットVout[x]1714を有し、アンロックトランザクション1720は、インプットVin[0]1722及びVin[1]1723と、アウトプットVout[0]1724と、を有する。Vout[x]1714のロックスクリプトは、インプットVin[0]1722に対して制約セット（Constraint Set1）を課し、インプットVin[0]1723に対して制約セット（Constraint Set2）を課す。制約セットは、アンロックスクリプト、参照、又はインプットの他の部分に対する制約を含むことができる。制約セットは、同じか、異なるか、又は2以上のインプットがあり、一部は同じインプットであり一部場異なるインプットであり得る。

【０２６５】

部分的には、これらの制約は、OP_SPENDINGTXINJECTIONスクリプトを使用するなどして、直列化されたアンロックトランザクションフィールドのセットをアンロックスクリプトにデータとして注入させることにより、アンロックトランザクションの部分を参照することができる。

【０２６６】

図18は、アンロックトランザクションのインプットに制約を課す、前のトランザクションのロックスクリプトの一部分1800の例のフローチャートである。この例は、アンロックトランザクションのアウトプットのアンロックスクリプトを処理する検証器によって実行されてもよい。検証器は、アンロックトランザクションの詳細を含むデータをスタックに残し、次いで、前のトランザクションのアウトプットのロックスクリプトを処理してこの処理を実行する。

【０２６７】

プロセスのステップ1802において、検証器は、アンロックスクリプトが特定のデータを含むことを検証し、検証器は、アウトプット毎に行う必要のない他の検証を実行し得る。ステップ1804において、検証器は、これらの制約が満たされているかどうかをチェックし、満たされていなければ、ステップ1806で停止し、アンロックトランザクションを無効にする。さもなければ、処理はステップ1808に進み、そこで検証器はチェックすべきアンロックトランザクションインプットがあるかどうかを決定する。1回目で、検証器は、インデックスi=0を設定することができ、チェックすべきインプットがそれ以上ない場合は、ステップ1810で別の処理フローに移動する。

【０２６８】

インプットの各インデックスについて、検証器は、ステップ1812、1814、及び1816を実行し、次いで、iをインクリメントし（ステップ1818）、考慮すべきインデックスがもう存在しなくなるまでステップ1808に戻る（又は、検証器は、必ずしもすべてのインデックス及び／又は必ずしも増加的順序ではなく、一連のインデックスを通じてループする；これらの場合、ステップ1818は相応に異なり、インデックスiを介してインプットがチェックされる間、ループする）。ステップ1812において、検証器は、アンロックトランザクションインプットVin[i]から、前のトランザクションへの参照及び自身のアンロックするアウトプットを抽出する。これはスタック内容に対すｒうOP_SUBSTR操作を使って実装され得る。次に、ステップ1814において、検証器は、前のトランザクションのアウトプットのロックスクリプトに基づいて、何の制約が参照に対してあるべきか、又は、前のトランザクションの注入を引き起こすためにその参照を使用するかを、該制約が前のトランザクションのフィールドに適用可能になった後に、決定する。ステップ1816において、検証器は、次に、これらの制約を検証する。このようにして、アンロックトランザクションのインプットによって移転される前のトランザクションのアウトプットのロックスクリプトは、そのアンロックトランザクションのインプットに制約を課すことができる。

【０２６９】

インプット制約の他の例は、要素1902、1904、1906、1910、1912、及び1914によって図

19に示されている。TxIDはトランザクションと1対1にマッピングされてよいので、これは、特定のトランザクションがインプットの中で参照されなければならないという制約であることを意味し得る。ロックスクリプトが依存するトランザクションは、最初に作成され、それによってロックスクリプトは、アンロックトランザクションのインプットに対する制約の中で、自身のTxIDを使用することができる。

【０２７０】

インプット中の参照に対応する前のトランザクションのフィールドに制約を設定できる。これは、前のトランザクションのフィールドに制約を課す前に、対応する前のトランザクションの注入を引き起こすために、インプットの中のTxIDを使用することによって可能であってよい。制約は、制約を課すロックスクリプトを持つ前のトランザクションのアウトプットをアンロックするインプットとは異なるインプットを参照できる。

【０２７１】

図20は、図18のプロセス1800におけるステップ1802及び1814に関連する制約の例を示す。図20では、ステップ2002、2004、2014及び2016を実施することができる。ステップ2002において、検証器は、アンロックトランザクションのインプットからトランザクションID（TxID）を抽出する。これは、TxIDがアンロックスクリプトで利用可能であれば、行うことができる。次に、検証器は、ステップ2004において、アンロックスクリプトが、そのTxIDのための直列化された前のトランザクションのコピーも含むことを検証することができる。ステップ1614で、検証器は、前のトランザクションから1つ以上のフィールドセットを抽出し、ステップ1616で、抽出されたフィールドセットに関する何らかの制約を検証する。

【０２７２】

図21は、インターロックスクリプト制約を示す。ここに示すように、前のトランザクション（Previous　Tx1）2102及び前のトランザクション2104（Previous　Tx2）は、アンロックトランザクション（Unlocking　Tx1）2106のインプットによって参照されてもよい。特に、矢印2110によって示されるように、Unlocking　Tx1のインプットVin[0]は、Previous　Tx1のVout[0]を指す。したがって、Unlocking　Tx1が有効であれば、Previous　Tx1のVout[0]をアンロックすることができる。また、矢印2112で示すように、Unlocking　Tx1のインプットVin[1]は、Previous　Tx2のVout[0]を指す。このような参照は、複数の前のトランザクションを参照する複数のインプットを伴うアンロックトランザクションのための従来の参照であってよい。

【０２７３】

前のトランザクションを参照するために、これらのトランザクションは、アンロックトランザクションの作成前に既に存在し、不変である必要がある。それにもかかわらず、前のトランザクションは、アンロックトランザクションに相互依存ロックスクリプト制約を適用することができる。アンロックトランザクションのインプットの集合に制約を課すことによって、相互依存ロックスクリプトを実装することができ、これは、とりわけ、トラストレスな決定性状態機械の文脈において、並行性を提供することができる。

【０２７４】

Previous　Tx1のアウトプットVout[0]のロックスクリプト2120は、インプットのポインタ2122（例えば、前のトランザクション及び前のトランザクションのアウトプットへの参照）がPrevious　Tx2及びPrevious　Tx2のVout[0]への参照であることを要求する、Unlocking　Tx1のインプットVin[1]に対する制約を有する。Previous　Tx2のアウトプットVout[0]のロックスクリプト2124は、インプットのポインタ2126（例えば、前のトランザクション及び前のトランザクションのアウトプットへの参照）がPrevious　Tx1及びPrevious　Tx1のVout[0]への参照であることを要求する、Unlocking　Tx1のインプットVin[0]に対する制約を有する。

【０２７５】

トランザクションのTxIDは、トランザクションが終了するまで知られていないので、トランザクションロックスクリプトは、アンロックトランザクションアンロックスクリプト

が参照すべきTxIDを直接指定することはできないが、基本的には、ロックスクリプトがインプットのポインタ（ロックスクリプトを持つ前のトランザクションのTxIDを含む）を抽出することによって、これを実行し、そのTxIDを使用して、前のトランザクションのフィールドに注入し、前のトランザクションの一部のフィールドが、想定される値に等しいことを検証できる。言い換えると、前のトランザクションがある値に設定されたフィールドを持ち、前のトランザクションのロックスクリプトがアンロックトランザクションインプットからTxIDを取得し、そのTxIDが参照するトランザクションを注入し、そのフィールドが特定の値に設定されていることをチェックし、そのフィールドがその特定の値に設定されていない場合、アンロックトランザクションインプットが異なる前のトランザクションを指している場合でなければならない。したがって、特定の値をチェックすることによって、前のトランザクションのアウトプットのロックスクリプトは、アンロックトランザクションインプットが前のトランザクションのアウトプットを参照しているかどうかを確認できる。

【０２７６】

このように、Previous Tx1は、その第1アウトプットが、特定のトランザクションの第1アウトプットを同様にアンロックするトランザクション（Previous Tx2）によってのみ移転可能であると述べており、Previous Tx2は、その第1アウトプットは、Previous Tx1の第1アウトプットを同様にアンロックするトランザクションによってのみ移転可能であると述べている。

【０２７７】

相互依存ロックスクリプトは、とりわけ、互いに条件を設定する複数のスマートコントラクトを可能にする。例えば、スマートコントラクト1は「AliceはCarolに1BTCを拠出することに同意するが、BobがCarolに2BTCを拠出する場合に限る」であり、スマートコントラクト2は「BobはCarolに2BTCを拠出することに同意するが、AliceがCarolに1BTCを拠出することに同意する場合に限る」である。有効であるためには、アンロックトランザクションは両方のスマートコントラクトをアンロックしなければならない。

【０２７８】

上述のように、ロックスクリプトは、特定のプロパティを持つトランザクションへの依存関係を符号化することができ、それらのプロパティは、そのトランザクションのフィールドを含むことができる。これは、ロックスクリプトを、特定のロックスクリプトを有するトランザクションに依存させることが可能であることを意味する。

【０２７９】

例えば、図21の例では、（Previous Tx1のVout[0]の）ロックスクリプト#1は、（Previous Tx2のVout[0]の）ロックスクリプト2124を参照するようUnlocking Tx1のVin[1]を制限し、ロックスクリプト#3 2124は、Previous Tx1のVout[0]のロックスクリプト#1を参照するようUnlocking Tx1のVout[0]を制限する。ここでは、これは、別のロックスクリプトに対する依存関係を符号化するロックスクリプトと呼ぶことができる。

【０２８０】

相互に依存するロックスクリプトを実装することは、因果関係の競合条件を作り出す可能性があり、第1ロックスクリプトは第2ロックスクリプトへの依存関係を符号化し、第2ロックスクリプトは第1ロックスクリプトへの依存関係を符号化する。なぜなら、Previous Tx1及びPrevious Tx2のうちの一方が他方よりも先に存在する可能性があるからである。ロックスクリプトAがロックスクリプトBへの依存関係をハードコードする場合、ロックスクリプトBはすでに既知であり確定されている必要があり、ロックスクリプトBが未知のロックスクリプトAへの依存関係をハードコードすることは不可能である。これは、少なくともロックスクリプトの1つにおいて、決定可能な依存関係を有することによって対処することができる。

【０２８１】

例えば、ロックスクリプトAはロックスクリプトXに依存し、ロックスクリプトXはアンロックスクリプトA内で提供され、ロックスクリプトBはアンロックスクリプトB内で提供

されるロックスクリプトYに依存する。アンロックトランザクションが作成されると、ロックスクリプトAとロックスクリプトBの両方が既知であるため、アンロックトランザクションのアンロックスクリプトAはロックスクリプトBをそのデータとして含み、アンロックトランザクションのアンロックスクリプトBはロックスクリプトAをそのデータとして含むことが可能となる。

【０２８２】

一部の変形では、アンロックトランザクションインプットを検証するために、そのアンロックスクリプトは、Previous Tx1及びPrevious Tx2を参照しなければならない。他の変形では、アンロックスクリプトは、Previous Tx1と同じロックスクリプトを持つ2つ（又はそれ以上）の前のトランザクションを参照することで十分である。

【０２８３】

いくつかの変形では、アンロックスクリプトが特定の前のトランザクション又は複数の前のトランザクションの集合を参照するという要件を課す代わりに、要件がアンロックスクリプトが特定の前のトランザクション又は複数の前のトランザクションの集合を参照しないことであってよいという点で、その要件は排他的であってよい。これらのバリエーションの組み合わせも可能であり、どんな前のトランザクションが参照されるかについて、ブール論理を効果的に実装することができる。例えば、アンロックトランザクションインプットを検証するために、そのアンロックスクリプトは、Previous Tx1及びPrevious Tx2への参照を持たなければならず、Previous TxAへの参照を持たず、Previous TxB又はPrevious TxCのいずれかへの参照を持たなければならない。他の論理表現も可能であってよい。

【０２８４】

状態機械

上記の技術を使用して、状態機械は、ブロックチェーントランザクションを使用して実装できる。ここで、トランザクションは、状態を有し、トランザクションマトリクスによるような状態機械構造によって符号化され、そして次の状態を許容される。アンロックトランザクションは、そのようなブロックチェーントランザクションのアウトプットをアンロックすることができ、事実上、状態機械の状態遷移を引き起こす。

【０２８５】

図22は、ブロックチェーントランザクションを使用して実装される状態機械の例2200を示す。具体的には、図22は、ブロックチェーントランザクションを使用して第1の状態から第2の状態に遷移する状態機械を示す。いくつかの例では、状態機械の状態遷移は、（1）現在状態、（2）1つ以上のインプット2226、及び（3）状態規則セット2206が与えられると、次の状態を決定するものとして説明される。図22の例2200は、状態規則セット2206と、パラメータに埋め込まれた第1状態2228Aとを有する前のトランザクション2202を示す。いくつかの実施形態では、アンロックトランザクション2204は、決定されたソースからのインプット2226を受け入れるために作成される。インプット2226は、第1状態2228Aと組み合わせて、状態規則セット2206を参照して、アンロックトランザクション2204のパラメータに埋め込まれた第2状態2228Bを決定するために使用可能であってもよい。

【０２８６】

実施形態では、状態規則セット2206は、ロックスクリプトによって課されるアンロックトランザクション2204に対する制約によって表すことができる状態トランザクションマトリクスを含む。そのような実施形態では、制約は、現在状態及び次の状態が決定されるインプットによってパラメータ化され得る。制約は、アンロックトランザクション2204が、特定のフィールドに次の状態値を含むアウトプットを含むことを確かめるためのチェックを含むことができる。

【０２８７】

実施形態では、現在状態は、トランザクションのアウトプットのロックスクリプトの一部など、トランザクションに埋め込まれたパラメータとして表現され、アンロックトランザクション2204は、アンロックトランザクション2204に埋め込まれた次の状態値を有して

もよい。次の状態値は、アンロックトランザクション2204のフィールド値のセットに対する現在状態であってもよく、上記のように、アンロックトランザクション2204が生成されるときにアクセス可能であってもよい。

