(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)公開番号
特開2023-78372
(P2023-78372A)
(43)公開日　令和5年6月6日(2023.6.6)

(51)国際特許分類　F I
G06Q 20/06 (2012.01)　G06Q 20/06

審査請求　有　請求項の数　14　ＯＬ　外国語出願　（全70頁）

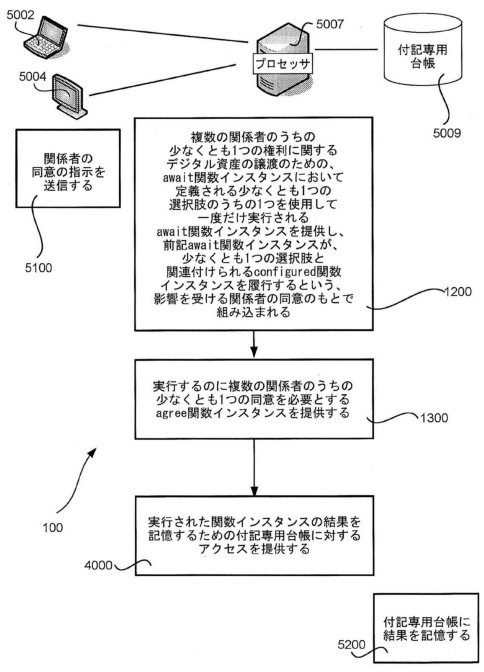
(21)出願番号　特願2023-45800(P2023-45800)
(22)出願日　令和5年3月22日(2023.3.22)
(62)分割の表示　特願2021-172368(P2021-172368)の分割
原出願日　平成29年4月28日(2017.4.28)
(31)優先権主張番号　62/329,888
(32)優先日　平成28年4月29日(2016.4.29)
(33)優先権主張国・地域又は機関　米国(US)
(31)優先権主張番号　15/181,144
(32)優先日　平成28年6月13日(2016.6.13)
(33)優先権主張国・地域又は機関　米国(US)

（特許庁注：以下のものは登録商標）
最終頁に続く

(71)出願人　521165253
デジタル・アセット・（スウィツァーランド）・ゲーエムベーハー
スイス・８０５０・チューリッヒ・トゥーアガウアーシュトラーセ・４０
(74)代理人　100108453
弁理士　村山 靖彦
(74)代理人　100110364
弁理士　実広 信哉
(74)代理人　100133400
弁理士　阿部 達彦
(72)発明者　ヴィンセント・ペイカート
アメリカ合衆国・ニューヨーク・１００１０・ニュー・ヨーク・フィフス・アヴェニュー・１６２・スイート・９０２内
最終頁に続く

(54)【発明の名称】　デジタル資産のモデリング

(57)【要約】　（修正有）
【課題】モデリングされたデジタル資産と、複数の関係者の権利に関するデジタル資産の展開とをモデリングおよび解釈するための、システムおよび方法を提供する。
【解決手段】複数の関係者の少なくとも1つの権利に関するデジタル資産譲渡のためのawait関数インスタンスにおいて定義される選択肢のうち、少なくとも1つを使用しawait関数インスタンスを一度だけ実行され、await関数インスタンスが、少なくとも1つの選択肢と関連付けられるconfigured関数インスタンスの履行が、影響を受ける関係者の同意のもとで組み込まれるステップと、複数の関係者のうちの少なくとも1つの同意のもとでagree関数インスタンスを実行するステップと、実行された結果を付記専用台帳に記憶するステップと、を含む。
【選択図】図４６



【特許請求の範囲】
【請求項１】
デジタル資産と、複数の関係者の権利に関する前記デジタル資産の展開とをモデリングするための、データ構造を操作するコンピュータで実施される方法(3900)であって、
前記複数の関係者のうちの少なくとも1つの前記権利に関する前記デジタル資産の譲渡のための、await関数(1200)インスタンスにおいて定義される少なくとも1つの選択肢のうちの1つを使用して一度だけ実行される前記await関数インスタンスを提供するステップであって、前記await関数インスタンスが、前記少なくとも1つの選択肢と関連付けられるconfigured関数インスタンスを履行するという、前記影響を受ける関係者の同意のもとで組み込まれる、ステップと、
実行するのに前記複数の関係者のうちの少なくとも1つの前記同意を必要とするagree関数(1300)インスタンスを提供するステップと、
前記実行された関数インスタンスの結果を記憶するための付記専用台帳(4000)に対するアクセスを提供するステップとを含む、方法。
【請求項２】
デジタル資産と、複数の関係者の関連する権利を含む前記デジタル資産の展開とをモデリングおよび/または記録するための、データ構造を操作するコンピュータで実施される方法であって、
前記デジタル資産の譲渡のための、await関数において定義される少なくとも1つの選択肢を含む前記await関数を決定するステップであって、前記少なくとも1つの選択肢が、関連するconfigured関数およびそれぞれの選択ステップ条件を有する、ステップと、
前記await関数インスタンスを一度だけ実行するステップであって、
前記少なくとも1つの選択肢により要求される関係者の同意の指示を受け取り、前記少なくとも1つの選択肢から有効な選択肢の選択を受け取ると、前記方法がさらに、前記少なくとも1つの選択肢の前記関連するconfigured関数インスタンスを実行するステップを含み、
前記await関数が、前記有効な選択肢の選択の前記選択ステップ条件が満たされるという決定とともに終了する、ステップと、
前記デジタル資産と関連付けられる合意記録を作成するためにagree関数を決定するステップであって、前記方法がさらに、前記合意に必要とされる関係者の同意の指示を受け取ったことに基づいて、agree関数インスタンスを実行するステップを含む、ステップと、
前記実行された関数インスタンスの結果を、記憶のために付記専用台帳に送るステップとを含む、方法。
【請求項３】
それぞれの権利が未解決である前記複数の関係者のうちの前記少なくとも1つが、その同意が必要とされる前記複数の関係者のうちの少なくとも1つと同じである、請求項1または2のいずれかに記載の方法。
【請求項４】
agree関数を無効にするためにまたは実行されないawait関数を無力化するために前記影響を受ける関係者の前記同意を必要とするdelete関数を提供するステップをさらに含み、前記付記専用台帳が、前記実行されたawait関数、agree関数、およびdelete関数の前記結果を記憶する、請求項1、2、または3のいずれか一項に記載の方法。
【請求項５】
前記デジタル資産が、現金および/または時価支払い、ファンジブル、エクイティ、債券、コモディティ、先物、権利、または商品のうちの少なくとも1つを備える、請求項1から4のいずれか一項に記載の方法。
【請求項６】
前記await関数の前記少なくとも1つの選択が、前記複数の関係者のうちの前記少なくとも1つの代表者により行われる、請求項1から5のいずれか一項に記載の方法。

【請求項７】
前記await関数の前記少なくとも1つの選択が、前記複数の関係者のうちの少なくとも2つのそれぞれの代表者により行われる、請求項1から6のいずれか一項に記載の方法。
【請求項８】
前記付記専用台帳がブロックチェーンを備える、請求項1から7のいずれか一項に記載の方法。
【請求項９】
前記付記専用台帳が、パターンマッチングに基づいてデジタル資産のステータスについてクエリされ得る、請求項1から8のいずれか一項に記載の方法。
【請求項１０】
前記付記専用台帳が、トップレベル定義に基づくクエリを使用して前記台帳の中のすべてのモデルのデジタル資産のステータスについてクエリされ得る、請求項1から8のいずれか一項に記載の方法。
【請求項１１】
アクティブモデルを非アクティブにし、今後の取引においてもはや利用可能ではないようにするための、delete関数を提供するステップをさらに含み、請求項1から10のいずれか一項に記載の方法。
【請求項１２】
モデリングされたデジタル資産および複数の関係者の権利に関する前記デジタル資産の展開を解釈する方法(100)であって、
前記複数の関係者のうちの少なくとも1つの前記権利に関する前記デジタル資産の譲渡のための、await関数(1200)インスタンスにおいて定義される少なくとも1つの選択肢のうちの1つを使用して一度だけ前記await関数インスタンスを実行するステップであって、前記await関数インスタンスが、前記少なくとも1つの選択肢と関連付けられるconfigured関数インスタンスを履行するという、前記影響を受ける関係者の同意のもとで組み込まれる、ステップと、
実行するのに前記複数の関係者のうちの少なくとも1つの前記同意を必要とするagree関数(1300)インスタンスを実行するステップと、
前記実行された関数インスタンスの結果を付記専用台帳(4000)へ記憶させるステップとを含む、方法。
【請求項１３】
モデリングされたデジタル資産と、複数の関係者の関連する権利を含む前記デジタル資産の展開とを解釈する、コンピュータで実施される方法であって、
await関数インスタンスを一度だけ実行するステップであって、
前記await関数インスタンスが、前記デジタル資産の譲渡のための、前記await関数インスタンスにおいて定義される少なくとも1つの選択肢を含み、前記少なくとも1つの選択肢が、関連するconfigured関数およびそれぞれの選択ステップ条件を有し、
前記少なくとも1つの選択肢により要求される関係者の同意の指示を受け取り、前記少なくとも1つの選択肢から有効な選択肢の選択を受け取ると、前記方法がさらに、前記少なくとも1つの選択肢の前記関連するconfigured関数インスタンスを実行するステップを含み、
前記await関数が、前記有効な選択肢の選択の前記選択ステップ条件が満たされるという決定とともに終了する、ステップと、
前記合意に必要とされる関係者の同意の指示を受け取ると、前記デジタル資産と関連付けられる合意記録を作成するためにagree関数インスタンスを実行するステップと、
付記専用台帳に記憶されるべき前記実行された関数インスタンスの結果を送るステップとを含む、方法。
【請求項１４】
それぞれの権利が未解決である前記複数の関係者のうちの前記少なくとも1つが、その同意が必要とされる前記複数の関係者のうちの少なくとも1つと同じである、請求項12

または13のいずれかに記載の方法。
【請求項１５】
agree関数を無効にするためにまたは実行されないawait関数を無力化するために前記影響を受ける関係者の前記同意を必要とするdelete関数を実行するステップと、前記実行されたawait関数、agree関数、およびdelete関数の前記結果を前記付記専用台帳に記憶するステップとをさらに含む、請求項12、13、または14のいずれか一項に記載の方法。
【請求項１６】
前記デジタル資産が、現金および/または時価支払い、ファンジブル、エクイティ、債券、コモディティ、先物、権利、または商品のうちの少なくとも1つを備える、請求項12から15のいずれか一項に記載の方法。
【請求項１７】
前記await関数の前記少なくとも1つの選択が、前記複数の関係者のうちの前記少なくとも1つの代表者により行われる、請求項12から16のいずれか一項に記載の方法。
【請求項１８】
前記await関数の前記少なくとも1つの選択が、前記複数の関係者のうちの少なくとも2つのそれぞれの代表者により行われる、請求項12から17のいずれか一項に記載の方法。
【請求項１９】
前記付記専用台帳がブロックチェーンを備える、請求項12から18のいずれか一項に記載の方法。
【請求項２０】
前記付記専用台帳が、パターンマッチングに基づいてデジタル資産のステータスについてクエリされ得る、請求項12から19のいずれか一項に記載の方法。
【請求項２１】
前記付記専用台帳が、トップレベル定義に基づくクエリを使用して前記台帳の中のすべてのモデルのデジタル資産のステータスについてクエリされ得る、請求項12から19のいずれか一項に記載の方法。
【請求項２２】
アクティブモデルを非アクティブにし、今後の取引においてもはや利用可能ではないようにするための、delete関数を実行するステップをさらに含む、請求項12から21のいずれか一項に記載の方法。
【請求項２３】
モデリングされたデジタル資産と、複数の関係者の権利に関する前記デジタル資産の展開とを解釈するように構成されるデジタルシステム(2000～3300、4200～4300)であって、
少なくとも1つのプロセッサであって、前記複数の関係者のうちの少なくとも1つの前記権利に関する前記デジタル資産の譲渡のための、await関数(1200)インスタンスにおいて定義される少なくとも1つの選択肢のうちの1つを使用して前記await関数インスタンスを一度だけ実行するように構成され、前記await関数インスタンスが、前記少なくとも1つの選択肢と関連付けられるconfigured関数インスタンスを履行するという、前記影響を受ける関係者の同意のもとで組み込まれ、前記複数の関係者のうちの少なくとも1つの前記同意を必要とする前記少なくとも1つの選択肢内のagree関数(1300)インスタンスを実行するように構成される、少なくとも1つのプロセッサと、
前記実行される関数インスタンスの解釈された結果を付記専用台帳へ記憶させるように構成される少なくとも1つのストレージデバイス(4000)とを備える、システム。
【請求項２４】
モデリングされたデジタル資産と、複数の関係者の関連する権利を含む前記デジタル資産の展開とを解釈するように構成されるデジタルシステムであって、
少なくとも1つのプロセッサであって、

await関数インスタンスを一度だけ実行するように構成され、
前記await関数インスタンスが、前記デジタル資産の譲渡のための、前記await関数インスタンスにおいて定義される少なくとも1つの選択肢を含み、前記少なくとも1つの選択肢が、関連するconfigured関数およびそれぞれの選択ステップ条件を有し、
前記少なくとも1つの選択肢により要求される関係者の同意の指示を受け取り、前記少なくとも1つの選択肢から有効な選択肢の選択を受け取ると、前記少なくとも1つのプロセッサがさらに、前記少なくとも1つの選択肢の前記関連するconfigured関数インスタンスを実行するように構成され、
await関数が、前記有効な選択肢の選択の前記選択ステップ条件が満たされるという決定とともに終了し、
前記合意に必要とされる関係者の同意の指示を受け取ると、前記デジタル資産と関連付けられる合意記録を作成するためにagree関数インスタンスを実行するように構成される、少なくとも1つのプロセッサと、
前記実行される関数インスタンスの解釈された結果を付記専用台帳へ記憶させるように構成される少なくとも1つのストレージデバイスとを備える、システム。