【０２８８】

いくつかの実施形態では、少なくとも1つのインプットが、アンロックトランザクション2204が生成されるときに決定されるパラメータ内の外部データとして提供される。セキュアであるために、このようなパラメータは、決定されたソースに由来する。これは決定性状態遷移を提供することができる。最後に、自己複製ロックスクリプトのスクリプトを使用することによって、トラストレスな決定性状態機械の実施形態を作成することができる。さらに、異なるロックスクリプトがトランザクションの異なるトランザクションインプット及びアウトプットに課されることを可能にすることによって、並行の（concurrent）トラストレスな決定性状態機械を実装することができる。

【０２８９】

図23は、この点を示す。この例では、トラストレスな決定性状態機械システム2300は、状態規則セット2306内の第1状態2328A（「S1」）における前のトランザクション2302を含んでもよい。状態規則セット2306は、次の状態に対して2つの可能な状態2330A（「S2」又は「S3」）を提供する。状態規則セット2306は、前のトランザクション2302のアウトプットのロックスクリプト内に符号化されてもよい。現在状態2328Aは、Lock time（Lock time）フィールドに埋め込まれてもよいし、ロックスクリプトに埋め込まれてもよい。次の状態、本例ではS2 2328Bは、前のトランザクションのアウトプットをアンロックして状態遷移を実行するアンロックトランザクションのアウトプットのロックスクリプト内に符号化される。

【０２９０】

例2300に見られるように、アンロックトランザクション2304は、インプットとして、そのアンロックスクリプト内のインプット2326と、前のトランザクション2302のフィールド値のセットに埋め込まれた第1状態2328A（「S1」）とを取り込み、状態規則セット2306から適切な第2状態2328B（「S2」）を決定する。さらに、例2300に見られるように、状態遷移マトリクスは、第2状態2328Bからの次の状態遷移に可能な新しい状態2330B（「S4」又は「S5」）を提供する。状態規則セット2306は、現在状態及び1つ以上のインプットによってパラメータ化されたswitch文又は他の条件文（例えば、「if－then－else」）として符号化されてもよいことに留意されたい。現在状態、S2 2328Bは、Lock timeフィールドに埋め込まれてもよいし、ロックスクリプトに埋め込まれてもよい。次の状態、本例ではS2 2328Bは、前のトランザクションのアウトプットをアンロックして状態遷移を実行するアンロックトランザクションのアウトプットのロックスクリプト内に符号化される。

【０２９１】

実施形態では、各可能な状態は、状態遷移マトリクスのような状態変化のための規則セットで、自己複製ロックスクリプト内で表現され、特定のトランザクションアウトプットは、特定の状態における状態機械を表す。このような実施形態では、トランザクションアウトプットのロックスクリプトは、デジタル資産の制御を、現在のトランザクションアウトプットにリンクされなければならない次のトランザクションに移転しようとするすべてのアンロックトランザクションに複製される。この過程は、終了条件が満たされるまで繰り返されてよい。インプットは確定されておらず、未決定のデータである可能性があるため、状態機械の状態は、特定の外部インプットに基づいて変更することができる。従って、未決定データは、次の状態に影響し得るインプットを提供する。

【０２９２】

説明のための例として、AliceはBobに資金を貸し付け、BobはAliceに返済することに同意する。本開示に記載されているような、トラストレスな決定性状態機械は、BobがAliceに対して行う支払いを表すスマートコントラクトとして実装することができる。例えば、スマートコントラクトは、Bobが今後3ヶ月間毎月Aliceに支払いを行い、支払いを怠るとBobの債務が回収段階に入るように構築することができる。従って、Bobが毎月支払う限り

、現在状態は返済状態のままである。しかし、Bobが支払いを怠ったことを示すインプットを外部のエンティティが提供する場合、状態は滞納状態に分岐する。滞納状態では、Aliceはトランザクションを解放し、それを債務徴収者に引き渡すことができ、その結果、トラストレスな決定性状態機械は債務徴収状態に切り替わる。債務徴収状態では、債務徴収者はBobから債権を回収する。このようなスマートコントラクトは、本明細書に記載するスクリプトのバリエーションを使用して作成することができる。

【０２９３】

別の例では、Aliceは非常に慈善的な人物で、毎月1単位のデジタル資産を寄付している。彼女の規則では、誰でもデジタル資産を請求できるが、月に1単位しか請求できない。Aliceは、本開示に記載される方法でスマートコントラクトを作成し、デジタル資産の初期プール3ユニットと共にそれを種蒔きする（seed）。Aliceは、任意のエンティティが月に1単位のデジタル資産を取得できるようなスクリプトを構築することができる。デジタル資産の残りの部分は、後続のスマートコントラクトに複製される。

【０２９４】

図24は、本開示の例示的なトラストレスな決定性状態機械のためのアンロックスクリプト及びロックスクリプトを示す。

【０２９５】

図25は、様々な実施形態による、トラストレスな決定性状態機械のためのプロセス2500の一例を示すフローチャートである。これを実行するスクリプトの例は、図24のスクリプトである。プロセス2500（又は記載の任意の他のプロセス、又はこれらのプロセスの変形及び／又は組み合わせ）の一部又は全部は、実行可能な命令及び／又は他のデータで構成される1つ以上のコンピュータシステムの制御下で実行可能であり、1つ以上のプロセッサ上で集合的に実行する実行可能な命令として実装可能である。実行可能な命令及び／又は他のデータは、非一時的なコンピュータ読み取り可能な記憶媒体（例えば、磁気媒体、光学媒体、又はフラッシュ媒体に永続的に記憶されたコンピュータプログラム）に記憶することができる。これらの命令は、ブロックチェーントランザクション検証プロセスの一部として実行される可能性がある。

【０２９６】

例えば、プロセス2500の一部又は全部は、図1の例示的なブロックチェーンネットワーク100など、例示的なブロックチェーンネットワーク内の検証ノードによって実行され得る。このような検証ノードは、（例えば、データセンター内のサーバ、クライアント計算装置、計算リソースサービスプロバイダの分散システム内の複数の計算装置、又は任意の適切な電子クライアント装置による）任意の適切な計算装置を含んでもよい。プロセス2500は、自己複製スマートコントラクトのロックスクリプトが検証され、現在状態が直列化された前のトランザクションから得られ、アンロックスクリプトからインプットが得られ、次の状態が少なくとも部分的に状態規則セットに基づいて決定される、一連の動作を含む。

【０２９７】

ステップ2502で、システムは、アンロックトランザクションを受信する。システムは、アンロックトランザクションインプットのアンロックスクリプトを実行することによって開始し、これにより、直列化された前のトランザクション及びロックスクリプトに埋め込まれたインプットがスタックに配置される。これらのインプットは、2512で取得することができる。2504では、プロセス2500を実行するシステムが、終了条件が満たされているか否かを判定する。実施形態において、終了条件は、完了すると状態機械遷移を終了させる条件である。終了条件が満たされる場合、プロセス2500を実行するシステムは、2506に進み、それによって、トラストレスな決定性状態機械は、自己複製及び／又は状態の伝搬を停止する。

【０２９８】

ステップ2508において、システムは、前のトランザクションのアウトプットのロックスクリプトがアンロックトランザクションのアウトプットのロックスクリプトと一致するこ

とを検証する。ロックスクリプトが一致しない場合、検証は失敗し、アンロックトランザクションは検証されない可能性がある。あるいは、ロックスクリプトが一致する場合、2510で、システムは直列化された前のトランザクションから可能な状態のセットのうち現在状態を抽出する。2512では、システムは、ロックスクリプトの実行の結果としてスタック上に置かれた1つ以上のインプットを得る。次に、2514において、システムは、現在状態及び1つ以上のインプットに基づいて、トラストレスな決定性状態機械の可能な状態セットの次の状態を決定するために、状態規則セットを適用する。2516において、システムは、次の状態（例えば、状態変数及び適用可能な場合には、他の状態関連データ）がアンロックトランザクションに埋め込まれていることを検証することができる。システムは、ロックスクリプト内で指定されているように、残りの制約を適用することもできる。2502～2516の動作が成功裏に完了した後、プロセスは2518で終了し、アンロックトランザクションは、プロセスを実行するシステムによって有効であるとみなすことができる。ステップ2502～2518で実行される動作のうちの1つ以上は、並列を含む様々な順序及び組み合わせで実行され得ることに留意されたい。

【０２９９】

開示された実施形態を説明する文脈において、別段の規定がない限り、「命令」が通常単独では実行しない動作（例えば、データの送信、計算など）を実行する実行可能な命令（コード、アプリケーション、エージェントなどとも呼ばれる）に関する表現の使用は、命令が機械によって実行されおり、それによって機械に指定された動作を実行させていることを意味する。

【０３００】

ブロックチェーントランザクションとしての状態機械

上述の技術及び装置を使用して、状態機械は、ブロックチェーントランザクションを使用して実装され得る。状態機械は、状態遷移マトリクスに沿って状態から状態に変化する状態を有することによって操作され得る。いくつかの状態図は、非常に複雑で、複数の経路を持つことができる。このような場合、状態機械が実行する計算のいくつかは、他の計算と並行に実行することができる。本明細書で説明するように、状態機械は、ブロックチェーントランザクションを使用して動作することができ、部分的に、並行に動作することができる。

【０３０１】

より詳細に説明されるように、トラストレスな決定性状態機械は、ブロックチェーントランザクションのロックスクリプト及びアンロックスクリプトを使用して実装され得る。ロックスクリプトは、アンロックスクリプトに課すことができる制約を含み、従って、アンロックトランザクション自身の直列化されたフィールド、実行されているロックスクリプトを含む前のトランザクションの直列化されたフィールド、現在状態値、決定されたソースからのインプット（任意）、及び状態機械の一連のフロー条件及び操作を定義する状態遷移マトリクスのような、アンロックトランザクションインプットのアンロックスクリプトへの特定のデータの注入を必要とすることができ、現在状態及び他のインプットから次の状態を計算する方法を規定する。アンロックトランザクションにおけるロックスクリプトの制約の一つは、アンロックトランザクションがロックスクリプトをそのアウトプットの1つの中に複製し、状態機械を伝搬しなければならないことである。

【０３０２】

図26は、特定の特徴を有する状態遷移マトリクス2600を使用する状態機械を示す。状態S1は、現在状態を示すためにハイライトされる。状態機械は、矢印で示される状態遷移に沿って任意の状態に遷移することができる。その状態遷移マトリクスでは、S1－S2－S3－S4－S8－S12に沿って遷移する可能性があるが、状態S5でもS1からの遷移が必要であってよい。同様に、状態S5はS6とS9に遷移する。状態S9から状態S10への遷移は、並行に生じる状態S6から状態S7への遷移が起こるまでは、状態遷移が生じてはならないという意味で、条件付きであり又はバリアを有していてもよい。状態S8への遷移は、状態S4と状態S7からの融合として起こる可能性がある。

【０３０３】
これらの状態の各々は、ブロックチェーントランザクションアウトプットによって表される状態機械によって開始され、各遷移はトランザクションアウトプットの単一パスによって処理され、アンロックトランザクションインプットが前のトランザクションのアウトプットをアンロックすると、図22又は図23に示されるような状態遷移が発生する。しかし、ブロックチェーントランザクションのインプット及びアウトプットを用いて、状態の並行処理を達成することができる。これは、スマートコントラクトを複数の並行ステップで実行する必要がある場合に便利である。契約の過程では、契約の過程で業務を並行に実行できることがある。例えば、家を建てる契約には、タイル工事の完成の様々な段階と造園工事の完成の様々な段階があるが、これらは並行して行うことができる。スマートコントラクト内での並行実行を可能にすることにより、コントラクトの異なるフラグメントを、非同期に、互いに独立して実行することができる。

【０３０４】
図26は、このような処理のための多くの必要性を示す。1つは、1つの状態から複数のパスを生成することである。これは、スマートコントラクトの枝分かれであり得る。スマートコントラクトが状態S1で始まると仮定する。このスマートコントラクトの分岐は、状態S2と状態S5の2つの状態機械を生成し得る。もう1つの必要性は、スマートコントラクトの複製であり、それぞれが同じ状態にあり、並列に動作する2つの同一の状態機械を持つことである。例えば、スマートコントラクトは、S2－S3－S4のステップの複数のインスタンスを要求し得る。これは、分岐の特別な場合である、2つ以上の新しいトランザクションアウトプットが同じ状態を持つ、スマートコントラクトのクローン化であり得る。

【０３０５】
もう1つの便利な特徴は、新規契約の創出である。例えば、図26の状態遷移マトリクス2600によって表されるスマートコントラクトは、状態S3において、スマートコントラクトの何らかのパーティが、スマートコントラクトに結び付けられ得るが、スマートコントラクトの一部ではない、及び／又はスマートコントラクトが作成された後に作成され得る別個のコントラクトを作成するようなものであり得る。本明細書では、これを創出（spawning）と呼ぶことができる。この新しいコントラクトは、最初に図26の状態遷移図に提供された最初のスマートコントラクトとは異なり得る。

【０３０６】
融合とは、状態S4と状態S7の場合のように、2つの状態機械又は2つのスマートコントラクトが融合（merge）することである。以下に説明するように、S2－S3－S4、S5－S6－S7のように、異なるパスを並行に実行することができる。これらの要素を用いて、並行のトラストレスな決定性状態機械をブロックチェーン環境で容易に実装できる。

【０３０７】
図27は、1つのインプットVin[0]及び2つのアウトプットVout[0]及びVout[1]を有するトランザクションの例を示す。このトランザクションのアウトプットをアンロックすることによって、2つの独立したトランザクションがブロックチェーン台帳上に置かれ、これらのトランザクションの作成者は、このトランザクションによって彼らに提示された何らかの価値（図示せず）によって補償されることになる。図に示すように、前のトランザクションのアウトプット（Vin[0]が指すもの）のロックスクリプトは、そのアウトプットをアンロックするアンロックトランザクションが状態S2とS5を含むことを要求する。状態S2を含むアンロックトランザクションのアウトプット（図27のVout[0]）をアンロックするために、将来のアンロックトランザクションは、現在状態がS2であることに基づき次の状態を含むように制約される。S5状態についても同様である。