【請求項２５】
それぞれの権利が未解決である前記複数の関係者のうちの前記少なくとも1つが、その同意が必要とされる前記複数の関係者のうちの少なくとも1つと同じである、請求項23または24のいずれかに記載のシステム。

【請求項２６】
前記プロセッサがさらに、agree関数を無効にするためにまたは実行されないawait関数を無力化するために前記影響を受ける関係者の前記同意を必要とするdelete関数を実行し、前記await関数、前記agree関数、および前記delete関数の前記実行結果を記憶し、前記付記専用台帳の中の関数を削除するように構成される、請求項23、24、または25のいずれか一項に記載のシステム。

【請求項２７】
前記デジタル資産が、現金および/または時価支払い、ファンジブル、エクイティ、債券、コモディティ、先物、権利、または商品のうちの少なくとも1つを備える、請求項23から26のいずれか一項に記載のシステム。

【請求項２８】
前記await関数の前記少なくとも1つの選択が、前記複数の関係者のうちの前記少なくとも1つの代表者により行われる、請求項23から27のいずれか一項に記載のシステム。

【請求項２９】
前記await関数の前記少なくとも1つの選択が、前記複数の関係者のうちの少なくとも2つのそれぞれの代表者により行われる、請求項23から28のいずれか一項に記載のシステム。

【請求項３０】
前記付記専用台帳がブロックチェーンを備える、請求項23から29のいずれか一項に記載のシステム。

【請求項３１】
前記付記専用台帳が、パターンマッチングに基づいてデジタル資産のステータスについてクエリされ得る、請求項23から30のいずれか一項に記載のシステム。

【請求項３２】
前記付記専用台帳が、トップレベル定義に基づくクエリを使用して前記台帳の中のすべてのモデルのデジタル資産のステータスについてクエリされ得る、請求項23から30のいずれか一項に記載のシステム。

【請求項３３】
前記プロセッサがさらに、アクティブモデルを非アクティブにし、今後の取引においてもはや利用可能ではないようにするために、delete関数を実行するように構成される、請求項23から32のいずれか一項に記載のシステム。

【発明の詳細な説明】
【技術分野】
【０００１】
相互参照
本出願は、２０１６年４月２９日に米国特許商標庁(USPTO)に出願された、Digital Asset Modelling Language Simulatorという表題の米国仮特許出願第６２/３２９,８８８号(代理人整理番号16-30013-US-PV、顧客整理番号DAH-2016011P)への米国特許法第１１９条による優先権を主張し、この仮出願の内容の全体が参照により本明細書に組み込まれる。
【０００２】
本開示は、デジタル資産モデリングシステムと、デジタル資産、債務、および取引をモデリングし、追跡し、清算するための方法とに関する。
【背景技術】
【０００３】
資産、債務、および取引を清算するために利用される既存の非公開の集中的に管理される台帳は、不明瞭であり誤りが生じやすいと考えられる。このことは監査を煩雑にし、多数の重複する処理および台帳を必要とし、不正の可能性を許す。既存の台帳の構造を代替する最初のかつ現在最大のものは、ブロックチェーンデータ構造を使用する、ビットコインと呼ばれる分散型のデジタル台帳により代表される。ビットコインの運営の基本的な原理は、システムが、公開-秘密鍵暗号を利用するピアツーピア取引機構としてセットアップされ、中間のまたは中央のリポジトリがなく、ネットワークの中のすべての参加者が台帳の完全なコピーの整合性をリアルタイムで保ち検証することを可能にする、というものである。ビットコインブロックチェーンは、世界中の変名の関係者と交換することができるトラストレスなネイティブ資産であるビットコインを作成するために設計された。
【０００４】
ビットコイン様の、またはブロックチェーン様のシステム上でデジタル資産をサポートするように構築された現在のプラットフォームは一般に、既存の取引業の多くに対して法律により求められ得るような、金融機関への包括的な保護を提供するような構造ではない。これらのプラットフォームは、一般に金融機関および金融取引に対する規制制度を考慮していない。結果として、機関投資家は、デジタル資産市場に参入することをためらい、既存の商取引のために分散型台帳を使用することを避けてきた。
【発明の概要】
【課題を解決するための手段】
【０００５】
本明細書全体で、「備える(comprise)」という語、または「備える(comprises)」もしくは「備えている(comprising)」などの変形は、述べられた要素、整数もしくはステップ、または要素、整数、もしくはステップのグループの包含を示唆するが、あらゆる他の要素、整数、もしくはステップ、または要素、整数、もしくはステップのグループの除外を示唆しないものとして、理解されるであろう。
【０００６】
本明細書全体で、「備える(comprise)」という語、または「備える(comprises)」もしくは「備えている(comprising)」などの変形は、述べられた要素、整数もしくはステップ、または要素、整数、もしくはステップのグループの包含を示唆するが、あらゆる他の要素、整数、もしくはステップ、または要素、整数、もしくはステップのグループの除外を示唆しないものとして、理解されるであろう。
【０００７】
本明細書で開示される実施形態は、デジタル資産のコンピュータにより実行される取引に柔軟性を加えるための機構を提供する。実施形態は、取引を実行するコンピュータシステムが、新しい有利な方式で動作することを可能にするための、新しいデータモデルおよび機能を提供する。デジタル資産と、複数の関係者の権利に関するデジタル資産の展開と

をモデリングする、例示的な実施形態の方法が提供され、この方法は、複数の関係者のうちの少なくとも1つの権利に関するデジタル資産の譲渡のためのawait関数インスタンスにおいて定義される少なくとも1つの選択肢のうちの1つを使用して一度だけ実行されるawait関数インスタンスを提供するステップであって、前記await関数インスタンスが、少なくとも1つの選択肢と関連付けられるconfigured関数インスタンスを履行するという、影響を受ける関係者の同意のもとで組み込まれる、ステップと、実行するのに複数の関係者のうちの少なくとも1つの同意を必要とするagree関数インスタンスを提供するステップと、実行された関数インスタンスの結果を記憶するための付記専用台帳を提供するステップとを含む。

【０００８】

デジタル資産とデジタル資産の展開とを、複数の関係者の関連する権利を含めてモデリングおよび/または記録する、例示的な実施形態の方法が提供され、この方法は、デジタル資産の譲渡のためのawait関数において定義される少なくとも1つの選択肢を含むawait関数を決定するステップを含み、少なくとも1つの選択肢は、関連するconfigured関数およびそれぞれの選択ステップ条件を有する。方法はさらに、await関数インスタンスを一度だけ実行するステップであって、少なくとも1つの選択肢により要求される関係者の同意の指示を受け取り、少なくとも1つの選択肢から有効な選択肢の選択を受け取ると、方法がさらに、少なくとも1つの選択肢の関連するconfigured関数インスタンスを実行するステップであって、await関数が、有効な選択肢の選択の選択ステップ条件が満たされるという決定とともに終了する、ステップと、デジタル資産と関連付けられる合意記録を作成するためにagree関数を決定するステップとを含み、方法はさらに、合意に必要とされる関係者の同意の指示を受け取ったことに基づいて、agree関数インスタンスを実行するステップを含む。方法はさらに、実行された関数インスタンスの結果を、記憶のために付記専用台帳に送るステップを含む。

【０００９】

上で説明されたデジタル資産とデジタル資産の展開とをモデリングおよび/または記録する方法を提供することができ、ここで、それぞれの権利が未解決である複数の関係者のうちの少なくとも1つは、その同意が必要とされる複数の関係者のうちの少なくとも1つと同じである。agree関数を無効にするためにまたは実行されないawait関数を無力化するために影響を受ける関係者の同意を必要とするdelete関数を提供するステップをさらに含む方法を提供することができ、ここで、付記専用台帳が、実行されたawait関数、agree関数、およびdelete関数の結果を記憶する。デジタル資産が、現金および/または時価支払い、ファンジブル、エクイティ、債券、コモディティ、先物、権利、または商品のうちの少なくとも1つを備える、方法を提供することができる。await関数の少なくとも1つの選択が複数の関係者のうちの少なくとも1つの代表者により行われる、方法を提供することができる。await関数の少なくとも1つの選択が複数の関係者のうちの少なくとも2つのそれぞれの代表者により行われる、方法を提供することができる。付記専用台帳がブロックチェーンを備える、方法を提供することができる。付記専用台帳がパターンマッチングに基づいてデジタル資産のステータスについてクエリされ得る、方法を提供することができる。付記専用台帳がトップレベル定義に基づくクエリを使用して台帳の中のすべてのモデルのデジタル資産のステータスについてクエリされ得る、方法を提供することができる。アクティブモデルを非アクティブにし、今後の取引においてもはや利用可能ではないようにするための、delete関数を提供するステップをさらに含む、方法を提供することができる。

【００１０】

モデリングされたデジタル資産および複数の関係者の権利に関するデジタル資産の展開を解釈する例示的な実施形態の方法が提供され、この方法は、複数の関係者のうちの少なくとも1つの権利に関するデジタル資産の譲渡のためのawait関数インスタンスにおいて定義される少なくとも1つの選択肢のうちの1つを使用して一度だけawait関数インスタンスを実行するステップであって、前記await関数インスタンスが、少なくとも1つの選

択肢と関連付けられるconfigured関数インスタンスを履行するという、影響を受ける関係者の同意のもとで組み込まれる、ステップと、実行するのに複数の関係者のうちの少なくとも1つの同意を必要とするagree関数インスタンスを実行するステップと、実行された関数インスタンスの結果を付記専用台帳に記憶するステップとを含む。

【００１１】

モデリングされたデジタル資産と、複数の関係者の関連する権利を含むデジタル資産の展開とを解釈する例示的な実施形態の方法が提供され、この方法は、await関数インスタンスを一度だけ実行するステップを含み、await関数インスタンスが、デジタル資産の譲渡のためのawait関数インスタンスにおいて定義される少なくとも1つの選択肢を含み、少なくとも1つの選択肢が、関連するconfigured関数およびそれぞれの選択ステップ条件を有し、少なくとも1つの選択肢により要求される関係者の同意の指示を受け取り、少なくとも1つの選択肢から有効な選択肢の選択を受け取ると、方法はさらに、少なくとも1つの選択肢の関連するconfigured関数インスタンスを実行するステップを含み、await関数が、有効な選択肢の選択の選択ステップ条件が満たされるという決定とともに終了する。方法はさらに、合意に必要とされる関係者の同意の指示を受け取ると、デジタル資産と関連付けられる合意記録を作成するためにagree関数インスタンスを実行するステップを含む。方法はさらに、実行された関数インスタンスの結果を付記専用台帳に送るステップを含む。

【００１２】

それぞれの権利が未解決である複数の関係者のうちの少なくとも1つが、その同意が必要とされる複数の関係者のうちの少なくとも1つと同じである、モデリングされたデジタル資産およびデジタル資産の展開を解釈する上で説明された方法を提供することができる。agree関数を無効にするためにまたは実行されないawait関数を無力化するために影響を受ける関係者の同意を必要とするdelete関数を実行するステップと、実行されたawait関数、agree関数、およびdelete関数の結果を付記専用台帳に記憶するステップとをさらに含む、方法を提供することができる。デジタル資産が、現金および/または時価支払い、ファンジブル、エクイティ、債券、コモディティ、先物、権利、または商品のうちの少なくとも1つを備える、方法を提供することができる。await関数の少なくとも1つの選択が複数の関係者のうちの少なくとも1つの代表者により行われる、方法を提供することができる。await関数の少なくとも1つの選択が複数の関係者のうちの少なくとも2つのそれぞれの代表者により行われる、方法を提供することができる。付記専用台帳がブロックチェーンを備える、方法を提供することができる。付記専用台帳がパターンマッチングに基づいてデジタル資産のステータスについてクエリされ得る、方法を提供することができる。付記専用台帳がトップレベル定義に基づくクエリを使用して台帳の中のすべてのモデルのデジタル資産のステータスについてクエリされ得る、方法を提供することができる。アクティブモデルを非アクティブにし、今後の取引においてもはや利用可能ではないようにするための、delete関数を実行するステップをさらに含む、方法を提供することができる。

【００１３】

モデリングされたデジタル資産と、複数の関係者の関連する権利に関するデジタル資産の展開とを解釈するように構成される例示的な実施形態のデジタルシステムが提供され、このシステムは、少なくとも1つのプロセッサであって、複数の関係者のうちの少なくとも1つの権利に関するデジタル資産の譲渡のためのawait関数インスタンスにおいて定義される少なくとも1つの選択肢のうちの1つを使用してawait関数インスタンスを一度だけ実行するように構成され、前記await関数インスタンスが、少なくとも1つの選択肢と関連付けられるconfigured関数インスタンスを履行するという、影響を受ける関係者の同意のもとで組み込まれ、複数の関係者のうちの少なくとも1つの同意を必要とする少なくとも1つの選択肢内のagree関数インスタンスを実行するように構成される、少なくとも1つのプロセッサと、実行される関数インスタンスの解釈された結果を付記専用台帳に記憶するように構成される少なくとも1つのストレージデバイスとを備える。

【００１４】
モデリングされたデジタル資産と、複数の関係者の関連する権利を含むデジタル資産の展開とを解釈するように構成される、ある例示的な実施形態のデジタルシステムが提供され、このシステムは、await関数インスタンスを一度だけ実行するように構成される少なくとも1つのプロセッサを備え、await関数インスタンスが、デジタル資産の譲渡のためのawait関数インスタンスにおいて定義される少なくとも1つの選択肢を含み、少なくとも1つの選択肢が、関連するconfigured関数およびそれぞれの選択ステップ条件を有し、少なくとも1つの選択肢により要求される関係者の同意の指示を受け取り、少なくとも1つの選択肢から有効な選択肢の選択を受け取ると、少なくとも1つのプロセッサがさらに、少なくとも1つの選択肢の関連するconfigured関数インスタンスを実行するように構成され、await関数が、有効な選択肢の選択の選択ステップ条件が満たされるという決定とともに終了する。少なくとも1つのプロセッサはさらに、合意に必要とされる関係者の同意の指示を受け取ると、デジタル資産と関連付けられる合意記録を作成するためにagree関数インスタンスを実行するように構成される。システムはさらに、実行される関数インスタンスの解釈された結果を付記専用台帳に記憶するように構成される少なくとも1つのストレージデバイスを備える。

【００１５】
それぞれの権利が未解決である複数の関係者のうちの少なくとも1つが、その同意が必要とされる複数の関係者のうちの少なくとも1つと同じである、上で説明されたシステムを提供することができる。agree関数を無効にするためにまたは実行されないawait関数を無力化するために影響を受ける関係者の同意を必要とするdelete関数を実行し、await関数、agree関数、およびdelete関数の実行結果を付記専用台帳に記憶するようにさらに構成されるプロセッサを伴う、システムを提供することができる。デジタル資産が、現金および/または時価支払い、ファンジブル、エクイティ、債券、コモディティ、先物、権利、または商品のうちの少なくとも1つを備える、システムを提供することができる。await関数の少なくとも1つの選択が複数の関係者のうちの少なくとも1つの代表者により行われる、システムを提供することができる。await関数の少なくとも1つの選択が複数の関係者のうちの少なくとも2つのそれぞれの代表者により行われる、システムを提供することができる。付記専用台帳がブロックチェーンを備える、システムを提供することができる。付記専用台帳がパターンマッチングに基づいてデジタル資産のステータスについてクエリされ得る、システムを提供することができる。付記専用台帳がトップレベル定義に基づくクエリを使用して台帳の中のすべてのモデルのデジタル資産のステータスについてクエリされ得る、システムを提供することができる。アクティブモデルを非アクティブにし、今後の取引においてもはや利用可能ではないようにするための、delete関数を実行するようにさらに構成されるプロセッサを伴う、システムを提供することができる。

【００１６】
本明細書で開示される実施形態は、デジタル資産のコンピュータにより実行される取引に柔軟性を加えるための機構を提供することができる。実施形態は、取引を実行するコンピュータシステムが新しく有利な方式で動作することを可能にする、新しいデータモデルおよび関数を提供することができる。

【００１７】
説明のための限定しない例示的な実施形態は、特に添付の図面とともに参照されると、以下の発明を実施するための形態からより明確に理解され得る。

【図面の簡単な説明】

【００１８】
【図１】本開示のある例示的な実施形態による、Digital Asset Modeling Language(DAML(商標))のための言語認識器を示すフローチャートである。
【図２】本開示のある例示的な実施形態による、アクションに対する状態変化を示すハイブリッド類比図(hybrid analogical diagram)である。
【図３】本開示のある例示的な実施形態による、図2の状態変化に対するデジタル化され

た状態を示すハイブリッド類比図である。
【図４】本開示のある例示的な実施形態による、台帳への図3の状態の記録を示すハイブリッド類比図である。
【図５】本発明のある例示的な実施形態による、図4の台帳の状態の変化、安全確保、および認可を示すハイブリッド類比図である。
【図６】本開示のある例示的な実施形態による、図5の変更可能な、安全な、かつ認可された台帳の複製を示すハイブリッド類比図である。
【図７】本開示のある例示的な実施形態による、図6の複製された台帳に対する分散型のコンセンサスを示すハイブリッド類比図である。
【図８】本開示のある例示的な実施形態と対比した、図7の分散型台帳内での無効な状態または矛盾した状態の気付かれない広がりを示すハイブリッド類比図である。
【図９】本開示のある例示的な実施形態と対比した、図8の分散型台帳内での無効な状態または矛盾した状態の気付かれる広がりを示すハイブリッド類比図である。
【図１０】本発明のある例示的な実施形態による、エントリの検証と、図8および図9の台帳内での無効な状態または矛盾した状態の防止とを示す、ハイブリッド類比図である。
【図１１】本発明のある例示的な実施形態による、図10の台帳に対する検証プロセスの動的な強化および拡張を示すハイブリッド類比図である。
【図１２】本発明のある例示的な実施形態による、単一の関係者に対する定義されたアクションを伴うAwait関数を使用したDAML(商標)コードを示すハイブリッド図である。
【図１３】本開示のある例示的な実施形態による、条件文内の複数の関係者間での現実世界の合意に対する定義されたアクションを使用したDAML(商標)コードを示すハイブリッド図である。
【図１４】本開示のある例示的な実施形態による、ネイティブのファンジブルデジタル資産に対して動作するDAML(商標)コードを示すハイブリッド図である。
【図１５】本開示のある例示的な実施形態による、図12の2つの関係者の論理および単一の関係者の論理による交互のアクションを伴うDAML(商標)コードを示すハイブリッド図である。
【図１６】本開示のある例示的な実施形態による、複数のステップを組み合わせるDAML(商標)コードを示すハイブリッド図である。
【図１７】本開示のある例示的な実施形態による、2つの関係者間での資産の交換のためのDAML(商標)コードを示すハイブリッド図である。
【図１８】本開示のある例示的な実施形態による、DAML(商標)コードの選択肢を示すハイブリッド図である。
【図１９】本開示のある例示的な実施形態による、DAML(商標)コードの合成を示すハイブリッド図である。
【図２０】本発明のある例示的な実施形態による、DAML(商標)の順序付けられた台帳エントリを示すハイブリッド図である。
【図２１】本開示のある例示的な実施形態による、順序付けられた台帳エントリを伴うDAMLベースの台帳を示すハイブリッド図である。
【図２２】本開示のある例示的な実施形態による、順序付けられタイムスタンプが付けられた台帳エントリを伴うDAMLベースの台帳を示すハイブリッド図である。
【図２３】本開示のある例示的な実施形態による、2つの例示的なDAML(商標)ストレージと、複数の関係者にわたる台帳論理の展開とを示すハイブリッド図である。
【図２４】本発明のある例示的な実施形態による、台帳エントリを伴うDAML(商標)の取引を示すハイブリッド図である。
【図２５】本発明のある例示的な実施形態による、台帳エントリおよび機密性を伴うDAML(商標)の取引を示すハイブリッド図である。
【図２６】本発明のある例示的な実施形態による、新しい台帳エントリの取引のコミットメントの複数関係者による認可を含む、DAMLベースの台帳を示すハイブリッド図である。

【図２７】本発明のある例示的な実施形態による、ハッシュ対エントリおよび詳細を含む、ＤＡＭＬベースの2層の台帳を示すハイブリッド図である。
【図２８】本発明のある例示的な実施形態による、ＤＡＭＬベースのハッシュ中心公開台帳層または公開ログを示すハイブリッド図である。
【図２９】本発明のある例示的な実施形態による、ＤＡＭＬベースの非公開の、かつ共有可能な非公開の台帳を示すハイブリッド図である。
【図３０】本開示のある例示的な実施形態による、関連する台帳エントリの特定および台帳エントリの活発さの追跡を伴う、2層のＤＡＭＬ(商標)台帳を示すハイブリッド図である。
【図３１】本開示のある例示的な実施形態による、ブロック指向論理を伴い、関連する台帳エントリの特定およびブロック中心の活発さの追跡を伴う、2層のＤＡＭＬ(商標)台帳を示すハイブリッド図である。
【図３２】本開示のある例示的な実施形態による、非公開台帳エントリのプロパティの証明書または証明可能なプロパティの関係者間での共有を示すハイブリッド図である。
【図３３】本開示のある例示的な実施形態による、分散型台帳のホストされたコピーおよびホストされないコピーを示すハイブリッド図である。
【図３４】本開示のある例示的な実施形態による、Ａｇｒｅｅ、Ａｗａｉｔ、およびＤｅｌｅｔｅコマンドのためのＤＡＭＬ(商標)台帳エントリ型を示すハイブリッド図である。
【図３５】本開示のある例示的な実施形態による、ＤＡＭＬ(商標)Ａｇｒｅｅコマンドの使用を示すハイブリッド図である。
【図３６】本開示のある例示的な実施形態による、インスタンス化される台帳エントリを示すハイブリッド図である。
【図３７】本開示のある例示的な実施形態による、エクイティモデルにおける外部への通知のための合意のＤＡＭＬ(商標)コードを示すハイブリッド図である。
【図３８】本開示のある例示的な実施形態による、キューブモデルにおけるＡｗａｉｔコマンドを使用したＤＡＭＬ(商標)コードを示すハイブリッド図である。
【図３９】本発明のある例示的な実施形態による、ｄｅｌｅｔｅコマンドを伴うＤＡＭＬ(商標)台帳を示すハイブリッド図である。
【図４０】本開示のある例示的な実施形態による、役割中心のフローを伴う台帳アルゴリズムを示すハイブリッド図である。
【図４１】本発明のある例示的な実施形態による、役割中心のフローおよびコメントを伴う台帳アルゴリズムを示すハイブリッド図である。
【図４２】本発明のある例示的な実施形態による、関係者中心の台帳アルゴリズムフローを示すハイブリッド図である。
【図４３】本開示のある例示的な実施形態による、台帳中心の台帳アルゴリズムを示すハイブリッド図である。
【図４４】本開示のある例示的な実施形態による、台帳取引を開始するための関数を示すハイブリッド図である。
【図４５】一例による、デジタル資産をモデリングおよび/または記録するためのデジタルシステムの概略図である。
【図４６】デジタル資産およびその展開をモデリングおよび/または記録するためにデータ構造を操作するコンピュータで実施される方法のフローチャートである。
【図４７】デジタル資産およびその展開をモデリングおよび/または記録するためにデータ構造を操作する別のコンピュータで実施される方法のフローチャートである。
【図４８】図46および図47において説明される方法への追加のステップのフローチャートである。
【図４９】契約テンプレートを表す概略図である。
【図５０】アクティブな契約に関する対話の概略図である。
【図５１】ａｗａｉｔ関数インスタンスを表す図である。
【図５２】非アクティブな契約および新しい契約の概略図である。