【０３０８】
このようにして、トランザクションアウトプットは、状態を持つことができ、アンロックトランザクションがどの状態であることができるかに制約を課すことができる。これにより、ブロックチェーントランザクションは、図26に示すような状態図に従って処理を実装することができ、トランザクションアウトプットのアンロックは、状態図の状態遷移に

対応する。誰かが複数のインプットを持つ1つのトランザクションを作成し、示されたトランザクションの両方のアウトプットをアンロックする場合を避けるために、ロックスクリプトは、そのトランザクションをチェックし、それを許可しないようにすることができる。有効なトランザクションがどのように生成されるかのルール、必要なアウトプットの数などは、ハードコードされた値又はスマートコントラクトが作成された後に提供されるパラメータとしてスマートコントラクト内に符号化される。

【０３０９】

ブロックチェーントランザクションを使用したスマートコントラクトの枝分かれとクローン化

枝分かれは、ある状態が2つ以上の（必ずしも一意ではない）次の状態に並行に遷移する必要がある場合に使用される。スマートコントラクトは、全ての可能な並行性を伴わず処理できる場合もあるが、並行性が存在すべき場合は、枝分かれがそれを開始し得る。上述のように、ブロックチェーン台帳上のスマートコントラクトトランザクションのアウトプットのロックスクリプトは、アンロックトランザクションのインプットに、及び／又はブロックチェーン台帳上のアンロックトランザクションのアウトプットのロックスクリプトに要件を「課す」ことができる。クローン化は、次の状態の集合がすべて同じ状態であってもよい特定のタイプの枝分かれであってもよい。アンロックトランザクションに関しては、複数のアウトプットが遷移マトリクスを複製し、それぞれの次の状態を埋め込むために制約されてよい。

【０３１０】

枝分かれ/クローン化は、トラストレスな決定性状態機械の制約をアウトプットの集合に適用することによって達成することができる。上述のように、一連のインデックスは、異なるアウトプットに異なる制約を適用することを可能にする。この一連のインデックスは、遷移マトリクス内でハードコード化すること、又は検証プロセスへのインプットの一部として提供することができる。

【０３１１】

所与のアンロックトランザクションは、複数のアウトプットを有してもよく、そのうちのいくつかは状態機械とは無関係であってもよく、又は複数の無関係の状態機械に関連していてもよいので、Lock timeフィールドのようなすべてのアウトプットに共通のフィールドに次の状態を埋め込むよりも、トランザクションのロックスクリプトに次の状態を埋め込む方がより実用的である。アウトプットのロックスクリプトのサブセットを制約することについては、前述のとおりである。

【０３１２】

図28は、スクリプトが枝分かれ操作にどのように関連するかを示す。図示のように、状態機械2810は、前のトランザクション2820及びアンロックトランザクション2830を使用して実装される。前のトランザクション2820のロックスクリプトは、アンロックトランザクション2830が少なくとも2つのアウトプットを有するように制約し、各々は、後続のアンロックトランザクションが指定された状態でアウトプットを有する(及び別々のトランザクションである）ように要求する。これにより、これらの状態S2、S3を独立して、かつ可能な場合には並行に実行することができる。

【０３１３】

状態機械を使用すると、サブタスクは、より高速で並行実行するスマートコントラクトを可能にするのに役立つ。サブタスクを処理するために、複数の状態が使用されてよく、1つの状態機械は通常、一度に単一の状態でのみ使用され得るため、同時並行性は、別々のトランザクションスレッドを有することによって提供され得る。状態機械は、複数のトランザクションロックスクリプトを作成することによって枝分かれできる。前のトランザクションのアウトプットからの現在状態に基づいて、アンロックトランザクションは、図28に示されるように、ロックスクリプトを有する2つ以上のアウトプットを有するように制約され得る。枝分かれが償還不能になること、及び早期に停止することを防ぐために、各アウトプット値をチェックし、状態機械を継続するのに十分な資金があることを確認す

るべきである。

【０３１４】

破線の矢印によって示されるように、アンロックトランザクション2830のインプットVin[0]は、前のトランザクション2820及びそのトランザクションのアウトプットVout[0]を指す。Vout[0]のロックスクリプトは、現在状態（「S1」）、状態機械のための遷移マトリクスとを含み、アンロックトランザクション2830のVout[x]が次の状態（「S3」）となるよう、及びアンロックトランザクション2830のVout[y]が別の次の状態(「S2」）となるように制約し、各々が状態遷移マトリクスを伝えるように制約され、場合によっては他の制約も含む。

【０３１５】

前のトランザクションのアウトプットのロックスクリプトは、<Current State><Transition Matrix><other script>の連結で構成され、一方、アンロックトランザクションのアウトプットのロックスクリプトは、<NextState><TransitionMatrix><other script>の連結で構成されてよい。ロックスクリプト全体を複製する代わりに、ロックスクリプトのサブセットだけが複製されることがあり、他のサブセットは状態を含むように制約されてよい。

【０３１６】

状態をロックスクリプトに格納する代わりに、状態機械の状態をトランザクションのフィールドに格納することができる。このフィールドは、Lock timeフィールドなどの他の目的には使用できない。実際には、別の未使用のトランザクションフィールドを使用することができる。これは、単一の状態で動作し、複数の状態を格納する必要がある場合にも動作するが、それはより複雑である可能性がある。

【０３１７】

図29は、枝分かれ又はクローン化プロセスを実行するために実行され得る疑似コードの例を示す。この擬似コードは、本明細書に記載されるOPコードを使用して実装され得る。この擬似コードは、トランザクションを生成する及び／又はトランザクションを検証するブロックチェーンノードのプロセッサによって実行され得る。いくつかの実施態様において、スクリプトの一部はアンロックスクリプトからのものであり、スクリプトの一部はロックスクリプトからのものである。

【０３１８】

図29の擬似コードでは、前のトランザクション、アンロックトランザクション、及び任意的なインプットは、ロックスクリプトがそれらにアクセスできるように注入される。その後、終了条件のチェックが行われてよい。スマートコントラクトが終了していない場合、プロセッサは、前のトランザクションのロックスクリプト及び前のトランザクションの現在状態を抽出し、これは、本明細書に記載される技術を使用して行うことができる。次に、プロセッサは、アンロックトランザクションのアウトプットを通じてループして、それらの各々について、そのロックスクリプトを抽出し、前のトランザクションロックスクリプト内のトランザクションマトリクスをチェックして、それが処理されているアンロックトランザクションアウトプットのロックスクリプト内のトランザクションマトリクスと一致するかどうかを判定する。それらが一致しない場合、プロセッサはそれをスクリプト失敗として処理し、アンロックトランザクションを検証しない。それらが一致する場合、プロセッサは、抽出された現在状態、状態遷移マトリクス、任意にインプット、及びアウトプットインデックス値から、状態機械の次の状態を決定する。次に、プロセッサは、処理中のアンロックトランザクションアウトプットの次の状態に対して決定された次の状態をチェックする。それらが一致しない場合、プロセッサはそれをスクリプトの失敗として処理し、停止し、アンロックトランザクションを検証しない。チェックすべき他の任意の制約がある場合、それらはこのときにチェックすることができる。すべての制約が満たされると、スクリプトは終わる（pass）。

【０３１９】

作業者創出（worker spawning）スマートコントラクト

前節では、スマートコントラクトを枝分かれアンドクローン化する能力について述べた。このセクションでは、新しいスマートコントラクトをアンロックトランザクション内に強制的に存在させる機能について詳述する。アンロックトランザクションは、完全に異なる状態機械、又は完全に異なるタイプのスマートコントラクトではないが状態機械ではないスマートコントラクトであり得る。

【０３２０】

状態機械である新しいスマートコントラクトを強制するための制約の例は、図30に示される。示されるように、前のトランザクションのアウトプットのロックスクリプトは、アンロックトランザクションのアウトプットロックスクリプトが所望の新しいスマートコントラクトスクリプトと等しくなければならないように、又はアウトプットロックスクリプトのハッシュが所望の新しいスマートコントラクトスクリプトのハッシュと等しくなければならないように、アンロックトランザクションのアウトプットに制約を与える。新しいスマートコントラクトスクリプト又はハッシュは、前のトランザクションのアウトプットのロックスクリプトでハードコードされるか、又はアンロックトランザクションのインプットのアンロックスクリプトを介してパラメータとして安全に提供される。アンロックトランザクションアウトプットにおける複数の新しいスマートコントラクトは、それぞれの新しいスマートコントラクトについて前述のように、ロックスクリプト内に対応する制約を持つことによって強制され得る。

【０３２１】

上述のように、状態機械であってよい新しいスマートコントラクトが作成される。これは、前のトランザクションと同じ状態機械、又は別の完全に異なる状態機械であり得る。新しい状態と遷移マトリクスをチェックできる。これは、別々に(例えば、新しい状態をチェックし、その後、遷移マトリクスの値をチェックする）、又は並行に行うことができる。

【０３２２】

図30には示されていないが、新しいスマートコントラクトは状態機械以外であってもよい。そのような場合、次の状態値が必要とされてよく、又は必要とされなくてもよいが、新しいスマートコントラクトは、スクリプト又はそのハッシュの等価性チェックを使用して検証され得る。

【０３２３】

新しいスマートコントラクトは、前のトランザクションの現在状態からの遷移と、アンロックトランザクションのアウトプットのための新しいスマートコントラクトとで生成することができる。図示の例では、アンロックトランザクションの第1アウトプットセットは、状態遷移マトリクスを複製し、次の状態を埋め込むように制約され、第2アウトプットセットは、新しいスマートコントラクトを含むように制約される。アンロックトランザクションのアウトプットは、新しいロックスクリプトを含むように制約することができ、複数のインデックスを使用することによって、異なるアウトプットに異なる制約を適用することができる。一連の新しいスマートコントラクトは、遷移マトリクスでハードコードすること、又はインプットの一部として提供することができる。

【０３２４】

状態機械の観点から、状態機械は、呼び出し側状態機械と並行に実行するために、新しい異なる状態機械をインスタンス化することができる。このアプローチを状態機械に適用するために、この操作のためのロックスクリプトは、枝分かれ/クローン操作のものと同様であってよいが、そのロックスクリプトにおける等価性についてチェックされた各新しい状態機械インスタンスの代わりに、新しいスマートコントラクトの存在についてチェックされてもよい。等価性チェックは、トランザクションのための直列化されたトランザクションフィールドのセット、ロックスクリプトバイトコード、又はこれらのハッシュを含む、様々な異なる値の範囲に渡り実行することができる。

【０３２５】

図30は、新しいスマートコントラクトの創出を示す。ここに示されるように、状態機械

3010は、前のトランザクション3020及びアンロックトランザクション3030を使用して実装される。前のトランザクション3020のアウトプットVout[0]のロックスクリプトは、アンロックトランザクション3030が少なくとも2つのアウトプット、Vout[x]及びVout[y]を有するよう制約し、各々は、後続のアンロックトランザクションが指定された状態でアウトプットを有すること(及び別個のトランザクションであること）を要求する。具体的には、示されるように、アンロックトランザクション3030のインプットVin[0]が、前のトランザクション3020のVout[0]をアンロックするために、アンロックトランザクション3030は、Vout[x]が遷移マトリクスを有し、状態S4にあるように制約され得る一方、Vout[y]が新しいスマートコントラクトを有するように制約され得る。

【０３２６】

このアプローチでは、新しいスマートコントラクトを状態機械と並行して処理することができる。明示的には示されていないが、新しいスマートコントラクトは、第2状態機械であり得る。上述したように、新しいスマートコントラクトはそれ自体が状態機械である必要はなく、その場合、結果として生じるスマートコントラクトは状態値のフィールドを持たなくてよい。

【０３２７】

図30の破線矢印によって示されるように、アンロックトランザクション3030のインプットVin[0]は、前のトランザクション3020及びそのトランザクションのアウトプットVout[0]を指す。これは、前のトランザクション3020のTxID、及びアウトプットVout[0]を指すように"0"に設定された整数値の形式であり得る。Vout[0]のロックスクリプトは、現在状態("S3"）と、状態機械の遷移マトリクスとを含み、アンロックトランザクション3030のVout[x]が次の状態の状態機械("S4"）になるよう、及びアンロックトランザクション3030のVout[y]が新しいスマートコントラクト("NSC"）であるように制約し、場合によっては他の制約である。

【０３２８】

一例として、作業者創出プロセスの一例及びアンロックトランザクションに新たなスマートコントラクトを強制的に存在させるプロセスの疑似コードは、図31に示すことができる。

【０３２９】

スマートコントラクトの融合（merging）

2つ以上の状態を同じ次の状態に融合する必要がある場合に、状態の融合又は結合を使用することができる。場合によっては、スマートコントラクトで必要とされない場合、並行に動作している状態機械は同じ状態にあるが、融合しないことがある。例えば、図26の状態図では、S4とS7が別々にS8に遷移し、2つの別々のトランザクションのままであることが可能である。

【０３３０】

図32は、状態機械3210の融合動作を示す。この場合、アンロックトランザクション3230(インプットVin[0]及びVin[1]、アウトプットVout[x]を有する）は、前のトランザクション3220及び前のトランザクション3222の両方のVout[0]をアンロックする。これらの技術を用いて、スマートコントラクトは、それ自身の新しいインスタンスをインスタンス化するか、異なるスマートコントラクトをインスタンス化することによって分岐することができ、複数のスマートコントラクトは、同一であるか否かにかかわらず、融合されて、対応する状態機械を介して単一のスマートコントラクトを形成することができる。

【０３３１】

各々の融合するアンロックトランザクションは、1つのロックスクリプトを持つことができる。このロックスクリプトは、単一のスマートコントラクトの実行を継続する。これは、融合されるスマートコントラクトの1つであるか、まったく新しいスマートコントラクトであるかのいずれかである。いずれの場合も、複数のチェックが実行されてもよい。スマートコントラクトの各々は、アンロックトランザクションへの有効なインプットであるとしてチェックされ得る。これは、前のトランザクションを使用する検証によって実行