【図５３】処理デバイスの例の図である。
【発明を実施するための形態】
【００１９】
本発明の概念は、例示的な実施形態が示されている添付の図面を参照してより完全に説明される。しかしながら、本発明の概念は、多くの異なる形態で具現化されてよく、本明細書に記載される実施形態に限定されるものと見なされるべきではない。本明細書全体で、同様の参照番号は同様の要素を指し得る。本明細書では、「モデル」という語は、合意または潜在的な取引の少なくとも1つの束として定義され、このモデルは、たとえば親契約(Master Contract)によって与えられ得るものなどの、ある適用規則のもとで、デジタル的に表現される合意または法的拘束力のある契約を表現する可能性を有することも有しないこともある。
【００２０】
図1に示されるように、Digital Asset Modeling Language(商標)(DAML(商標))のための言語認識器が、参照番号100によって全体的に示されている。言語認識器は、コンピュータプロセッサの制御下で実行されるルーチンである。DAML(商標)は、コンピュータ可読データ構造として記憶され得るあらかじめ定義されたステートメントを含む。言語認識器は開始ブロック110を含み、開始ブロック110は制御権を入力ブロック112に渡す。入力ブロック112は、メモリまたは通信チャネルなどから、コンピュータ実行可能ソースコードを受け取り、制御権を機能ブロック116に渡す。ブロック116は、DAML(商標)言語のために定義される単項パーサコンビネータブロックの助けにより、受け取られたソースコードに対して字句分析と構文分析の両方を実行し、構文解析されたDAML(商標)言語の抽象構文木118の産生に成功すると、制御権を終了ブロック120に渡す。抽象構文木118はデータ構造として記憶される。DAML(商標)処理論理の現在の例示的な実施形態は、DAML(商標)抽象構文木構造を受け入れ、それを新しいDAML(商標)抽象構文木に変換し、シリアライズされたDAML(商標)抽象構文木構造をデータ構造として記憶する各DAML(商標)台帳エントリのデータ構造に基づく。DAML(商標)処理論理の代替的な実施形態は、たとえば再帰下降などの、DAML(商標)ソースコードをDAML(商標)抽象構文木に変換するための代替的なパーサ技術を使用することができる。DAML(商標)台帳エントリの代替的な実施形態は、たとえば、DAML(商標)言語のより高次の関数型構造を1次の構造で置き換えるために、型推論ステップおよび脱関数化ステップの助けをさらに借りてDAML(商標)抽象構文木をコンパイルすることなどによって、DAML(商標)言語表現のデータ中心の表現を記憶することができる。
【００２１】
動作中、DAML(商標)認識器は、コンピュータプロセッサの制御下で、デジタル資産をモデリングするのに適した関数および構文を解釈する。例示的な実施形態のDAML(商標)認識器は、すべてのDAML(商標)の中核の言語特徴を実装する。
DAML(商標)のコメントは次のようにHaskellスタイルを使用してサポートされる。

```
--これはコメント行です
{-
これは複数行のコメントです。コメントは{-ネストする--ことができます-}
-}
```

【００２２】
プリミティブ型は次のものを含む。Boolは2つのブール値TrueおよびFalseである。Textはユニコード文字の列である。Integerは任意のサイズの符号付き整数である。Decimalは10進の浮動小数点数である。Partyは法的主体の固有の識別子である。Timeは実装形態に依存する精度を持つ絶対的な時間の点である。RelTimeは点と点の間の時間的な差分(同じ精度を受け継ぐ)である。ContractIdは有形な契約のインスタンスの識別子(有形な契約が印刷される書類という物理的な証明書に対するデジタル的な等価物)である。

【００２３】
関数型に対して合成される型があり、これらは、Haskellにおけるように演算子-〉を使用して構築される。たとえば、「Integer-〉Text」は、Integer型の1つの引数を受け入れてText値を返す関数のためのものである。
【００２４】
DAML(商標)は加えて、次のような特別な型をサポートする。Recordは、実行時にそのアクセスが確認されるラベリングされた値の記録である。Choiceは、関係者に与えられる選択肢である。ある例示的な実施形態では、各選択は関係者により行われる。ある代替的な実施形態では、モデル契約は、関係者の代表者がそのような選択を行えるように、関係者の代表者の意見および/または公開鍵基盤(PKI)などの認可スキームを組み込むことができる。Agreementは、発生しなければならない台帳外のイベントについての合意である。Contractは、モデルまたは潜在的な契約の条件をカバーする。Updateは、台帳更新の記述であり、これは台帳においてアクティブモデルを作成して非アクティブ化する。Scenarioは、複数の関係者の契約に関する対話の記述であり、DAML(商標)の実施形態はこれを単体テストのために使用し得る。Assertionは、あるブール条件がTrueであると評価されることのアサーションである。PureValueは、ledger-updateまたはscenarioの両方において副作用のないステップとして使用され得るようにラップされた値である。
【００２５】
以下の語は、この言語のキーワードである:await、at、named、if、chooses、then、else、such、that、exercises、agree、agrees、with、on、let、in、create、commit、commits、scenario、update。
【００２６】
プリミティブ型に関する表現は、リテラル、組み込み関数、関数適用、およびラムダ抽象から構築される。2つのBoolリテラルはTrueおよびFalseである。Textリテラルは、ダブルクオートを使用して書かれ、Haskell Stringリテラル(Haskell Report 2010のセクション2.6参照)と同じエスケープ規則を使用する。例示的なtextリテラルは、文字列「Hello world」を表記する「Hello world」である。Integerリテラルは10進法として書かれ、負の数に対して-が前に置かれる可能性がある。例は1024および-1である。Decimalリテラルは10進の浮動小数点を表す。これは、小数点を有することでintegerリテラルとは区別される。たとえば、1234.56789、0.5、または1.0である。decimalリテラルの一般的な形は、正規表現[0-9]+＼.[0-9]+により与えられる。DAML(商標)はDecimalリテラルにおいて末尾の0を無視することに留意されたい。たとえば、1.5==1.50==1.5000である。Partyリテラルは、シングルクオートとシングルクオートの間の英数字の列として書かれる。例は「CITI GROUP」および「DA」である。ある例示的な実施形態では、法的主体の固有の識別子を形成するのに英数字が十分であると仮定される。ある代替的な実施形態では、法的主体を参照するために具体的な規格(たとえば、LEI)が使用され得る。
【００２７】
Timeリテラルは、常に協定世界時として解釈され、ISO-8061に従ってリテラルとして書かれ、次の正規表現により受け入れられる部分集合に制約される:[0-9]{4}-[0-9]{2}-[0-9]{2}T[0-9]{2}:[0-9]{2}(:[0-9]{2}(＼.[0-9]+)?)?Z。たとえば、2007-04-05T14:30Z、2007-04-05T14:30:00Z、および2007-04-05T14:30:00.0Zはすべて、2007年4月5日の協定世界時14:30を表記する。Timeリテラルは、任意の1秒未満の精度を有し得る。しかしながら、その精度のうちのどれだけが実際に表現できるかは、実装形態により決まる。RelTimeリテラルは、2つの絶対的な時点と時点の間の名目上の差分を表す。2つの時点の間の名目上の差分は、閏秒があったかどうか、または挿入されたかどうかとは無関係に、常に同じである。相対的な時間は秒の端数として表され、たとえば、toRelTime(24.0*60.0*60.0)は一日を構成する秒の量を表記する(toRelTimeの説明については以下を参照されたい)。RelTimeリテラルは任意の秒の端数を指定

できるが、その精度のうちのどれだけが実際に表現できるかは実装形態により決まる。しかしながら、すべての実装形態が、RelTimeおよびTimeが同じ精度を有することを保証する。代替的な実施形態では、「1d」などの代替的なリテラルが、実装形態の最大限の精度において単一の日の始まりと終わりの間の相対的な時間を表記するために使用され得る。

【００２８】

型ContractIdのリテラルを書くための固定された手段はなく、それは、ContractIdの形が実行モデルに依存するからである。RecordはJavaScriptのように波括弧を使用して書かれる。たとえば、{"x":1, "y":2}は、その値が1::Integerおよび2::Integerである、xおよびyとラベリングされた2つのフィールドを伴うRecordである。DAML(商標)実施形態は現在、どのフィールドが型の中に存在するかを追跡しない。その上、ラベルにはリテラル文字列しか使用することができない。DAML(商標)の実施形態は、空の記録に対して{}を使用する。

【００２９】

DAML(商標)の実施形態は、ブーリアンの論理和、論理積、および否定に対する標準的なHaskell演算子||、&&、notをサポートする(Haskell Report 2010の9章を参照)。DAML(商標)の実施形態は、通常の優先順位で、加算、減算、乗算、整数の除算、および整数のべき乗に対する、標準的な算術演算子+、-、*、/、^をサポートする。Haskellにおけるように、それらを接頭の表記で使用するには、それらを丸括弧の中に書くことができる。たとえば、(+)1 2は1 + 2を書く別の方法である。Haskellとは対照的に、DAML(商標)の実施形態は、中置演算子がスペースにより囲まれることを要求する。これにより、2 - -2の曖昧さがなくなり、2 - (-2)を意味するようになる。絶対的な時間および相対的な時間とともに使われるときの、-および+の多重定義されたバージョンについては、以下を参照されたい。

【００３０】

DAML(商標)の実施形態は、Decimalのセットを、d == n / 10 ^ kであるような整数nおよびkが存在するすべての有理数dとなるように定義する。すなわち、Decimalは、小数の形で書かれるとき、有限小数、無限小数、循環小数、非循環小数の有理数および無理数を含む小数のうちの、小さな部分集合である。Decimal(大文字「D」を伴う)は、小数のうちの有限小数の部分集合だけを含む。以下では、DAML(商標)がDecimalとともに機能することをどのようにサポートするかを説明する。DAML(商標)の実施形態は、Decimalの加算、減算、および乗算を実行するために、算術演算子+、-、*を多重定義する。これらの演算子のいずれも丸めを実行せず、それは、Decimalの集合が、加法、減法、および乗法について閉性を持っているからである。

【００３１】

DAML(商標)の実施形態は、関数round :: Integer -〉 Decimal -〉 Decimalを用いて所与の精度へのDecimalの丸めをサポートし、ここで、round prec dはDecimal dをn / 10 ^ precの形の最も近いDecimalに丸め、端数0.5(tie)はDecimalを偶数nに丸めることによって解決され、これは銀行家の丸めモードとしても知られている。たとえば、

```
round 0 2.5 == 2.0     round 0 (-2.5) == -2.0
round 0 3.5 == 4.0     round 0 (-3.5) == -4.0
round 0 3.2 == 3.0     round 0 (-3.2) == -3.0
round 0 3.8 == 4.0     round 0 (-3.8) == -4.0
```

である。

【００３２】

「正しい」丸めモードはない。DAML(商標)の実施形態は、必要に応じてさらなる丸めモードを追加し得る。

【００３３】

Decimalの集合が除法について閉性を持たないこと、たとえば、2つのDecimalを割

ることが必ずしもDecimalを生まないことに留意されたい。たとえば、1.0 / 3.0 == 0.3333...は有理数であるが、(有限小数の)Decimalではない。したがって、DAML(商標)の実施形態は、所与のInteger精度に丸められた2つの小数の有理数除算の結果を計算するために、関数divD :: Integer -〉 Decimal -〉 Decimal -〉 Decimalを提供する。この丸めモードは丸めのうちの1つと同じである。DAML(商標)の実施形態は、関数remD :: Integer -〉 Decimal -〉 Decimal -〉 Decimalを用いて、この除算の剰余へのアクセスを提供する。divDとremDの関係は、1)すべてのprec x yに対して、y * divD prec x y + remD prec x y == xであり、2)すべてのprec x yに対して、abs (remD prec x y) 〈 abs yであり、3)すべてのprec x yに対して、sign (remD prec x y) = sign y* sign xであるという法則が当てはまるような関係である。

【００３４】

DAML(商標)の実施形態は、関数fromInteger :: Integer -〉 DecimalおよびtoInteger :: Decimal -〉 Integerを用いた、DecimalからIntegerへの、およびIntegerからDecimalへの変換をサポートし、toInteger dはround 0 dの結果を変換する。DAML(商標)の実施形態は、すべての実行モデルが内部的に、固定されているが実装形態固有の精度で、RelTimeをDecimalとして表現すると仮定する。したがって、DAML(商標)の実施形態は関数fromRelTime :: RelTime -〉 DecimalおよびtoRelTime :: Decimal -〉 RelTimeを用いて、DecimalからRelTimeへの、およびRelTimeからDecimalへの変換をサポートし、fromRelTimeはRelTimeに対応するDecimal秒数を与え、toRelTime dはRelTimeの実装形態固有の精度に所与の秒数dを丸める。これは、すべてのdtに対してtoRelTime (fromRelTime dt) == dtであることを示唆することに留意されたい。