することができる。

【０３３２】

図32に示すように、Vin[0]は、前のトランザクション3220及びそのアウトプットVout[0]を示し、一方、Vin[1]は、前のトランザクション3222及びそのアウトプットVout[0]を示す。両方のVout[0]のロックスクリプトは、アンロックトランザクション3230のVout[x]を、次の状態S8及び状態遷移マトリクスを有するように制約する。これらの制約は、新しいスマートコントラクトがアンロックトランザクションのアウトプットであることをチェックする前のトランザクション3222のロックスクリプトによって強制され得る。

【０３３３】

並行状態機械でこれを達成するために、トランザクションは、結合される必要がある各状態機械のためのインプットを持つ複数のインプットを使用することができる。状態機械遷移マトリクス内で、結合されたすべての独立状態機械の状態をチェックするために、制約が追加され得る。結合状態機械の1つが収束可能な状態にない場合、トランザクションは拒否されてよい。結合操作を確実にするために、アンロックトランザクションの状態に基づいて、2つのアンロックスクリプトがあることを確認するために、状態機械に制約を追加することができる。各スクリプトに対して、各スクリプトが遷移に必要な正しい状態にあることを確認するために、チェックを実行することができる。本明細書の他の箇所で説明するように、相互依存ロック制約もまた、使用され得る。

【０３３４】

図33は、弱い依存関係を有するアンロックトランザクションにおけるスマートコントラクトを融合するプロセスの一例としての擬似コードシーケンスを示す。融合では、一連の現在状態(必ずしも一意の状態ではない）から共有された次の状態への遷移が起こる。アンロックトランザクションに関して、そのインプットのセットは、同じアウトプットが遷移マトリクスを複製し、同じ次の状態を埋め込むという制約を共有するロックスクリプトのセットをアンロックすることができる。弱い依存関係の場合、同じアウトプットに対して同じ次の状態を持つ複数の状態機械が存在してよい。

【０３３５】

図32及び図33による融合は、弱い依存関係で起こり得る。ロックスクリプトは、相互依存、枝分かれ、新しいスマートコントラクトの創出など、他の制約が含むことができる。弱い依存関係の例として、融合が偶然に起こる場合、ロックスクリプト1は、状態Aから状態X及び状態Yへの枝分かれであってもよく、ロックスクリプト2は、状態Bから状態Xへの通常の遷移であってもよい。これらの2つのロックスクリプトは、たまたま、アンロックトランザクションのアウトプットに状態Xが含まれるように制約する。

【０３３６】

図33の擬似コードシーケンスでは、その擬似コードシーケンスに従ってスクリプトを処理するプロセッサが、前のトランザクション、アンロックトランザクション、及び任意でいくつかのインプットデータを注入し、処理を進める前に終了条件をチェックしてよい。次に、プロセッサは、前のトランザクションロックスクリプト及びアンロックトランザクションロックスクリプトを抽出する。次に、プロセッサは、前のトランザクションロックスクリプト内のトランザクションマトリクスが、アンロックトランザクションロックスクリプト内のトランザクションマトリクスと等しいことをチェックし、もしそうでなければ、中止してもよい。そこから、プロセッサは、前のトランザクションの現在状態値を抽出し、それから及びトランザクションマトリクスとインプット(使用されている場合）から、次の状態を決定し、次の状態をチェックして、それがアンロックトランザクションロックスクリプト内の次の状態と一致するかどうかを決定する。任意で、他の制約をここでチェックできる。

【０３３７】

図34は、明示的な依存関係を有するアンロックトランザクションにおけるスマートコントラクトを融合するためのプロセスの例として、擬似コードシーケンスを示す。明示的な依存関係では、複数の状態機械があってよく、そしてそれらが同じアウトプットに対して

同じ次の状態を持つことでは十分ではなく、加えてそれらは固有の別個の状態機械であってよい。図32及び図34による融合は、明示的な依存関係によって生じ得る。強力な制約がある場合、アンロックトランザクションは、2つ以上の固有の融合状態機械を指すように要求されてよい。

【０３３８】

図34の擬似コードシーケンスでは、その擬似コードシーケンスに従ってスクリプトを処理するプロセッサが、前のトランザクション、アンロックトランザクション、及び任意でいくつかのインプットデータを注入し、処理を進める前に終了条件をチェックすることができる。次に、プロセッサは、アンロックトランザクションロックスクリプトを抽出する。次に、プロセッサは、一連のインプットインデックスを通じてループして、対応する前のトランザクションロックスクリプトを抽出し、対応する前のトランザクションロックスクリプト内の遷移マトリクスが、アンロックトランザクションロックスクリプト内の遷移マトリクスと等しいかどうかをチェックし、そうでない場合は中止する。そこから、プロセッサは、前のトランザクションの現在状態値を抽出し、それを現在状態のリストに追加し、他のインプットについて繰り返す。現在状態、及び遷移マトリクスとインプット(使用されている場合）から、プロセッサは次の状態を決定し、次の状態をチェックして、アンロックトランザクションロックスクリプト内の次の状態に一致するかどうかを決定する。任意で、他の制約をここでチェックできる。

【０３３９】

並列スマートコントラクト/バリア

前述したように、弱い又は強い制約により、2つ以上のスマートコントラクトを1つのスマートコントラクトに融合することができる。また、並列に実行できる2つ以上のスマートコントラクトを持つことも有用である。これは、2つの並行パスが1つのアンロックトランザクションを通過しなければならないという制約があるという点で、並行処理を上回るものであってよい。これはまた、並行状態機械内の「バリア（barrier）」のために使用することができ、特定の遷移は、状態機械内の他の場所で別の遷移が発生するまで発生することは許されない。

【０３４０】

図35は、並列処理動作を示す。この場合、アンロックトランザクション3530は、前のトランザクション3520及び前のトランザクション3530の各々に1つの、2つのインプット(Vin[0]、Vin[1]）、及びそれぞれ別々のスマートコントラクトを継続するための2つのアウトプット(Vout[x]、Vout[y]）を有する。アンロックトランザクション3530は、スマートコントラクトの前のトランザクションの各々を含むアンロックスクリプトを含む。さらに、各スマートコントラクトのためのロックスクリプトが、その実行を継続するために要求されることがある。並列スマートコントラクトを使用すると、各コントラクトを独立して遷移できないが、代わりに(ロックスクリプト内で定義されている）すべてのコントラクトを並行に行う必要がある。

【０３４１】

先の例は、スマートコントラクトを枝分かれし、2つのスマートコントラクトを融合し、2つの(又はそれ以上のスマートコントラクトの）並列動作を提供した。これらは、並列スマートコントラクトと組み合わせて使用することができる。バリアは、状態遷移が他の状態遷移と論理的な関係にある場合に使用されてもよい。一例として、それは、他の状態機械が特定の状態にあることを要求するか、又はその状態機械からのインプットデータを要求してよい。

【０３４２】

この機能を容易にするために、複数のインプットとアウトプットを含むアンロックトランザクションを作成することができる。バリアに必要な状態機械(遷移するもの、又は制約に関係するもの）のそれぞれは、それら自身のアンロックスクリプトに含まれてよい。遷移を実行する各状態機械に対して、ロックスクリプトが使用されてもよい。図26に示す例では、状態S9からS10への遷移は許容されるが、S6からS7への遷移が並行に起こる場合

にのみ許容される。アンロックトランザクションの場合、状態S6の状態機械のために1つ、及び状態S9の状態機械のために1つの、2つのアンロックスクリプトが存在する。次に、アンロックトランザクションは、2つのロックスクリプトを持つ。1つはS7のままである状態機械用であり、もう1つは現在S10にある状態機械用である。

【０３４３】

図36は、並列トランザクションインプット／アウトプットを必要とする並行状態機械におけるバリアを使用するための例としての擬似コードシーケンスである。この例では、疑似コードシーケンスに従ったスクリプトは、前のトランザクションの各々がアンロックトランザクション内のロックスクリプトと同一のロックスクリプトを持っていることをチェックする。この例では、第1の前のトランザクションに一致するように1つのロックスクリプトが設定されている、等である。次に、その擬似コードシーケンスによるスクリプトは、両方の前のトランザクションから現在状態を抽出し、両方の状態機械のインプットを読み取り、両方の状態遷移に設定されたバリアをチェックする。

【０３４４】

図35に戻ると、種々の依存関係が、破線矢印及び番号付きの円で示されている。依存関係#1は、Vin[0]が前のトランザクション3520を参照しなければならないこと、依存関係#2は、Vin[0]が指定トランザクション(前のトランザクション3520）の第1アウトプット(Vout[0]）を参照しなければならないこと、依存関係#3は、Vin[1]が前のトランザクション3522を参照しなければならないこと、及び依存関係#4は、Vin[1]が指定トランザクション(前のトランザクション3522）の第1アウトプット(Vout[0]）を参照しなければならないことであってよい。これは、単に2つの依存関係、つまり、Vin[0]がトランザクション3520のVout[0]を参照しなければならない依存関係と、Vin[1]がトランザクション3522のVout[0]を参照しなければならない依存関係、であってよい。

【０３４５】

前のトランザクション3520のVout[0]のロックスクリプトは、Vout[x]が状態=S8を有し、状態機械1の遷移マトリクスを有することを要求する依存関係#5を課し、Vout[y]が状態=S8を有し、状態機械2の遷移マトリクスを有することを要求する依存関係#6を課す。前のトランザクション3522のVout[0]のロックスクリプトは、同様の方法で、依存関係#7及び#8を課す。いくつかの例では、状態機械の制約の代わりに、スマートコントラクトの他の形式の制約であってよい。2つ(又はそれ以上）の前のトランザクションのアウトプットは、2つ(又はそれ以上）のアンロックトランザクションアウトプットを制約するので、並列スマートコントラクトを提供することができる。

【０３４６】

バリアは、融合とは異なってよい。バリアとは、一連の現在状態(必ずしも一意の状態ではない）からの遷移が並行に起こるように制約される場合である。アンロックトランザクションに関して、そのインプットのセットは、相互に依存するロックスクリプトのセットをアンロックすることであってよく、それらの全ては、アウトプットのセットにそれらのそれぞれの制約を適用してよい。相互依存ロックスクリプトは、要件及び共有される次の状態のセットであってよいが、可能な場合には、要求されなくてよい。相互依存ロックスクリプトは、特定の遷移がこの相互依存関係を必要とするように、遷移マトリクスに埋め込むことができる。

【０３４７】

図37は、バリア/並列スマートコントラクトの使用例の状態図を示す。AliceとBobがダンスを学んでおり、ダンスのレッスンに行くとする。ダンス学生の成長を維持する方法として、彼らの成長は状態機械の使用により記録される。第一に、等級の間に状態機械が存在してよく、すべてのダンスのタイプの全体的なダンス能力に関して状態が含まれている。これは、状態S1、S3、S5、最後にS6で表される。これらの等級は、学生がダンス部によりサポートされる最高の等級に達するまで、順次進む。次の等級(例えば、S1～S3）に進むためには、学生は、いくつかのタイプのダンスに到達する必要があってよい。各ダンスのタイプは、より小さな埋め込み状態機械(例えば、S2及びS4）を使用して監視されてよ

い。この使用例では、以下の操作が可能である。

【０３４８】

（1）枝分かれ（fork）、及びクローン化（clone）。これは、点S1及びS5で使用され、新しい等級の開始を示すことができ、それによって、各ダンスタイプに1つずつ、複数の状態機械への枝分かれが存在してよい。

【０３４９】

（2）作業者創出(worker spawning）。これは、特定のダンスの種類がより多くのレッスンを必要とする場合、又は他のダンスタイプとは異なる評価が提供される場合に使用することができる。

【０３５０】

（3）融合（merge）。このプロセスは、学生が新しい等級に達したときに、点S3及びS6で使用することができる。新しい等級では、埋め込まれたダンス状態機械の最終状態S4で各生徒を評価する必要がある。

【０３５１】

（4）並列トランザクション（Parallel Transaction）/バリア（barrier）。ある状態機械内の状態遷移を別の状態機械で制約するために使用することができる。この使用例の場合、S2とS4の遷移は並行に起こる必要があると仮定することができる。なぜなら、1回のレッスンで1つのダンスタイプを実行することはできないからである。

【０３５２】

したがって、本明細書及び図面は、制限的な意味ではなく例示的な意味に考えられるべきである。しかし、特許請求の範囲に記載された本発明の範囲から逸脱することなく、様々な修正及び変更をそれに加えることができることは明らかであろう。同様に、他の変形も本開示の範囲内である。従って、開示された技術は、種々の修正及び代替の構成を容易に受けるが、それらのいくつかの例示された実施形態は、図面に示され、上記で詳細に説明された。しかしながら、本発明を開示された特定の形態に限定する意図はなく、逆に、添付の特許請求の範囲に定義されているように、本発明の範囲内にあるすべての変更、代替の構成及び均等物をカバーすることが意図されていることは理解されるべきである。

【０３５３】

「a」及び「an」並びに「the」及び類似の参考の用語の、開示された実施形態を説明する文脈における使用は(特に、以下の特許請求の範囲の文脈において）、別段の指示又は明白な矛盾がない限り、単数形及び複数形の両方を含むものと解釈する。「construction」、「having」、「include」及び「include」という用語は、別段の指示がない限り、オープンエンドの用語(すなわち、「含むが、これに限定されない」を意味する）と解釈する。「connected」という用語は、修正されず物理的接続を指す場合、何かの介在がある場合であっても、部分的又は全体的に、内部に含まれ、取り付けられ、又は連結されているものと解釈する。本開示における値の範囲の記載は、特に指示がない限り、個々の値が個々に記載されているかのように、個々の値が範囲内に含まれる個々の値を個々に参照する簡単な方法として機能することを意図するに過ぎない。「セット」(例えば、「アイテムのセット」）又は「サブセット」という用語は、文脈によって別段の注記又は矛盾がない限り、1以上の構成要素からなる空でない集合として解釈されるべきであり、さらに、文脈によって別段の注記又は矛盾がない限り、対応するセットの用語「サブセット」は、必ずしも対応するセットの適切なサブセットを示すものではないが、サブセット及び対応するセットは等しくてもよい。

【０３５４】

「A、B、及びCの少なくとも1つ」又は「A、B、又はCの少なくとも1つ」の形式のフレーズなどの結合語は、特に別段の記載がない限り、又は文脈によって明確に矛盾しない限り、一般的に使用される文脈と理解され、アイテム、用語などは、A、B、又はC、又はA、B、及びCのセットのいずれかの非空のサブセットであり得ることを示す。例えば、3つの構成要素を有するセットの説明のための例において、「A、B、及びCの少なくとも1つ」及び「A、B、又はCの少なくとも1つ」とは、以下のセットのいずれかを指す:{A}、{B}、{C}、

{A}、{A、B}、{A}、{C}、{A、{B、C}}。従って、そのような結合語は、一般に、特定の実施形態が、各々が存在するために、Aの少なくとも1つ、Bの少なくとも1つ、及びCの少なくとも1つを必要とすることを意味するようには意図されない。

【０３５５】

記載されたプロセスの操作は、他に指示がない限り、又は文脈によって明らかに矛盾しない限り、任意の適切な順序で実行することができる。記載されるプロセス(又はその変形及び／又は組み合わせ）は、実行可能な命令を備えて構成される1つ以上のコンピュータシステムの制御下で実行することができ、ハードウェア又はそれらの組み合わせによって、1つ以上のプロセッサ上で集合的に実行するコード(例えば、実行可能な命令、1つ以上のコンピュータプログラム又は1つ以上のアプリケーション）として実装することができる。コードは、例えば、1つ以上のプロセッサによって実行可能な複数の命令を含むコンピュータプログラムの形式で、コンピュータ読取可能な記憶媒体に記憶することができる。コンピュータ読取可能記憶媒体は、非一時的であってもよい。

【０３５６】

提供されるあらゆる実施例、又は例示的言語(例えば、「のような」）の使用は、本発明の実施形態をより良く解明することのみを意図したものであり、他に請求されない限り、本発明の範囲を制限するものではない。本明細書中の言葉使いは、主張されていない如何なる要素も本発明の実施に不可欠であることを指示していると解釈されてはならない。

【０３５７】

本発明を実施するために発明者に知られている最良の形態を含め、本開示の実施形態が記載される。これらの実施形態の変形例は、前述の説明を読むことにより、当業者に明らかになり得る。本発明者らは、当業者がそのような変形を適宜使用することを期待しており、本発明者らは、本開示の実施形態が、具体的に記載された以外の方法で実施されることを意図している。従って、本開示の範囲は、適用法によって許容されるように、本明細書に添付された特許請求の範囲に記載された主題のすべての修正及び均等物を含む。さらに、上述の要素のあらゆる可能な変形における任意の組み合わせは、別段の指示がない限り、又は文脈によって明らかに矛盾しない限り、本開示の範囲に包含される。

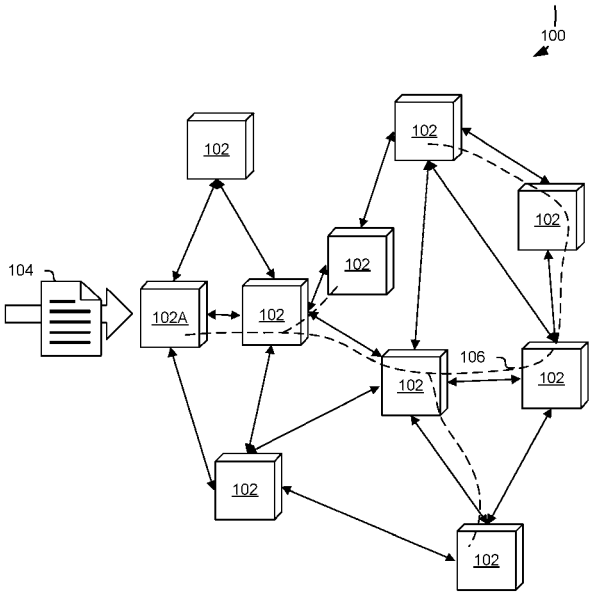
【０３５８】

引用された刊行物、特許出願、及び特許を含む全ての参考文献は、各参考文献が個別にかつ具体的に参考文献として援用され、全体として記載されているのと同じ程度に、参考文献として本明細書に組み込まれる。

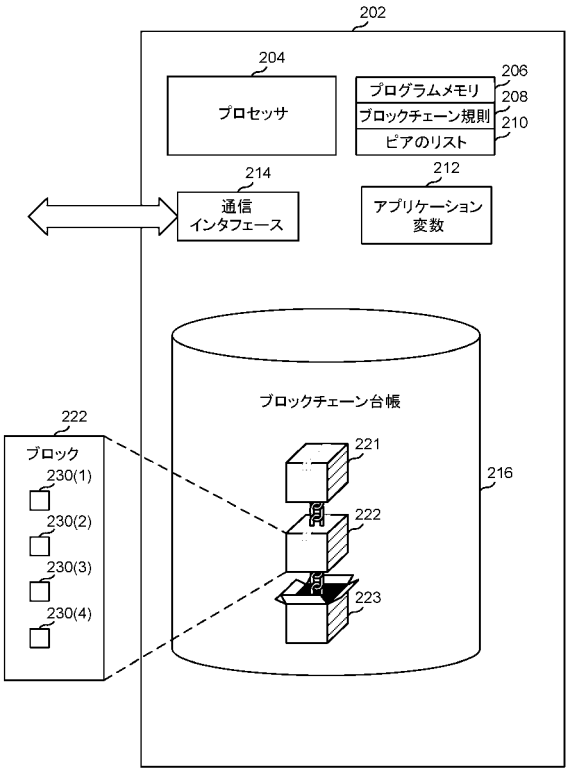
【０３５９】

上述の実施形態は、本発明を限定するものではなく、本発明を例示するものであり、当業者は、添付の特許請求の範囲によって定義されるように、本発明の範囲から逸脱することなく、多くの代替実施形態を設計することができることに留意されたい。特許請求の範囲においては、括弧内に付した引用符号は、クレームを限定するものと解釈してはならない。「含む（comprising、comprises等）」及び類似の語は、いずれかの特許請求の範囲又は明細書全体に列挙されたもの以外の要素又はステップの存在を除外しない。本明細書において、「有する」、「含む」（comprising、comprises）は「有する」、「含む」、「から成る」（including、includes、consisting of）を意味する。要素の単数形は、そのような要素の複数形を除外するものではなく、その逆もまた同様である。本発明は、いくつかの別個の要素を含むハードウェアによって、及び適切にプログラムされたコンピュータによって実施することができる。幾つかの手段を列挙する装置クレームにおいては、これらの手段の幾つかは、一つの同一のハードウェアアイテムによって具体化することができる。特定の措置が相互に異なる従属クレームに記載されているという単なる事実は、これらの措置の組み合わせが有利に利用できないことを示すものではない。

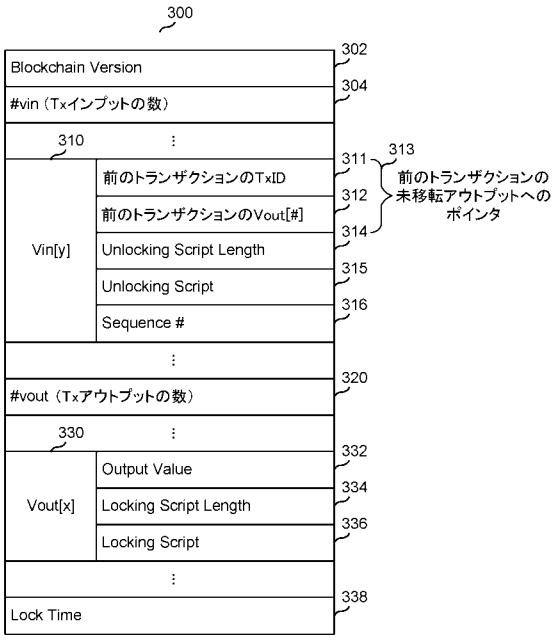
【図１】


100
102
102
102
102
102
104
102A
102
106
102
102
102

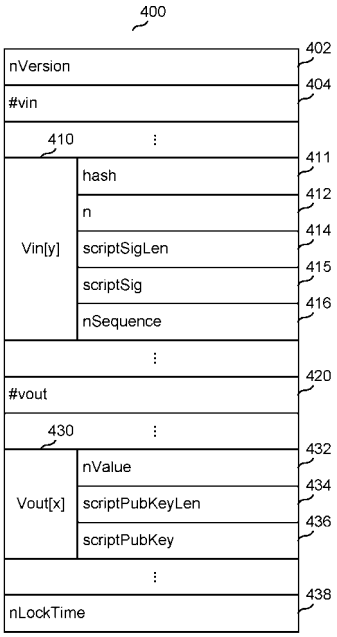
【図２】


202
204
プロセッサ
206
プログラムメモリ
208
ブロックチェーン規則
210
ピアのリスト
214
通信
インタフェース
212
アプリケーション
変数
ブロックチェーン台帳
222
ブロック
230(1)
230(2)
230(3)
230(4)
221
222
223
216

【図３】


300
Blockchain Version
302
#vin（Txインプットの数）
304
310
311
313
前のトランザクションのTxID
312
前のトランザクションのVout[#]
前のトランザクションの
未移転アウトプットへの
ポインタ
Vin[y]
314
Unlocking Script Length
315
Unlocking Script
316
Sequence #
320
#vout（Txアウトプットの数）
330
332
Output Value
334
Locking Script Length
Vout[x]
336
Locking Script
338
Lock Time

【図４】


400
402
nVersion
404
#vin
410
411
hash
412
n
414
scriptSigLen
Vin[y]
415
scriptSig
416
nSequence
420
#vout
430
432
nValue
434
scriptPubKeyLen
Vout[x]
436
scriptPubKey
438
nLockTime

【図５】

OP_GENSIG Script

インプット

```
<SIGHASH Byte><m><a><k>
```

アウトプット

```
(1) // K = k x G
    OP_DUP OP_TOALTSTACK <PubK G> OP_ECPMULT

(2) // r = x mod n
    OP_ ECPX <N> OP_BIGMOD OP_DUP OP_TOALTSTACK

    // k-1(r x d + m) mod n
(3) <N> OP_BIGMODMUL <N> OP_BIGMODADD OP_FROMALTSTACK OP_SWAP
    OP_FROMALTSTACK <N> OP_BIGMODINVERSE <N> OP_BIGMODMUL

    // DER encode (r,s) and append SIGHASH Type
(4) OP_DERENCODE OP_SWAP OP_CAT
```

【図６】

OP_PREVTXINJECTION Script

インプット

```
<Serialized Previous Tx (for Tx ID X)>
<SIGHASH Type>
<Serialized Set of Unlocking Tx Fields>
```

アウトプット

```
<X>

// OP_PREVTXINJECTION Below
// Push <X> to Alt Stack
OP_TOALTSTACK

// Causes Injection of Unlocking Tx (consumes SIGHASH Type)
OP_SWAP OP_OVER OP_SPENDINGTXINJECTION OP_VERIFY

// Extract Tx ID <X> from Unlocking Tx
OP_FROMALTSTACK OP_EXTRACTTXID

// Create Tx ID from Serialized Previous Tx
OP_SWAP OP_HASH256

// Check the Tx IDs are the same
OP_EQUAL
```

【図７】

OP_SELFTXINJECTION Script

インプット

```
<Serialized Previous Tx (for input being signed)>
<SIGHASH Type>
<Serialized Set of Unlocking Tx Fields>
```

アウトプット

```
// Save a copy of SIGHASH Type
OP_OVER OP_TOALTSTACK

// Only 1 Tx ID, which corresponds to Prev Tx of
// this Locking Script
<0> OP_PREVTXINJECTION OP_VERIFY

// Make sure SIGHASH Type is SIGHASH_X | ANYONECANPAY
OP_FROMALTSTACK <SIGHASH_ ANYONECANPAY> OP_AND OP_OVER
<SIGHASH_ANYONECANPAY> OP_AND
```

【図８】

OP_SPENDINGTXINJECTION Script

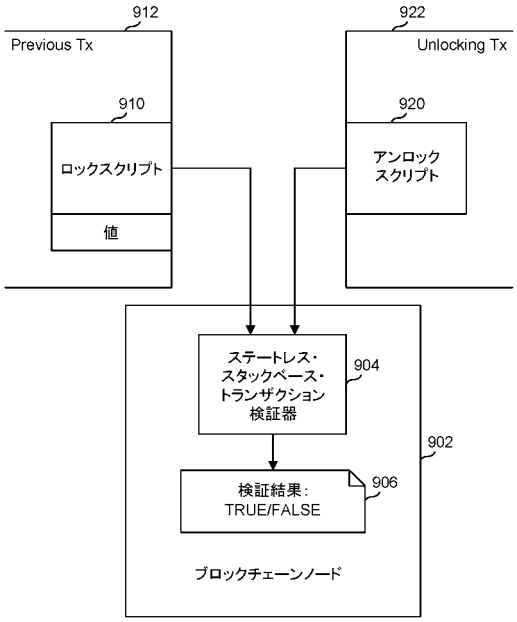
インプット

```
<SIGHASH Type>
<Serialized Set of Unlocking Tx Fields>
```