【００３５】

DAML(商標)の実施形態は、すべてのプリミティブ型に対する全順序を次のようにサポートする。Bool: False 〈 Ture。Text:実装形態により定義される。Integer:整数の全順序。Decimal:小数の全順序。Party:実装形態により定義される。Time:絶対的な時間が直線の時間軸に沿って順序付けられる。より正式には、∀ t1, t2 ⇔ Time, t1 〈 t2 = (t1 - t2) 〈 toRelTime 0 1である。RelTime:相対的な時間がそのDecimal表現に従って順序付けられる。ContractId:実装形態により定義される。

【００３６】

DAML(商標)の実施形態は、a == b ⇔ not (a 〈 b ∨ b 〈 a)であるようなすべてのプリミティブ型に対する等式をサポートする。DAML(商標)の実施形態は、Haskellと同じ中置演算子を使用しており、たとえば、DAML(商標)の実施形態は、等しいこと、等しくないこと、以下、以上、より小さい、より大きいをそれぞれ表記するために、==、/=、〈=、〉=、〈、〉を使用する(Haskell Report 2010の9章参照)。加えて、DAML(商標)の実施形態は、演算子(～) :: ContractId -〉 Contract -〉 Boolをサポートし、ここでcoid～coは、coidが台帳の中のアクティブな契約を参照するcontract-idであることと、このアクティブな契約がcoのインスタンスであることとを意味する。

【００３７】

DAML(商標)の実施形態は、(〈〉) :: Text - 〉 Text -〉 Text演算子を使用したText値の連結をサポートする。DAML(商標)の実施形態は、プリミティブ型の任意の値を実装形態により定義される方式でテキスト値に変換する、多重定義された演算toTextをサポートする。DAML(商標)の実施形態はまた、任意の型の任意の値を実装形態により定義される方式でテキスト値に変換する、show演算をサポートする。これは、scenarioの開発の間にデバッグのために使用される。

【００３８】

DAML(商標)の実施形態は、(+) :: Time -〉 RelTime - 〉 Timeが時間の点を相対的な時間だけシフトすることを表記し、(-) :: Time -〉 Time -〉 RelTimeが第1の引数と第2の引数との間の時間差の計算を表記するように、+および-演算子を多重定義する。これらの演算子の間の関係は、すべてのt1, t2 :: Timeに対して、t1 + (t2 - t1) =

= t2となるような関係である。

【００３９】

Recordへのアクセスは、JavaScriptのように括弧を使用して書かれる。たとえば、表現r["x"]は、record rの中の「x」とラベリングされたフィールドにアクセスすることを試みる。Recordの値は、JavaScriptのように波括弧およびリテラル文字列を使用して構築される。たとえば、{"x":1, "y":2}は、2つのフィールド「x」および「y」を伴うrecordである。

【００４０】

DAML(商標)の実施形態は、Haskellのようにラムダ抽象を使用するが、常に型とアノテートされ、たとえば、＼(a :: Integer) (b :: Integer) (c :: Text) -〉 toText (a + b) 〈〉 cである。関数適用はHaskellのように並置を使用して書かれる。これは左側結合であり、すべての中置演算子より強く結合する。

【００４１】

DAML(商標)の実施形態は、Haskellのように非再帰的なlet束縛を使用するが、複数の束縛はセミコロンによって分離されなければならず、たとえば、let x = 4; y = 5; in x + yである。正式な構文は、表現において、let var1 = expr1; var2 = expr2; ... varN = exprNにより与えられる。再帰的なlet束縛は許可されない。

【００４２】

DAML(商標)の実施形態は、Haskellのように、遅延評価されるif-then-else分岐表現を使用する。正式な構文は、if condition then expression1 else expression2である。たとえば、if owner == 'ACME' then "sell" else "buy"である。

【００４３】

DAML(商標)は、どの合意がいつどのような制約のもとで有効になるかを記述することがモデルの主な目的であるという仮定に基づく。これらの合意は、現実世界における影響との、台帳の中のモデルのつながりを形作る。合意の意味は、書類による契約において行われるように、人の解釈によって、かつ任意選択で適用可能な法体系の文脈で、決定され得る。

【００４４】

入門的なDAML(商標)の実施形態は、合意、選択肢、およびモデルを含む。選択肢、台帳更新、およびモデルの正式な構文ならびにセマンティクスが与えられる。代替的なDAML(商標)の実施形態は、このモデリング言語の理解を深めるのを助けるためのさらなる例を与える。

【００４５】

ある例示的な実施形態では、DAML(商標)の実施形態は、agreeというキーワードを使用して合意を規定し、その構文は、party 1, ..., partyN agree textOfAgreementである。たとえば、'UBS', 'Alice' agree "'UBS' deposits 100 CHF on account 'CH42 1234 5'は、この合意が有効になる時点で、'UBS'が口座'CH42 1234 5'に100スイスフラン(CHF)を預けなければならないという、'UBS'と'Alice'との間での合意を表記することが意図されている。

【００４６】

DAML(商標)の実施形態は、ラムダ抽象を使用して、そのような合意を合意のテンプレートに変換する。たとえば、deposit = ＼(obligor :: Party) (owner :: Party) (amount :: Integer) (account :: Text) -〉 obligor, owner agree toText obligor 〈〉 "deposits" 〈〉 toText amount 〈〉 "CHF on"' 〈〉 account 〈〉 "'.";という定義は、上記の合意をdeposit 'UBS' 'Alice' 100 "CH42 1234 5"と書くことを可能にする。

【００４７】

合意のテキストによる記述が、モデルが解釈される法体系の文脈において正確であることを確実にすることは、モデルを書く人に任されている。モデル言語の観点からは、DAML(商標)の実施形態は、合意の法的なテキストを解釈せず、関係者間の1つ1つの合意を

これらの関係者の義務と見なす。

【００４８】

DAML(商標)の実施形態は、関係者に選択肢を与えることによって関係者の許可をモデリングする。たとえば、'Alice' chooses account :: Text then deposit 'UBS' 'Alice' 100 accountという選択肢は、'UBS'が100CHFを預けなければならない口座を'Alice'が選択できることを意味する。モデルは、(場合によっては複数の)関係者に対して同時に利用可能な選択肢のグループである。たとえば、iouSellSettle = ＼(obligor :: Party) (owner :: Party) (amount :: Integer) -〉 await {"settle": owner chooses account :: Text then deposit obligor owner amount account, "sell": owner chooses newOwner :: Party then iouSellSettle obligor newOwner amount};は、清算または売却され得るCHFにおけるI-owe-youモデル(IOU)のテンプレートである。モデルiou 'Alice' 'UBS' 100は、'UBS'が'Alice'に100CHFの借りがあることの、具体的なIOUモデリングを表記する。

【００４９】

上記のiouSellSettleの例は、モデルがどのように規定されるかについての直観を与える。以下では、DAML(商標)の実施形態は、選択肢およびモデルの構文とセマンティクスの両方に対する正確な定義を与える。これらの定義は説明するには非自明であり、それは、それらが、DAML(商標)の実施形態において台帳更新と呼ばれる第3の概念を介して相互に再帰的であるからである。直観的には、これらの3つの概念は次のような関係がある。1．モデルは、相互に排他的な選択肢のグループである。2．選択肢は、ある特定の時間にある特定の種類の台帳更新を実行するためにある特定の関係者に与えられた許可である。3．台帳更新は、新しい合意およびモデルを作成し、既存のモデル上で選択肢を実行し、既存のモデルを削除し、または特定の時間に台帳の状態についてのステートメントをアサートすることによって、どのように台帳を更新するかの記述である。

【００５０】

以下では、DAML(商標)の実施形態はまず、モデル、選択肢、および台帳更新の構文を与えて説明する。次いで、DAML(商標)の実施形態は、台帳の状態がどのように表現されるか、およびDAML(商標)の実施形態が台帳更新を使用して状態間をどのように遷移するかを説明する。DAML(商標)の実施形態は、ContractIdBinder named contractNameExpr {"choiceNamel":choice1 , ... ,"choiceNameN":choiceN}として特定される以下の構文.awaitを使用して、モデルを規定する。そのような表現は、choiceName1からchoiceNameNとラベリングされたN個の選択肢をこれらの選択肢の制御権のある関係者に与える、モデルを規定する。変数contractIdBinderは、型がContractIdであり、選択肢1からNの定義において、ならびにcontractNameExpr表現において、具体的なモデルインスタンスの識別情報を参照するために使用され得る。ThecontractIdBinderは、固有のcontract-idを引数として複数の選択肢に与えることによって複数の選択肢を関連付けるのに有用である。variablecontractNameは、型がTextであり、デバッグを容易にするためのモデルに対する任意の名前である。named contract NameおよびidentifiedasasContractIdBinderの部分は、必要ではない場合省略されてもよい。

【００５１】

DAML(商標)の実施形態は、次の構文を使用して選択肢を規定する。controllingPartyExpr chooses valueBinder1 :: Type1, ..., valueBinderL :: TypeL at choiceTimeBinder such that booleanChoiceCondExpr then updateExpr.そのような選択肢の表現は、未来のある時間において要求される型の値を選択するためにcontrollingPartyExprにより表記される関係者に選択肢が与えられ、これらの値およびそれらが選択された時間がbooleanChoiceCondExprを満たす場合、updateExprにより表記される台帳更新が実行されることを規定する。この選択肢は、この台帳更新の実行が成功する場合にのみ成功する。

【００５２】

台帳更新は、取引を反映し、データ構造として記憶され、以下の組み込み関数のうちの1つを使用して、または更新ブロックを使用して構築される。create :: ContractOrAgreement -〉 Update. delete :: ContractId -〉 Contract -〉 Update. exercises :: Party -〉 Label -〉 Any -〉 .... -〉 Any -〉 ContractId -〉 Update. assert :: Bool -〉 Update. pure :: Any -〉 Update.直観的には、DAML(商標)の実施形態は、モデルまたは合意を作成するためにcreateを使用し、アクティブモデルを非アクティブにするためにdeleteを使用し、アクティブモデル上で選択肢を実行するように関係者に強いるためにexercisesを使用し、台帳の現在の状態についてのステートメントをアサートするためにassertを使用し、値を返すだけで台帳を変更しない台帳更新を構築するためにpureを使用する。

【００５３】

更新ブロックは、複数の連続的な台帳更新を1つのアトミックな台帳更新として実行することを可能にする。更新ブロックは次の構文を使用して規定される。update [updateStatement1 〜〉 binder1 , ..., updateStatementN 〜〉 binderN, lastUpdateStatement]. DAML(商標)の実施形態は、個々の更新ステートメントの実行の結果を名付けるために、曲がった矢印〜〉を使用する。これらの名前が次いで、これらの結果を参照するために後の更新ステートメントにおいて使用され得る。

【００５４】

以前のiouSellSettleの例に示されるように、モデルは再帰的であり得る。そのような再帰を規定する方法は、セクション「DAML(商標)プログラム」において後で説明されるように、名付けられたトップレベル定義の使用を介する。DAML(商標)の実施形態は、糖衣構文を使用して、モデル規定する構文的な多重定義を減らす。DAML(商標)の実施形態は、モデルおよび合意が更新として使用されることを可能にし、DAML(商標)の実施形態は、更新の記録が更新として使用されることを可能にする。

【００５５】

モデルcを更新として使用するとき、その効果はcreate cの効果と同じである。DAML(商標)の実施形態は、モデルを作成することが最も一般的な更新アクションであるので、この機能を追加する。record {"11":upd1, ..., "IN":updN}を更新として使用するとき、その効果はupdate [upd1 〜〉 v1 , ..., updN 〜〉 vN , pure {"l1":v1, ..., "lN":vN}]の効果と同じである。DAML(商標)の実施形態は、しばしば多数の中間更新アクションの結果を返す必要があるので、この機能を追加する。

【００５６】

DAML(商標)の実施形態は、モデル、選択肢、および台帳更新のセマンティクスを遷移系として説明する。この遷移系の状態は台帳であり、台帳はContractIdから有効なモデルおよび合意のログへの連想配列である。遷移は台帳更新の解釈によって与えられる。DAML(商標)の実施形態は、台帳がどのように変更されるかということと、どのような結果値が台帳更新によって返されるかということの両方を、各解釈に対して規定することに留意されたい。DAML(商標)の実施形態は、更新ブロックを解釈するときにこれらの結果を必要とし、後の台帳更新はそれより前の台帳更新の結果を参照することができる。

【００５７】

DAML(商標)の実施形態は、agreementExprを評価し、それが実際にparty1, ..., partyN agree legalTextの形の合意であることを確認し、次いでこの合意が今有効であることを記録することによって、create agreementExprという表現を解釈する。この台帳更新の結果はこの合意そのものである。