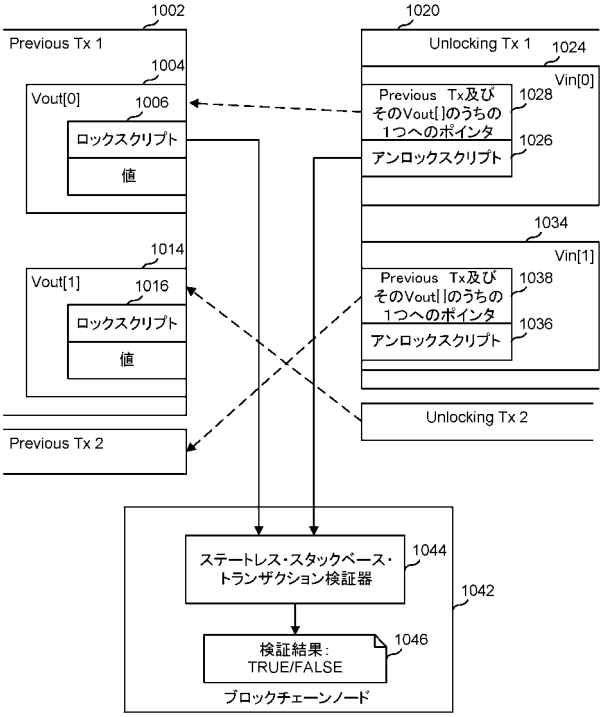
アウトプット

```
// Double SHA256 the Message
OP_HASH256

// Generate a Signature using SIGHASH Type, Message,
// Private Key and k
<a><k> OP_GENSIG

// Perform Signature Verification
<PubK A> OP_CHECKSIG
```

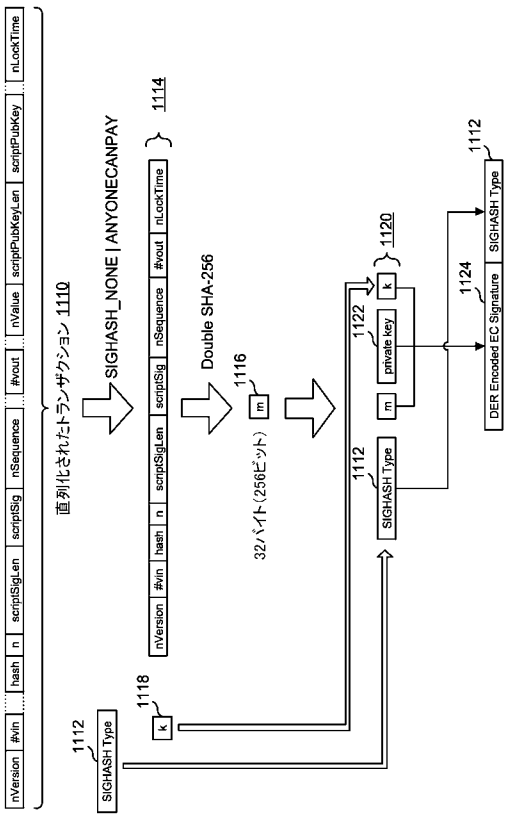
【図９】



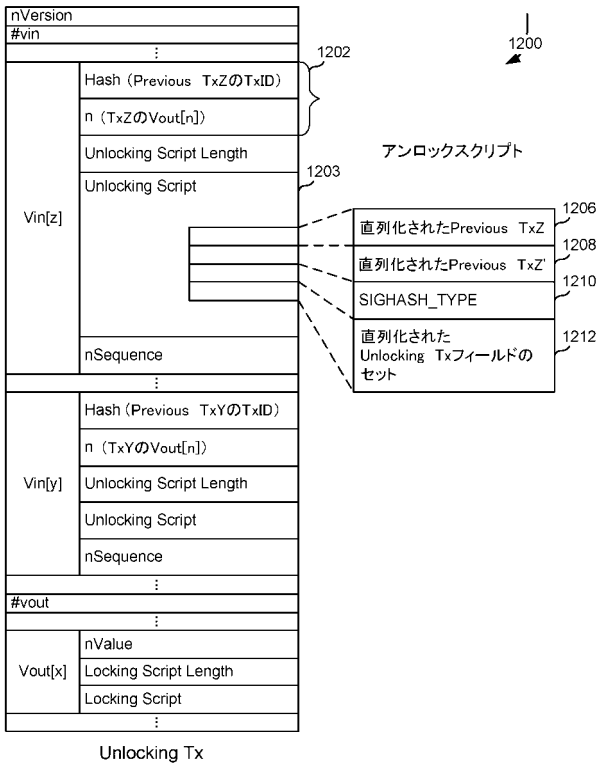
【図１０】



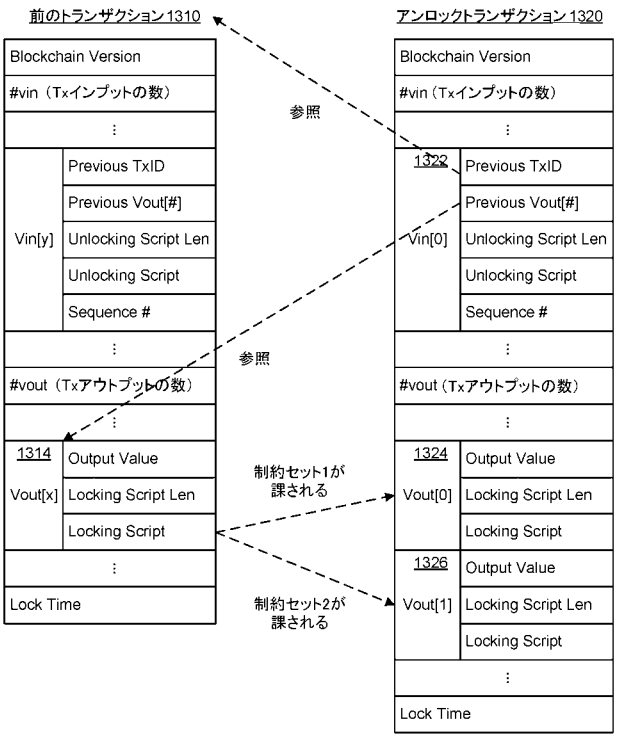
【図１１】



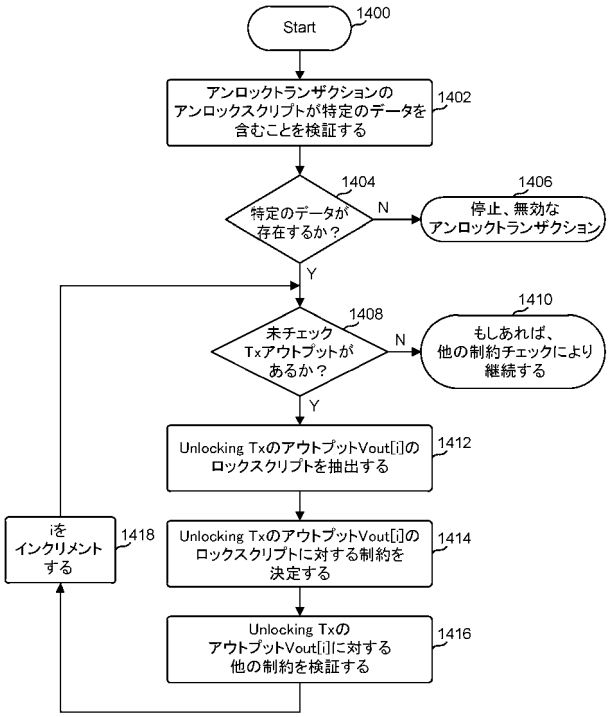
【図１２】



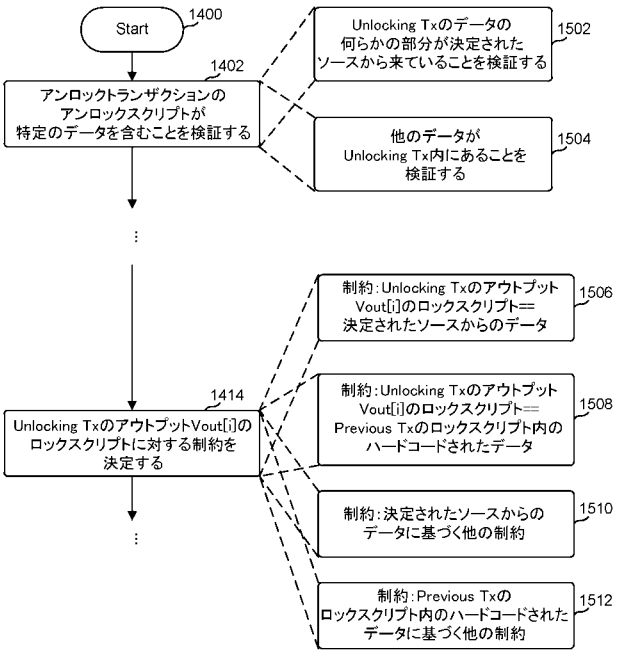
【図１３】



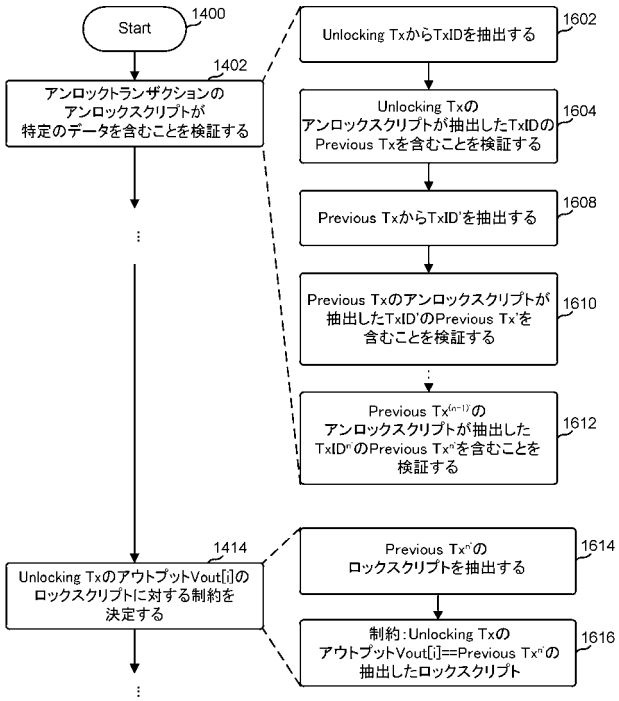
【図１４】



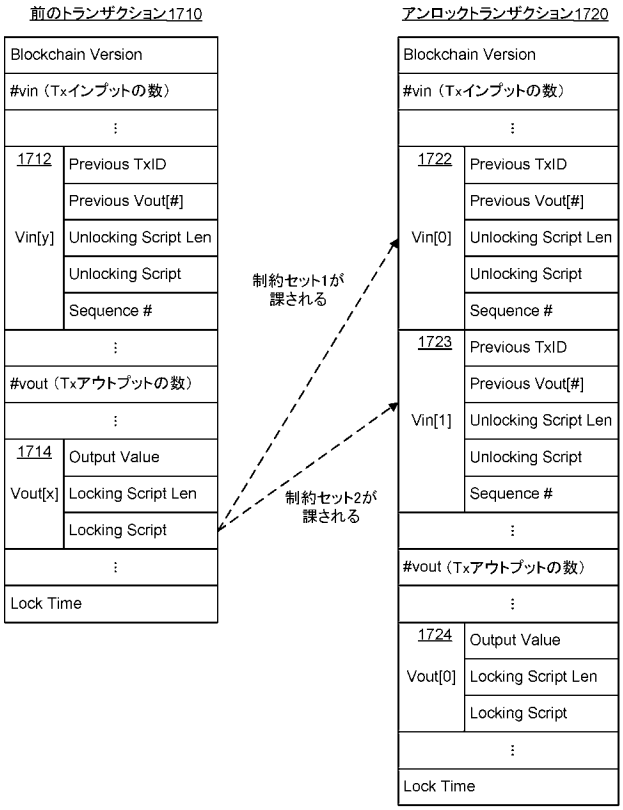
【図１５】



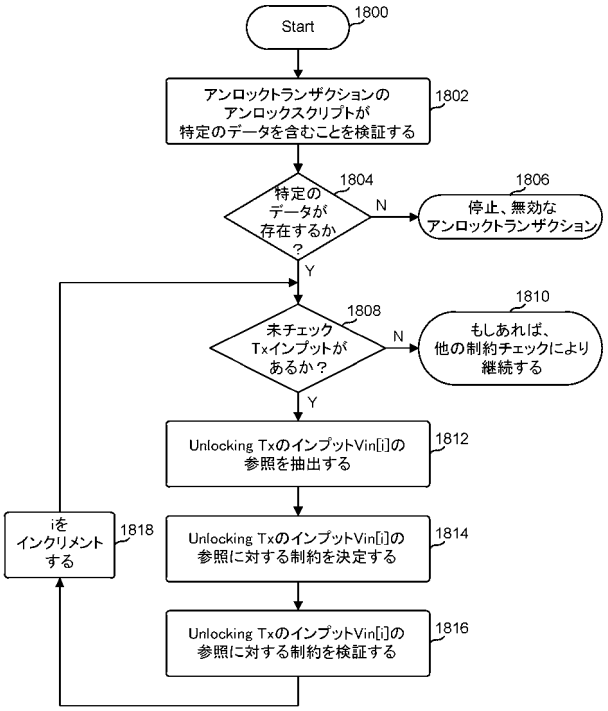
【図１６】



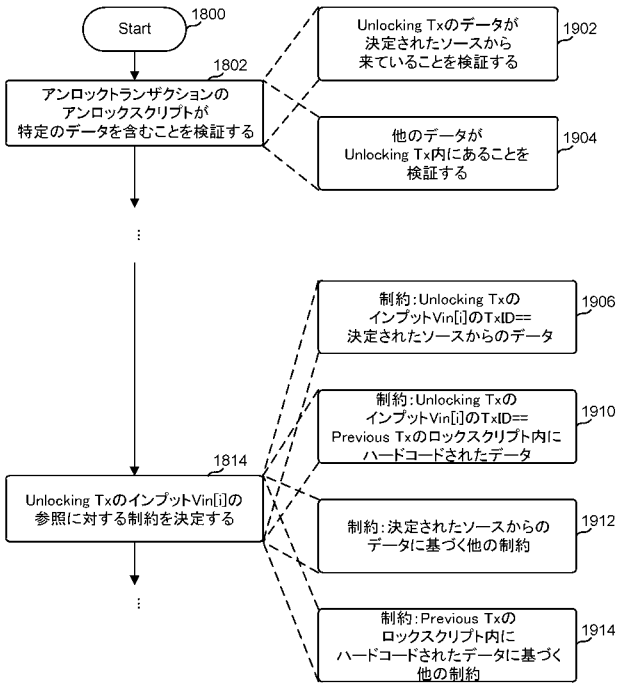
【図１７】



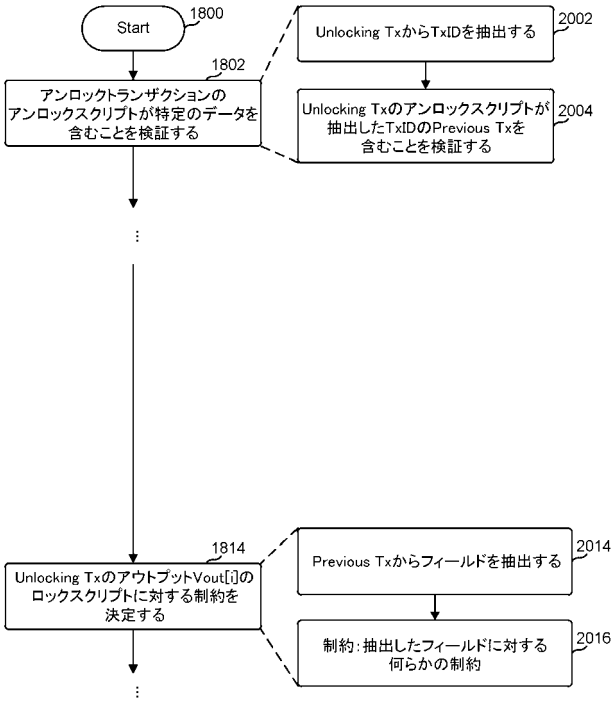
【図１８】



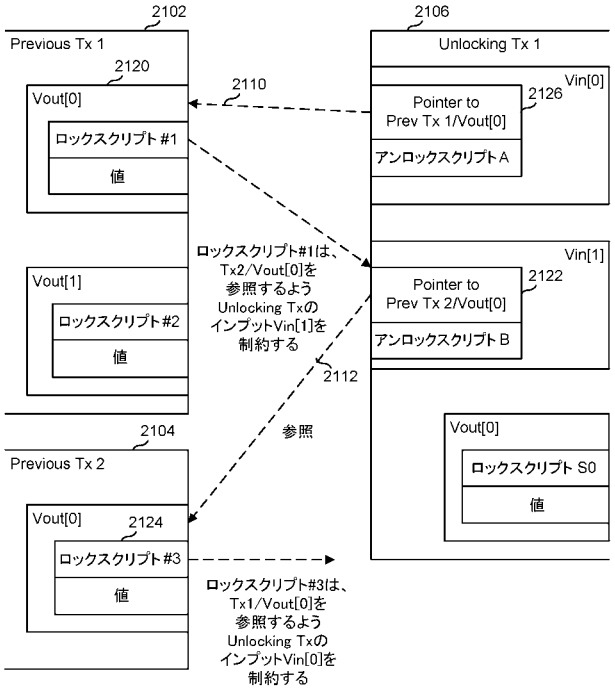
【図１９】



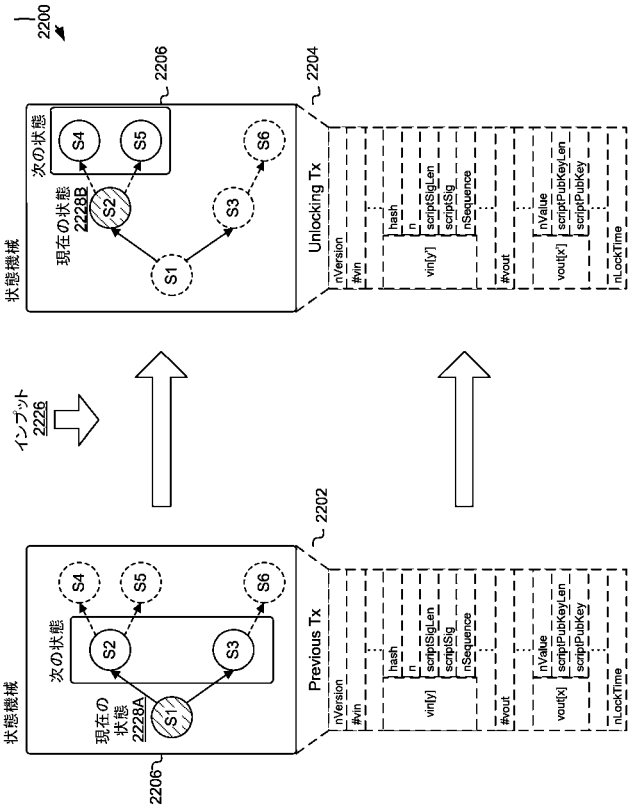
【図２０】



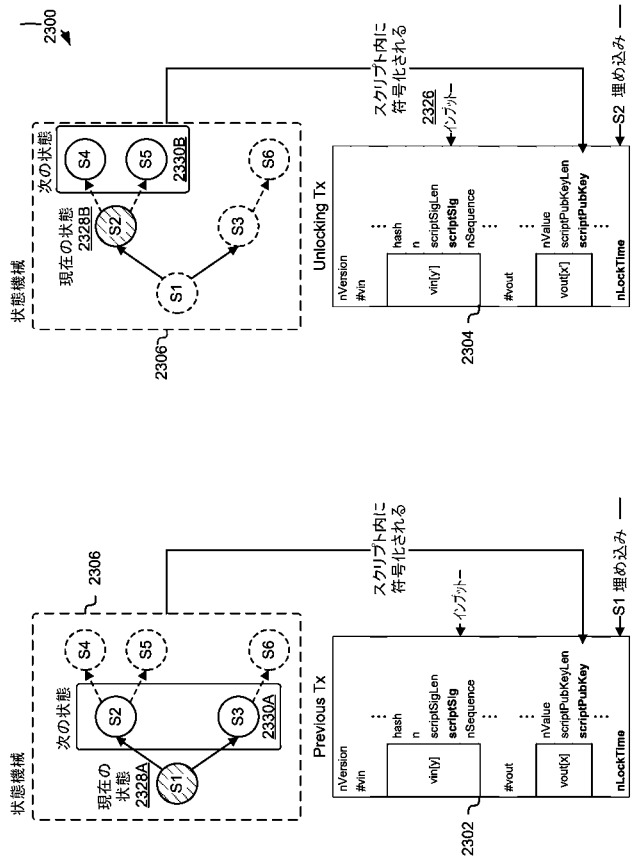
【図２１】



【図２２】



【図２３】



【図２４】

トラストレス決定性状態機械スクリプト

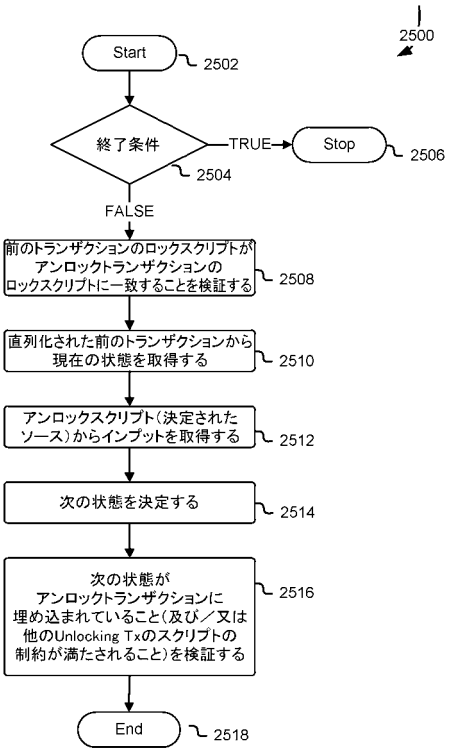
インプット／アンロックスクリプト

```
<External Sig'><External Data>
<Serialized Previous Serialized Tx>
<SIGHASH Type>
<Serialized Set of Unlocking Tx Fields>
```

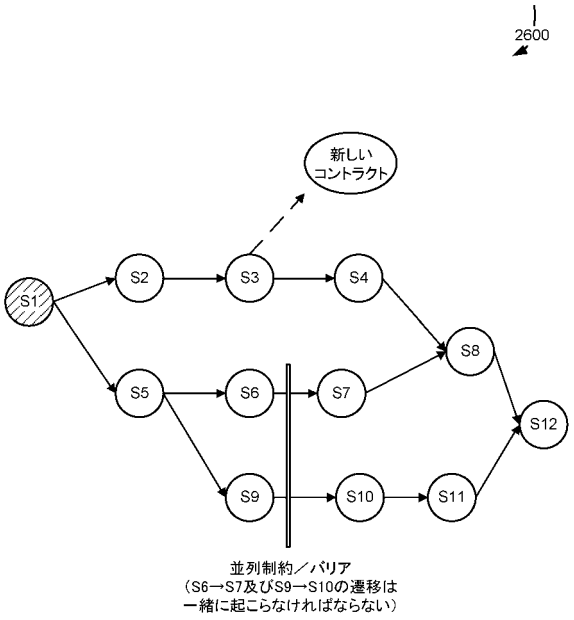
アウトプット／ロックスクリプト

```
<Script to Check Termination Condition>

// If not terminated
OP_NOTIF

// Check Self-replicating Smart Contract
// Extract Locking Script from Unlocking Transaction
<Script to Extract Locking Script>

// Extract Locking Script from Previous Transaction
<Script to Extract Locking Script>

// Check the Locking Scripts are the same
OP_EQUALVERIFY

// Extract Current State
<Script to Extract Data from Serialized Previous Tx>

// Read Input
<Script to Read Input from unlocking script>

// Verify State-transition matrix constraint
<Script to Determine State Transition>
<Script to Verify Constraint on Unlocking Tx>
OP_VERIFY
OP_ENDIF

// Check that Previous Tx, SIGHASH Type, and Unlocking Tx
// have been injected
OP_SELFTXINJECTION
```

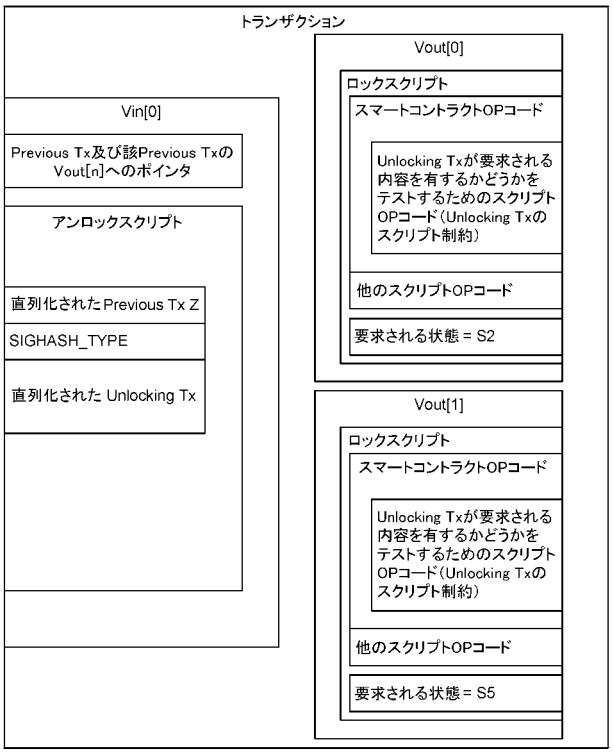
【図２５】



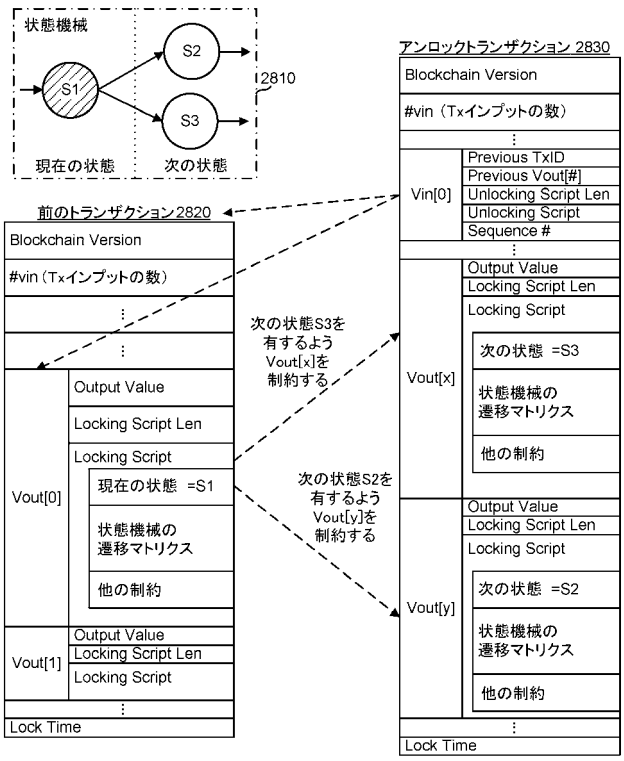
【図２６】



【図２７】



【図２８】

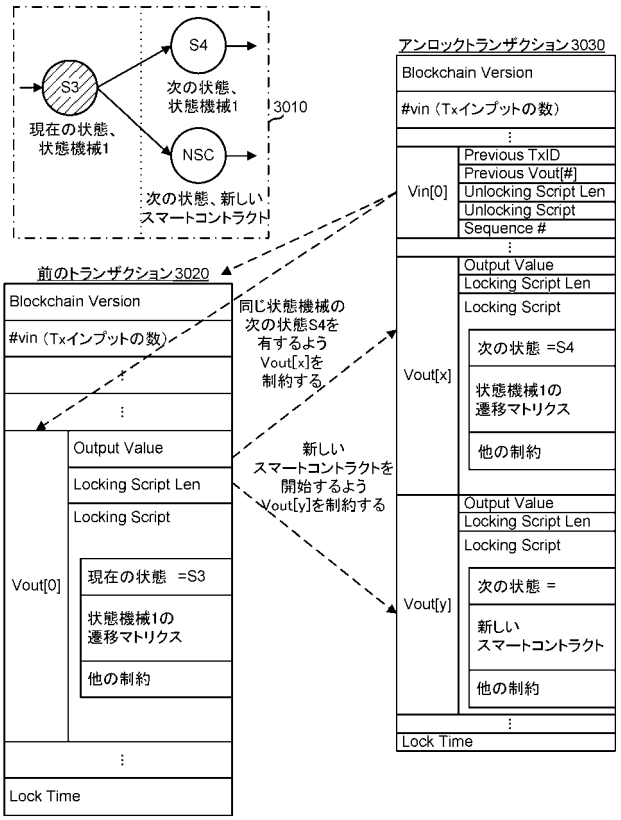


【図２９】

枝分かれ／クローン化プロセスの状態機械スクリプト

```
1. Cause injection of Previous Tx, Unlocking Tx, and Input (optional)

2. Check Termination Condition
      2a. If true, goto step 6

3. Extract Previous Tx Locking Script

4. Extract Previous Tx Current State

5. Loop through set of Outputs in Unlocking Tx
      5a. Extract Locking Script from Output

      5b. Check Transition Matrix in Previous Tx Locking Script ==
Transition Matrix in Locking Script
            5bi. If false, script fails

      5c. Determine Next State from Current State, Transition Matrix, Input
(optional), and Output Index

      5d. Check Next State == Next State in Output
            5di. If false, script fails

6. Script passes (optionally, other constraints can be checked here)
```