【００５８】

DAML(商標)の実施形態は、contractExprを評価することによって、create contractExprという表現を解釈する。この評価がモデルcoについて成功する場合、DAML(商標)の実施形態はこのモデルcoを新たに割り振られたcontract-id coidのもとで記憶する。この台帳更新の結果はcontract-id coidである。

【００５９】

DAML(商標)の実施形態は、contractIdExprとcontractExprの両方をまず評価することによって、delete contractIdExpr contractExprという表現を解釈する。この評価がリテラルcontract-id coidおよびモデルcoについて成功する場合、DAML(商標)の実施形態は次いで、モデルcoに等しいアクティブモデルをcoidが特定することを確認する。特定する場合、DAML(商標)の実施形態は台帳からcoidを除去する。DAML(商標)の実施形態は、このdeleteによりどの関係者が影響を受けるかについての静的解析を可能にするために、deteleの一部としてモデルcoが規定されることを要求することに留意されたい。

【００６０】

DAML(商標)の実施形態は、partyExpr exercises "choiceLabel" with choice Value1, ... choiceValueN on coidという形の台帳更新を次のように解釈する。DAML(商標)の実施形態はまず、partyExprを評価する。この結果がリテラルparty name actorをもたらす場合、DAML(商標)の実施形態は、coidと関連付けられるアクティブモデルを探す。coidがアクティブモデルcoを特定する場合、DAML(商標)の実施形態は、それを非アクティブであるものとしてマークする。次いで、DAML(商標)の実施形態は、coの中の"choiceLabel"によって特定される選択肢を探す。actorが選択肢の制御権者に等しい場合、DAML(商標)の実施形態はこの選択肢を実行する。たとえば、DAML(商標)の実施形態はまず、選択肢の条件およびそのフォローアップの台帳更新の両方を、所与のchoiceValuesおよび現在時刻を用いてインスタンス化する。次いで、DAML(商標)の実施形態は、選択肢の条件を確認し、それが成功する場合、次いで選択肢のフォローアップを解釈する。exercisesを使用して構築された台帳更新の結果は、選択肢のフォローアップの結果である。この解釈は潜在的に再帰的であるが、終了することが保証されていることに留意されたい。

【００６１】

DAML(商標)の実施形態は、booleanExprを評価することによってassert booleanExprという表現を解釈し、次いで、結果がTrueであるかどうかを確認する。Trueである場合、解釈は成功する。そうではない場合、解釈は失敗する。assert ledger-updateの結果は空のrecord{}である。

【００６２】

DAML(商標)の実施形態は、xを評価してそれをこの台帳更新の結果として返すことによって、pure xという表現を解釈する。したがって、DAML(商標)の実施形態は、ある特定の結果を返す副作用のない台帳更新を構築するために、pureを使用することができる。

【００６３】

DAML(商標)の実施形態は、以前のupdateStatementの結果をbinderの代わりに用いた後でupdateStatementを次々に解釈することによって、更新ブロックupdate [updateStatement1 〜〉 binder1 , ..., updateStatementN 〜〉 binderN , lastUpdateStatement]を解釈する。更新ブロックの結果は、lastUpdateStatementの結果である。更新ブロックの解釈は、そのステートメントのいずれかが失敗すると失敗し、更新ブロックが成功する場合にのみ、台帳上でのすべての効果が適用される。したがって、更新ブロックは、アトミックな合成式の台帳更新を構築することを可能にする。

【００６４】

また、他のモデルテンプレートを組み合わせるモデルテンプレートを定義することもできる。たとえば、定義option = \(controller :: Party)(tlb :: Time)(tub :: Time)(next :: Contract) -〉 await{"exercise": controller chooses at t such that tlb 〈= t && t 〈= tub then next};は、optionと呼ばれるコンビネータを与え、これは、関係者がある特定の時間間隔の間にモデルに入ることを可能にする。

【００６５】

すべての上の例において、選択肢は、0個から多数の新しいモデルの作成につながるだけである。分割または統合され得るIOUの以下の例は、中に統合されたモデルをアトミッ

クに削除(非アクティブ化)して、より多くの量の新しいIOUモデルを作成するために、「merge」選択肢において更新ブロックをどのように使用するかを示している。iouChf = \(obligor :: Party)(owner :: Party)(amount :: Integer) -> await{"settle": owner chooses account :: Text then deposit obligor owner amount account , "sell": owner chooses newOwner :: Party then iouChf obligor newOwner amount , "split": owner chooses newAmount :: Integer such that 0 < newAmount && newAmount < amount then {"iou1": iouChf obligor owner newAmount , "iou2": iouChf obligor owner (amount - newAmount)}, "merge": owner chooses otherIou :: ContractId, otherAmount :: Integer then update [delete otherIou (iouChf obligor owner otherAmount), iouChf obligor owner (amount + otherAmount)]};.

【００６６】

DAML(商標)の実施形態はまた、特定の関係者が決断を実行することを必要とする選択肢を使用することによって、他のモデルへの変更を調整するためにモデルを使用することができる。たとえば、以下のテンプレートは、ある時間までに支払いの所有権を移転するように支払人に要求するために使用され得る。

```
      --スイスの法廷を前提に2つの関係者間で協議されなければならない契約違反のた
--めのテンプレート
      contractBreachedBy =
       \(defendant :: Party)(plaintiff :: Party) ->
          defendant, plaintiff agree
           toText defendant <> "has breached the contract, and" <>
           toText plaintiff <> "can sue" <> toText defendant <>
           "in any court of Switzerland according to Swiss law."
      ;
      --上で定義されたようなIOUの所有権の移転を介してある時間内に支払いを要求す
--るモデルのテンプレート
      mustPayIouUntil =
       \(payer :: Party)
        (payee :: Party)
        (payment :: Contract)
        (maxPayTime :: Time)
       ->
        await
         {"pay":
          payer chooses paymentId :: ContractId
          such that
           --'paymentId'がアクティブな'payment'モデルを参照することの確認
           paymentId〜payment
         then payer exercises "sell" with payee on paymentId
         --支払人に与えられた支払いのための時間の後で懲罰の項/選択肢が受取人に対
--して利用可能になる
        ,"breach":
         payee chooses at tbreached
         such that
          maxPayTime <= tbreached
         then contractBreachedBy payer payee
        } ;
```

次いで、iouChf paymentsを使用するコンビネータが、次のように定義され得る。

```
      payInChfUntil =
```

```
      \(payer :: Party)
       (payee :: Party)
       (obligor :: Party)
       (amount :: Integer)
       (maxPayTime :: Time)
     ->
      mustPayIouUntil payer payee (iouChf obligor payer amount) maxP
ayTime;
```

【００６７】

DAML(商標)の実施形態は、トップレベル定義のグループをDAML(商標)プログラムと呼ぶ。DAML(商標)の実施形態は現在、a = b c = dがa = b cとして構文解析されてその後の=dに対する構文解析が失敗するという、局所的な構文解析の曖昧さにより、セミコロンを使用してこれらの定義を分離している。したがって、DAML(商標)の実施形態は、セミコロンの必要性を軽減するために、各トップレベル定義に対してa = b; c = dのように末尾のセミコロンを使用する。

【００６８】

Scenarioは、台帳に記憶されている1つまたは複数のモデルを介して複数の関係者がどのように対話するかの記述である。DAML(商標)の実施形態は、言語にscenarioのための特別な表記を含み、それは、それらがモデルテンプレートのための不可欠なドキュメンテーションおよびテストケースとなるからである。

これは、以前のセクションにおけるIOU定義のための例示的なscenarioである。

```
days = \(x :: Integer) -> toRelTime (fromInteger (x * 24 * 60 * 60));
createAndSettleIou =
 scenario
  [commit (iouSellSettle 'UBS' 'UBS' 100) ~> ubsIou
  ,'UBS' commits 'UBS' exercises "sell" with 'Alice' on ubsIou ~>
aliceIou
  ,assert (aliceIou ~ iouSellSettle 'UBS' 'Alice' 100)
  ,pass (days 10) ~> now
  ,'Alice' commits 'Alice' exercises "settle" with "CH42 1234 5" on
aliceIou ~> settled
  ,assert (settled == (
    'UBS', 'Alice' agree "'UBS' deposits 100 CHF on 'CH42 1234 5'."))
  ];
```

【００６９】

更新ブロックに関して、DAML(商標)の実施形態は、scenarioアクションの結果を拘束するために曲がった矢印を使用し、scenarioの結果はその最後のステップの結果である。上の例における拘束された変数の型は、ubsIou :: ContractId; aliceIou :: Record; now :: Time; settled :: Record;である。aliceIou recordの形は{"iou": ContractId}であり、清算されるrecordの1つは{"settle": agreement}であることに留意されたい。これは、上のiou定義の"sell"および"settle"選択肢におけるフォローアップのラベリングから決定される。

【００７０】

DAML(商標)の実施形態は、scenarioステップの結果のrecordへとパターンマッチングを行うことができる。たとえば、'Alice' exercises "settle" with "CH42 1234 5" on aliceIou["iou"] ~> {"settle" : agreement}である。これは、このステップの結果のrecordの"settle"エントリを変数名agreementに拘束する。DAML(商標)の実施形態は、recordを任意の深さへとパターンマッチングすることができ、たとえば、これは有効なパターン{"foo" : {"bar" : {"baz" : varName}}}である。

【００７１】

パターンは完全である必要はない。すなわち、結果のrecordの中に存在するラベルは、パターンにおいて省略され得る。結果のrecordの中に存在しないラベルに対するパターン一致はエラー(たとえば、解釈される場合ランタイムエラー)を引き起こす。変数名のシャドーイングは、それらのラベルキーの順序により決定される。前の例では、変数varNameのラベルキーは["foo", "bar", "baz"]である。この変数は、["foo", "bar"]のラベルキーを用いて変数名をシャドーイングする。この例{"a" : varName, "b" : varName}では、varNameは["b"]によりアクセスされるエントリを拘束し、それはキー["b"]がキー["a"]の後に来るからである。

【００７２】

あるscenarioのデフォルトの解釈は次の通りである。空の台帳から開始して、scenarioのステップが順番に実行される。

【００７３】

台帳更新をコミットする。party1, ..., partyN commit updateExprという形の表現は、関係者party1, ..., partyNが、byupdateExprと表記される台帳更新をコミットすることを望んでいることに共同で合意することを表記する。updateExprが現在の台帳上での状態遷移として解釈されることに成功できる場合、これは成功する。DAML(商標)の実施形態は、コミットに合意する関係者を明示的に指定するものを必要とし、それは、それらの関係者だけが、解釈された台帳更新において義務を負うことが許される関係者であるからである。代替的な実施形態は、義務を負うことができる参加者という概念を規定し得る。

【００７４】

時間を制御する。pass relTimeExpr 〜〉 newTimeBinderのステップは、現在のscenario時間をrelTimeExprだけ進め、新しいscenario時間をnewTimeBinderに拘束する。この機能を使用して、scenarioの初期時間をpass(toRelTime 1.0)〜〉 nowのように決定することができる。

【００７５】

予想を表現する。assert booleanExprのステップは、ブール表現を評価し、その表現が偽である場合scenarioを失敗させる。そうでない場合には効果を持たない。キーワードmustFailは、あるscenarioステップが失敗することになっていることを示すようにそのscenarioステップを修飾することができる。そのようなステップは、内部ステップが失敗しない場合に失敗し、内部ステップが失敗する場合に失敗しない。たとえば、モデルを2回清算することができないことを述べることは、理にかなっている。

```
mustFailExample = scenario
 ['UBS' commits (iouSellSettle 'UBS' 'Alice' 100) 〜〉 iouId
 ,'Alice' commits 'Alice' exercises "settle" with "CH12 345 6" on iouId
 ,mustFail ('Alice' exercises "settle" with "CH12 345 6"on iouId)
 ]
;
```

【００７６】

デバッグする。trace textExprのステップは、Text表現を評価し、台帳をどのような方法でも変更しないがステップの記述に評価されるtextを含む、scenarioステップを作成するので、これは、scenarioをデバッグしtoText関数を使用して値を調べるために使用され得る。例: trace ("gergely's trace" 〈〉 toText (42 + 123))

【００７７】

記述アノテーションを使用したテキスト表現を用いて、あらゆる表現をアノテートすることができる。これらは、{@ DESC textExpr @} someOtherExpression andItsArgumentsという構文を使用して書かれる。記述アノテーションは、可能な限り右の方に結び付き、関数自体をアノテートしたいだけである場合には、括弧を使用しなければならない。たとえば、

```
＼(f :: Integer -〉 Integer) (arg :: Integer) -〉
```

```
        ({@ DESC "the function's description" @} f)
        ({@ DESC "the argument's description" @} arg)
```

【００７８】

DAML(商標)の実施形態はこの方法を使用し、それは、モデルまたは選択肢全体をアノテートする一般的な場合において、必要な括弧がより少ないからである。同じ表現に対する複数のアノテーションの場合、最も内側のアノテーションが保たれ、他のアノテーションは無視されることに留意されたい。具体的には、DAML(商標)の実施形態は、GUIの目的で法的拘束力がなく人が読めるテキストを使用してモデルを略記するために、記述アノテーションを使用する。たとえば、DAML(商標)の実施形態は、named iouモデルを次のように導入することができる。

```
      iou =
       \(obligor :: Party) (owner :: Party) (amount :: Integer)
       ->
        {@ DESC
         toText obligor <> " --(" <> toText amount <> ")--> " <> toT
ext owner
        @}
       await
        {"sell": owner chooses newOwner :: Party then {"iou": iou obligor
newOwner amount }
        ,"settle":
           owner chooses account :: Text
           then deposit obligor owner amount account
        }
      ;
      traceExample =
       scenario
        ['Bank' commits (iou 'Bank' 'Alice' 1 ) ~> alice
        ,'Bank' commits (iou 'Bank' 'Bob' 2) ~> bob
        ,trace ("Bob's contract: " <> toText bob)
        ]
      ;
```

【００７９】

awaitキーワード上での記述アノテーションは、インタプリタによって覚えられ、最終的な台帳を印刷するときに使用される。たとえば、あるscenarioを完了するとき、DAML(商標)の実施形態はこの出力を有し得る。

```
      final ledger:
      [contract 0c created at 1970-01-01T00:00:00Z 'Bank' --(1)--> 'Alice'
         ["sell":
          'Alice'
          chooses newOwner5 :: Party
          then {"iou": iou newOwner5 'Bank' 1}
         ]
      ,contract 1c created at 1970-01-01T00:00:00Z 'Bank' --(2)--> 'Bob'
         ["sell":
          'Bob'
          chooses newOwner6 :: Party
          then {"iou": iou newOwner6 'Bank' 2}
         ]
      ]
```

【００８０】
ここで、DAML(商標)の実施形態は、BankがAliceに1ドルの借りがあり、Bobに2ドルの借りがあることを示す、IOUの短い記述を理解することができる。
【００８１】
多くのモデルは複数の関係者による選択を必要とするが、これらの選択が行われる順序には無関心である。そのようなモデルは、ここまでに説明された言語の特徴を用いて規定され得る。しかしながら、これらの規定のサイズは、選択を行わなければならない関係者の数に対して指数関数的に増大し、それは、DAML(商標)の実施形態がすべてのあり得る選択の順序を数え上げなければならないからである。そのような指数関数的な膨張を避けるために、DAML(商標)の実施形態は、複数の関係者による並列の選択を明示的にサポートする。
【００８２】
DAML(商標)の実施形態は、並列の選択に対するサポートを次のように説明する。DAML(商標)の実施形態はまず、導入のための例を与える。DAML(商標)の実施形態は次いで、並列の選択と、個々の並列の選択に対する判断との両方のための、正式な構文を規定する。最後に、DAML(商標)の実施形態は、並列の選択のセマンティクスを説明する。
【００８３】
以下の例の契約は、2つの関係者の間で2つの売却可能契約を交換するためのオプションに対する提案をモデリングする。

```
optionalSwap =
 \(alice :: Party)
  (aliceGood :: Contract)
  (bob :: Party)
  (bobGood :: Contract)
-> 
 await
  {"swap":
   { | "alice": alice chooses ca :: ContractId such that ca ~ aliceGood
   ,"bob": bob chooses cb :: ContractId such that cb ~ bobGood
   | }
   then
   {"bob's": alice exercises "sell" with bob on ca
   ,alice's": bob exercises "sell" with alice on cb
   }
  ,"alice cancels" : alice chooses then {}
  ,"bob cancels" : bob chooses then {}
  };
```

【００８４】
前の例とは対照的に、awaitの選択肢のフォローアップは2つの並列な選択ステップによって保護される。第1のステップは彼女の商品をプレコミットするというaliceの選択をモデリングし、第2のステップは彼の商品をプレコミットするというbobの選択をモデリングする。これらの2つのステップはプレコミットにすぎず、それは、aliceとbobの両者が、相手がその商品を提供しない限り、自身の個々の商品について選択をまだ実行できるからである。これは交換のためのオプションに過ぎないので、DAML(商標)の実施形態は、aliceとbobの両者に、交換を取り消すための選択肢を与える。aliceまたはbobが「交換」の選択の自分自身の部分を実行しただけである限り、これらの選択肢の両方が利用可能なままである。商品の実際の交換は、相手がプレコミットの選択を行うとアトミックに発生することに留意されたい。
並列の選択ステップについての選択肢の正式な構文は次の通りである。

```
{ | "choiceStep1":
```

```
        controllingPartyExpr1
        chooses
          valueBinder1_1 :: Type1_1, ..., valueBinder1_L :: Type1_L at c
hoiceStepTimeBinder1
        such that
         booleanChoiceStepCondExpr1
        then followUpResultBinder1 〈- followUpExpr1
      , ...
      ,"choiceStepN":
        controllingPartyExprN
        chooses
          valueBinderN_1 :: TypeN_1, ..., valueBinderN_M :: TypeN_M at
choiceStepTimeBinderN
        such that
         booleanChoiceStepCondExprN
        then followUpResultBinderN 〈- followUpExprN
      ¦}
      such that
       booleanChoiceCondExpr
      then followUpExpr
```

【００８５】

値binderの範囲は、選択ステップごとの拘束される値を対応する選択ステップごとのブール条件において参照でき、すべての拘束される値を選択肢のブール条件およびすべての選択肢のフォローアップにおいて参照できるような範囲である。選択ステップのグループは空であってはならず、すべてのステップが異なるようにラベリングされなければならないことに留意されたい。DAML(商標)の実施形態は、recordと並列の選択肢とが明確に区別可能であることを確実にするために、異なる種類の括弧、すなわち{¦and¦}の変形を使用することを選んだ。これにより、DAML(商標)の実施形態は、recordを後で一般化することの負担がない状態に構文を保つ。

【００８６】

DAML(商標)の実施形態は、選択肢のラベルを参照できるだけではなく選択ステップのラベルも参照できるように、判断の正式な構文を拡張する。これにより、選択ステップのうちの1つを実行するという関係者の判断を規定することが可能になる。具体的な構文は、partyExpr exercises "choiceLabel" "choiceStepLabel" with choiceStepValue1, ..., choiceStepValueN on contractIdExpr.である。たとえば、以下のscenarioは、iouのoptionalSwapを作成し、それを実行する。

```
      optionalSwapTest =
       scenario
        ['UBS' commits (iouSellSettle 'UBS' 'Alice' 100) 〜〉 aliceIou1
        ,'CS' commits (iouSellSettle 'CS' 'Bob' 160) 〜〉 bobIou1
        ,commit
          (optionalSwap 'Alice' (iouSellSettle 'UBS' 'Alice' 100)
                 'Bob' (iouSellSettle 'CS' 'Bob' 160)
          ) 〜〉 optSwapId1
        ,'Bob' commits 'Bob' exercises "swap" "bob" with bobIou1 on
         optSwapId1 〜〉 optSwapId2
        ,'Alice' commits 'Alice' exercises "swap" "alice" with aliceIou1 o
n optSwapId2 〜〉 {"alice's": aliceIou2
                 ,"bob's": bobIou2
                 }
```

```
        ,assert (aliceIou2 ～ iouSellSettle 'CS' 'Alice' 160)
        ,assert (bobIou2 ～ iouSellSettle 'UBS' 'Bob' 100)
        ];
```

【００８７】

並列の選択を実行するセマンティクスは次の通りである。単一の選択ステップにより保護される選択肢cは、その条件が選択ステップの条件およびcの条件の組合せであるような、普通の選択肢と同じように振る舞う。1つより多くの選択ステップにより保護される選択肢について、N個の選択ステップによって保護される選択肢cの"step_i"で、判断actor exercises "choice Label""step_i" with v1, ..., vN at time nowを実行する。

```
      { ¦ "step_1": step_1,
      ,...
      ,"step_i":
       ctrl chooses x1 :: type1, ..., xN :: typeN at t such that choiceStepCond
      ,...
      ,"step_N": step_N
       ¦ }
      such that
       choiceCond
      then followUps
```

【００８８】

これは次のように機能する。DAML(商標)の実施形態はまず、actorがstep_iの制御権を持つかどうかを確認し、次いで、DAML(商標)の実施形態は、値v1, ..., vNの型が予想される型と一致することを確認し、最後に、DAML(商標)の実施形態は、step_iの選択ステップ条件が満たされるかどうかを確認する。これらの確認のすべてが成功すると、DAML(商標)の実施形態は現在のモデルを削除し、新しいモデルを作成して、この新しいモデルが選択ステップstep_iに対する判断を記録する。

【００８９】

より具体的には、モデルcが上記のoptionalSwapTestscenarioにおいてoptSwapId1により指し示されるモデルであると仮定しよう。そうすると、時間tにおいて実行されるステップ'Bob' exercises "swap" "bob" with bobIou1 on optSwapId1 ～〉optSwapId2が、optSwapId1を非アクティブなものとしてマークし、c after [exercising "swap" "bob" at t with bobIou1]を指し示す新しいcontract-id optSwapId2を導入し、ここでafterはモデルに適用されなければならない未解決の判断のリストをマークする特別なキーワードである。

【００９０】

代替的なDAML(商標)の実施形態は、DAMLの入力構文において未解決の判断の規定を可能にする。これらの特徴は、基準のセマンティクスにおいて完全に規定され実装される。しかしながら、それらは必ずしもすべての実行モデルにおいてサポートされない。したがって、DAML(商標)の実施形態は、どの実行モデルにおいてそれがサポートされるかを、各特徴に対して述べる。

【００９１】

たとえば、利害関係者の追認を伴うHyperLedgerは現在、Point-Wise ObservablesまたはTabular Contract Query LanguageというDAML(商標)の特徴をサポートしない。金融の契約はしばしば、株式の終値または配当支払いの告知などの、外部の公開の評価値に依存する。DAML(商標)の実施形態は、そのような外部の公開の評価値を観測可能データと呼ぶ。観測可能データを形式化するための多くのモデルがある。DAML(商標)の実施形態は、daml-design-considerations.md文書においてそれらのモデルのうちのいくつかを探究している。ここで、このセクションでは、DAML(商標)の実施形態は、1.タイムスタンプが押されたイミュータブルなデータ値の固有の名称を有するデータフィードを各関係者が公表でき、2.モデルがこれらのデータフィードの点別の観測に

依存し得るような、単純なモデルに対するサポートを説明する。

【００９２】

DAML(商標)の実施形態は、これらのデータ値が台帳上で公表されることを義務付けない。代わりに、DAML(商標)の実施形態は、関係者が、データフィードの値を通信し、ある時点におけるある値をデータフィードが取得したことを証明することを可能にする、何らかの手段があることを仮定する。概念的には、データフィードの状態は、部分的に定義された区分定数関数にいずれの時間においても対応する。データフィードの状態が展開するにつれて、対応する関数はより定義されるようになるが、すでに定義された時点における値を決して変更しない。より正式には、DAML(商標)の実施形態は、型aの値のデータフィードの状態を、型Feed a = [(Time, a)]の空ではないリストとして表現することができ、ここで時点は厳密に単調である。feed = [(t_0, v_0), (t_1, v_1), (t_n, v_n)]を型Feedの空ではないリストとする。そうすると、対応する部分的に定義された区分定数関数は、時点tを以下にマッピングする関数である。

1. t_i 〈= t 〈 t_(i+1)となるような、ある時間における値(t_i, v_i)と(t_(i+1), v_(i+1))の連続するペアが存在する場合、v_i
2. t_n == tである場合、v_n
3. 上の2つの条件のいずれもが満たされない場合、定義されない

【００９３】

この定義は、フィードの中の最後の時間における値(t_n, v_n)だけが、t == t_nに対する対応する関数の値を定義することを示唆していることに留意されたい。すべてのt 〉 t_nに対して、対応する関数は定義されない。