【図３０】

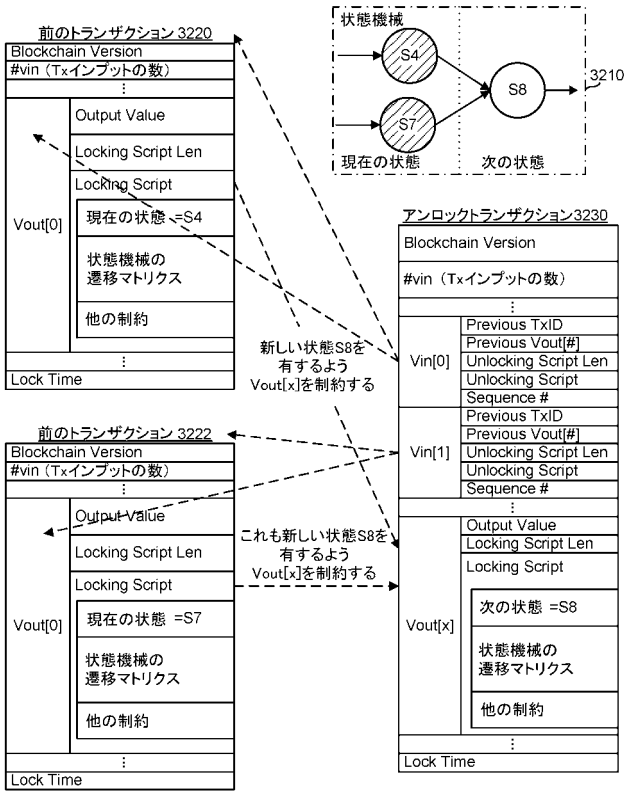


【図３１】

アンロックトランザクションの中で新しいスマートコントラクトの創出を強制するスクリプト

```
1. Cause injection of Previous Tx, Unlocking Tx, and Input (optional)

2. Check Termination Condition
      2a. If true, goto step 12

3. Extract Previous Tx Locking Script

4. Extract Unlocking Tx Locking Script 1 (Input 1. State Machine)

5. Check Transition Matrix in Previous Tx Locking Script == Transition
Matrix in Unlocking Tx Locking Script 1
      5a. If false, script fails

6. Extract Current State from Previous Tx Locking Script

7. Determine Next State from Current State, Transition Matrix and Input
(optional)

8. Check Next State == Next State in Unlockding Tx Locking Script 1
      8a. If false, script fails

9. Extract Unlocking Tx Locking Script 2 (Input 2. New Smart Contract)

10. Determine Constraints on New Smart Contract

11. Check Constraints on New Smart Contract
      11a. If false, script fails.

12. Script passes (optionally, other constraints can be checked here)
```

【図３２】



【図３３】

融合プロセスのスクリプトー弱い依存関係

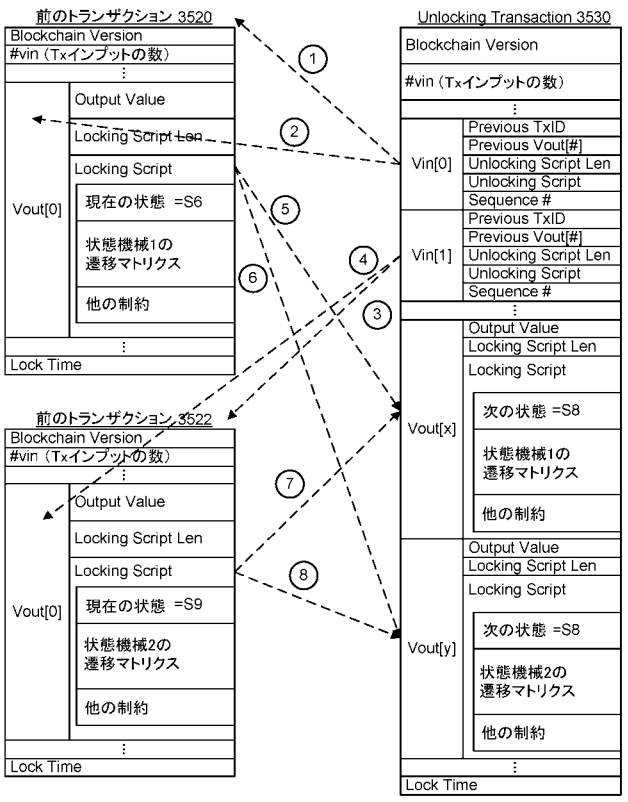
```
==============================
Merge State Machines - Weak Dependency (Case of multiple state machines that
so happen to have the same next state for the same output)
==============================

1. Cause injection of Previous Tx, Unlocking Tx, and Input (optional)

2. Check Termination Condition
      2a. If true, goto step 9

3. Extract Previous Tx Locking Script

4. Extract Unlocking Tx Locking Script

5. Check Transition Matrix in Previous Tx Locking Script == Transition
Matrix in Unlocking Tx Locking Script
      5a. If false, script fails

6. Extract Previous Tx Current State

7. Determine Next State from Current State, Transition Matrix and Input
(optional)

8. Check Next State == Next State in Unlocking Tx Locking Script
      8a. If false, script fails

9. Script passes (optionally, other constraints can be checked here)
```

【図３４】

融合プロセスのスクリプトー強い依存関係

```
==============================
Merge State Machines - Strong Dependency (Cases of different statemachine)
==============================

1. Cause injection of Previous Txs, Unlocking Tx, and Input (optional)

2. Check Termination Condition
      2a. If true, goto step 7

3. Extract Unlocking Tx Locking Script

4. Loop through set of Input Indexes in Unlocking Tx

      4a. Extract Previous Tx Locking Script (Corresponding Input)

      4b. Check Transition Matrix in Previous Tx Locking Script  ==
Transition Matrix in Unlocking Tx Locking Script
            4bi. If false, script fails

      4c. Extract Previous Tx Current State and append to list of Current
States

5. Determine Next State from Current States, Transition Matrix and Input
(optional)

6. Check Next State == Next State in Unlocking Tx Locking Script
      6a. If false, script fails

7. Script passes (optionally, other constraints can be checked here)
```

【図３５】

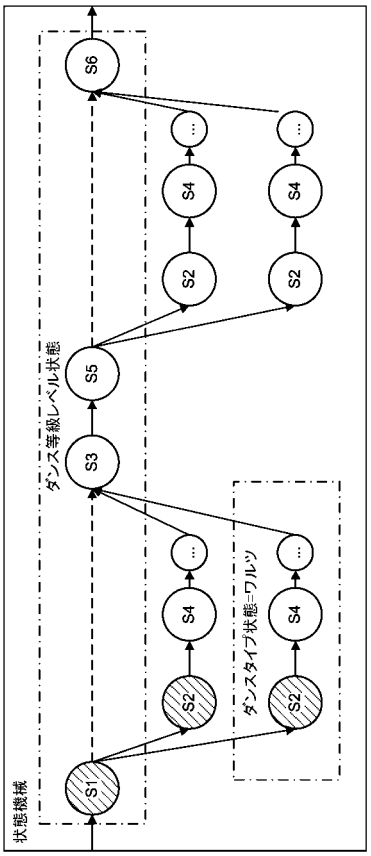


【図３６】

並列実行／バリアプロセスのスクリプト

```
1. Cause injection of Previous Txs, Unlocking Tx, and Input (optional)

2. Check Termination Condition
      2a. If true, goto step 5

3. Loop through set of Output/Input Index Pairs in Unlocking Tx

      3a. Extract Unlocking Tx Locking Script (Corresponding Output)

      3b. Extract Previous Tx Locking Script (Corresponding Input)

      3c. Check Transition Matrix in Unlocking Tx Locking Script  ==
Transition Matrix in Previous Tx Locking Script
            3ci. If false, script fails

      3d. Extract Previous Tx Current State and append to list of Current
States

4. Loop through set of Output Indexes in Unlocking Tx

      4a. Extract Unlocking Tx Locking Script (Corresponding Output)

      4b. Determine Next State from Current States, Transition Matrix, Input
(optional), and Output Index

      4c. Check Next State == Next State in Unlocking Tx Locking Script
            4ci. If false, script fails

5. Script passes (optionally, other constraints can be checked here)
```

【図３７】



【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No
PCT/IB2018/056431

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06Q20/02 G06Q20/06 G06Q20/40
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06Q

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	MÖSER MALTE ET AL: "Bitcoin Covenants", 31 August 2016 (2016-08-31), MEDICAL IMAGE COMPUTING AND COMPUTER-ASSISTED INTERVENTION - MICCAI 2015 : 18TH INTERNATIONAL CONFERENCE, MUNICH, GERMANY, OCTOBER 5-9, 2015; PROCEEDINGS; [LECTURE NOTES IN COMPUTER SCIENCE; LECT.NOTES COMPUTER], SPRINGER INTERNATIONAL PUBLISHING, CH, XP047354488, ISSN: 0302-9743 ISBN: 978-3-319-24946-9 [retrieved on 2016-08-31] the whole document ----- -/--	1-15

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
6 November 2018	15/11/2018

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Raymaekers, Jens

3

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/IB2018/056431

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	Andreas M. Antonopoulos: "Mastering Bitcoin - Unlocking Digital Cryptocurrencies" In: "Mastering bitcoin : [unlocking digital cryptocurrencies]", 20 December 2014 (2014-12-20), O'Reilly Media, Beijing Cambridge Farnham Köln Sebastopol Tokyo, XP055306939, ISBN: 978-1-4493-7404-4 page 111 - page 138	1-15
A	Bitfury Group: "Smart Contracts on Bitcoin Blockchain", , 4 September 2015 (2015-09-04), XP055382678, Retrieved from the Internet: URL:http://bitfury.com/content/5-white-papers-research/contracts-1.1.1.pdf [retrieved on 2017-06-19] the whole document	1-15
A	Anonymous: "Smart Contracts/EVM FAQ ¦ Counterparty", , 19 July 2017 (2017-07-19), XP055513060, Retrieved from the Internet: URL:https://web.archive.org/web/20170719092538/https://counterparty.io/docs/faq-smartcontracts/ [retrieved on 2018-10-05] the whole document	1-15

3

Form PCT/ISA/210 (continuation of second sheet) (April 2005)

フロントページの続き

(31)優先権主張番号 1713790.2
(32)優先日 平成29年8月29日(2017.8.29)
(33)優先権主張国・地域又は機関
英国(GB)

(81)指定国・地域 AP(BW,GH,GM,KE,LR,LS,MW,MZ,NA,RW,SD,SL,ST,SZ,TZ,UG,ZM,ZW),EA(AM,AZ,BY,KG,KZ,RU,TJ,TM),EP(AL,AT,BE,BG,CH,CY,CZ,DE,DK,EE,ES,FI,FR,GB,GR,HR,HU,IE,IS,IT,LT,LU,LV,MC,MK,MT,NL,NO,PL,PT,RO,RS,SE,SI,SK,SM,TR),OA(BF,BJ,CF,CG,CI,CM,GA,GN,GQ,GW,KM,ML,MR,NE,SN,TD,TG),AE,AG,AL,AM,AO,AT,AU,AZ,BA,BB,BG,BH,BN,BR,BW,BY,BZ,CA,CH,CL,CN,CO,CR,CU,CZ,DE,DJ,DK,DM,DO,DZ,EC,EE,EG,ES,FI,GB,GD,GE,GH,GM,GT,HN,HR,HU,ID,IL,IN,IR,IS,JO,JP,KE,KG,KH,KN,KP,KR,KW,KZ,LA,LC,LK,LR,LS,LU,LY,MA,MD,ME,MG,MK,MN,MW,MX,MY,MZ,NA,NG,NI,NO,NZ,OM,PA,PE,PG,PH,PL,PT,QA,RO,RS,RU,RW,SA,SC,SD,SE,SG,SK,SL,SM,ST,SV,SY,TH,TJ,TM,TN,TR,TT

(74)代理人 100070150
弁理士 伊東 忠彦
(74)代理人 100091214
弁理士 大貫 進介
(72)発明者 チャン, イーン
イギリス国 シーエフ１０ ２エイチエイチ カーディフ チャーチル ウェイ チャーチル ハウス ７ス フロア アーカートーダイクス アンド ロード エルエルピー 内
(72)発明者 クレイマー, ディーン
イギリス国 シーエフ１０ ２エイチエイチ カーディフ チャーチル ウェイ チャーチル ハウス ７ス フロア アーカートーダイクス アンド ロード エルエルピー 内