【００９４】

前に述べられたように、DAML(商標)の実施形態は、分散型台帳の設定において使用されるべきデータフィードのための公表スキームを取り決めない。しかしながら、上の定義を選ぶ理由の1つは、それが単純な公表スキームを可能にするからである。すなわち、公表側の関係者は、以前の公表へのハッシュベースのリンクと一緒に新しい値に署名することによって、新しい値をデータフィードに公表することができる。たとえば、以下の定義は、2つの連続する時点t0およびt1 〉 t0においてデータフィードを公表することを例示する。

```
hash_0 = hash(t_0, v_0);
feed_0 = sign('PublisherPk', ("feedName", hash_0))
hash_1 = hash(hash_0, (t_1, v_1));
feed_1 = sign('PublisherPk', ("feedName", hash_1))
...
```

【００９５】

この例は、ハッシュhash_0およびhash_1に対応するバイト文字列が、公表されたメッセージに埋め込まれるか、またはIPFSのようにコンテンツ指定可能なファイルシステムを介して分配されるかのいずれかであると仮定する。そのような公表スキームの効率を改善するための多くの方法(たとえば、ある時点の値のブロックを構築すること、またはTESLAのように認証されたストリーム-ブロードキャストプロトコルを使用することなど)があることは明らかである。DAML(商標)の実施形態は、これらの効率改善の大半が、データフィードの定義可能性および不変性についての仮定と矛盾しないことを期待する。したがって、DAML(商標)の実施形態は、この文書においてデータフィードのためのこれらの公表スキームの詳細をさらに追求することはしない。

【００９６】

モデル言語の基準の実装形態では、DAML(商標)の実施形態は、データフィードのクエリと、新しいある時点の値の関係者固有のデータフィードへの公表とを管理するために、次の2つの関数を提供する。

```
observeFeedAt :: Text -〉 Time -〉 Any
publishToFeed :: Party -〉 Text -〉 Any -〉 Scenario Unit
```

【００９７】
DAML(商標)の実施形態は、公表者によって公表されたデータフィードfeedNameが時間tにおいて有する値をクエリするために、observeFeedAt publisher feedName tという表現を使用する。フィードの値がtにおいて定義されない場合、この表現の評価は失敗する。これは、tがフィードfeedNameの最初の公表されるデータ点の前である場合、または最後の公表されるデータ点の後である場合に当てはまる。
【００９８】
DAML(商標)の実施形態は、exprの値を公表者のフィードfeedNameに公表するscenarioステップを表記するために、publishToFeed publisher feedName exprという表現を使用する。この公表される値のタイムスタンプは現在のscenario時間であり、これはデータを未来に向かって公表できないことを示唆する。
【００９９】
以下の例は、observeFeedAtとpublishToFeedとの間の相互作用を示す。

```
seconds = \(t :: Integer) -> toRelTime (fromInteger t);
observeSixSmi = observeFeedAt 'SIX' "SMI";
publishToSixSmi = \(value :: Integer) ->
 scenario
  [pass (seconds 0) ~> now
  ,publishToFeed 'SIX' "SMI" value
  ,pure now
  ];
publicationAndObservationTest =
 scenario
  [publishToSixSmi 7000 ~> t0
  ,assert (observeSixSmi t0 == 7000)
  ,mustFail (assert (observeSixSmi(t0 - seconds 1) == 7000))
   --^'t0'の前は'SIX'の"SMI"フィードが定義されていないので失敗する
  ,mustFail (assert (observeSixSmi(t0 + seconds 1) == 7000))
   --^この時点では't0'に等しい現在のscenario時間の後は'SIX'の"SMI"フィードが
        --定義されていないので失敗する
  ,pass (seconds 1) ~> t1
  ,mustFail (assert (observeSixSmi (t0 + seconds 1) == 7000))
   --^'SIX'の"SIM"フィードの中の最後の公表された値がタイムスタンプ't0'にあ
     --るので失敗する
  ,publishToSixSmi 8000
  ,assert (observeSixSmi t1 == 8000)
   --ここで、現在のscenario時間において'SIX'の"SMI"フィードの値が固定された
     --のでこれは成功する。DAML(商標)の実施形態はこの明示的な固定を必要とし
     --、それは、それがなければ、ある特定の時点における観測される値が非確定
    --的に変化し得るからである。現在の解決法では、DAML(商標)の実施形態はこ
     --の非確定性を避け、それは、定義されない値がモデルの選択において観測さ
    --れ得ないからである。
  ];
```

【０１００】
1つ1つのフィードが1つ1つの時間において明示的に公表されなければならないという制約は、署名されたハッシュチェーンの1つ1つの拡張が明示的に公表されなければならない分散型台帳の見地から想起されたものである。しかしながら、上記のプリミティブでは、これはテストscenarioにおいて少し煩雑になることがある。したがって、DAML(商標)の実施形態は、代替的なDAML(商標)の実施形態が、すべてのフィードの最後の値を現在のscenario時間でタイムスタンプを押された値として公表する、関数fixAllFee

ds :: Scenario Unitを、そのような関数が必要であれば導入できることを見越している。

【０１０１】

モデルテンプレートは、データフィードにわたってパラメータ化され得る。上で説明されたデータフィードに対するサポートは最小限である。それでも、DAML(商標)の実施形態は、これらの外部のデータフィードに依存する選択肢を用いてモデルテンプレートを形式化するための高い表現性をそれが提供することを期待する。そのようなモデルテンプレートを形式化するための主要な考え方は、時間依存のレートを型Time -〉 Decimalの関数として表し、時間依存の条件を型Time -〉 Boolの関数として表すことである。そうすると、そのようなテンプレートの呼出者は、データフィードを調査するために所与の時間値を使用することができる。

【０１０２】

この決断は、言語における関数が完全には純粋ではないことを意味することに留意されたい。関数は、(～関数が原因で)台帳の状態に依存し、(observeFeedAt関数が原因で)データフィードの状態に依存する。これは、DAML(商標)の実施形態が厳密な検証セマンティクスを有するので許容可能であり、このことは、DAML(商標)実施形態がいつ表現を評価するかが常に明確であることを確実にする。代替的なDAML(商標)の実施形態のための評価セマンティクスは、アプリケーションおよび引数を熱心に評価し、飽和していないアプリケーション、await、または選択肢に遭遇すると停止し得る。

【０１０３】

Tabular Contract Query Languageは、契約ベースの台帳にクエリすることをサポートする。このクエリはデータ構造の構文上のパターンマッチングに基づき、DAML(商標)の実施形態は、このクエリを、契約ベースの台帳からの読取りにおいて中心的な役割を果たすものと考える。このセクションにおいて説明される言語の特徴は、文脈においてパターンマッチングがどのように振る舞うかに関する基礎として機能する。この理解が、GUIの開発および選択肢実行のための契約固有の自動化の開発において使用され得る。

【０１０４】

コンピュータシステム上で具現化される実行環境は、DAML(商標)において規定されるモデルを台帳に記憶する。DAML(商標)の実施形態は、台帳の中のアクティブな契約およびそれらのパラメータのために、そのようなデータベースをクエリするための手段を提供することを望む。たとえば、DAML(商標)の実施形態は、どれだけのアクティブな「IOU」契約を台帳が現在含んでいるかを、台帳にクエリし得る。または、obligorが'UBS'という名称の「IOU」契約がどれだけあるかをクエリし得る。GUIは、台帳の中のすべてのアクティブな「IOU」に対するクエリの結果を、表形式で次のように提示することを望み得る。

ContractId	CreatedAt	Amount	Obligor	Owner
0xffeeafa	2007-04-05T14:30Z	23	UBS	Bob
0xa123001	2016-03-02T12:00Z	1000	CS	Alice

【０１０５】

契約は固定されたデータスキームを使用して記憶されず、抽象構文木(AST)を表現するデータ構造として記憶されるので、そのような表を、台帳の基礎となるデータベース実装形態によって提供することができない。DAML(商標)の実施形態は、台帳の中のアクティブモデルのAST表現の構文上のパターンマッチングによって、所与のパラメータを伴うアクティブモデルに対する台帳をクエリするための能力を提供する。この概念は、例を見ることによって恐らく最も良く理解される。

```
testQuery =
 scenario
  ['Bank1' commits (iouSellSettle 'Bank1' 'Alice' 100) 〜〉 iou1
  ,'Bank1' commits (IouSellSettle 'Bank1 "Bob' 20) 〜〉 iou2
  ,'Bank2' commits (iouSellSettle 'Bank2' 'Bob' 40) 〜〉 iou3
```

```
      ,traceMatchingContracts (iouSellSettle ?obligor ?owner ?amount)
       --これは以下の追跡結果を出力し、ここで〈iouN〉は変数'iouN'の値を指す
       -- Found 3 matching contracts:
       -- 1. contract 〈iou1〉 with
       --   {"obligor":'Bank1'
       --   ,"owner":Alice
       --   ,"amount":100
       --   }
       -- 2. contract 〈iou2〉 with
       --   {"obligor":'Bank1'
       --   ,"owner":Bob
       --   ,"amount":20
       --   }
       -- 3. contract 〈iou3〉 with
       --   {"obligor":'Bank2'
       --   ,"owner":Bob
       --   ,"amount":40
       --   }
       -- DAML(商標)の実施形態はまた、いくつかの値を固定することで、等価性によりフィルタリングすることができる
      ,traceMatchingContracts (iouSellSettle ?obligor 'Alice' ?amount)
       -- これは次の追跡結果を出力する。
       -- Found 1 matching contracts:
       -- 1. contract 〈iou1〉 with
       --   {"obligor":'Bank1'
       --   ,"owner":Alice
       --   ,"amount":100
       --   }
      ];
```

【０１０６】

DAML(商標)の実施形態は、関数traceMatchingContractsに、あるscenarioにおいてインタプリタ台帳に対してクエリを行うための構文traceMatchingContracts contractPatternExprを提供する。contractPatternExprは、パターン変数を格納する表現である。パターン変数は、先頭の'?'の文字によって普通の変数と区別される。パターンの単純な例は、?a + 2である。このパターンは、構文的には3 + 2 for ?a = 3と一致するが、このパターンは3 + 9とは一致しない。以下の表がさらなる例を与える。

パターン	表現	一致
(?a + ?b)	(1 + 2)	{"a": 1,"b": 2}
(?a + ?a)	(1 + 2)	{}
(?a + days ?b)	(1 + days 2)	{"a": 1,"b": 2}
(＼a -〉 a + ?b)	(＼x -〉 x + 2)	{"b": 2}
(＼a -〉 a + ?a)	(＼x -〉 x + 2)	{"a": 2}

【０１０７】

DAML(商標)の実施形態は、パターン変数の引数を用いて関数を呼び出すことによってパターンを作成することができる。たとえば、iouSellSettle ?obligor ?owner ?amountは、iouSellSettleモデルテンプレートを使用して作成されたモデルと一致するパターンである。

【０１０８】

上の例が示すように、traceMatchingContractsは、パターン変数を伴う型Contractの表現を受け入れる。得られる契約パターンが次いで、すべてのアクティブモデルと

照合され、その結果が解釈の一部として報告される。DAML(商標)の実施形態は、その結果をScenarioの結果においてまだ公表せず、それは、DAML(商標)の実施形態がまだ結果のリストを表現できないからである。

【０１０９】

2つのパターン変数?ContractId、?ChoiceTimeは、予備のパターン変数名であり、モデルパターンにおいては使用することができない。それらはそれぞれ、モデルのcontractIdBinderおよび選択肢におけるchoiceTimeBinderと一致する。モデルがこれらの2つの変数のいずれかによってパラメータ化される場合、それらの値は、普通のパターン変数と同じ方法でインタプリタによって報告される。

【０１１０】

台帳の中のすべてのモデルをパターンマッチングすることはできるか。ある具体的な台帳を仮定して、パターンマッチングクエリを使用してその台帳のアクティブモデルのすべてを取り出すことが可能であるかどうかを尋ねることがあるかもしれない。これは確かに可能であり、それは、クエリとして変数を何ら伴うことなく、モデルのASTを使用して各モデル自体についてクエリすることが常に可能であるからである。よって、この疑問を次のように再編成することができる。ある会社が取引に関心のあるすべてのモデルのためのテンプレートを定義するある具体的なDAML(商標)のファイルを仮定すると、この会社に影響を与えるすべてのモデルを、このDAML(商標)のファイルからのトップレベル定義に基づくクエリを使用してパターンマッチングすることができるか?これは一般には当てはまらず、それは、対応するトップレベル定義がないアクティブモデルを生み出す、ネストされたawaitステートメントがあり得るからである。しかしながら、すべてのawaitが次のように構築されたトップレベル定義だけにおいて発生することをDAML(商標)の実施形態が要求する場合、得られるモデルのすべてをマッチングすることができる。

```
contractTemplateDef = \(paraml :: tyl) ... (paramN :: tyN) ->
 let abbrevl = bodyl;
   ...
   abbrevM = bodyM;
 in await
  { ....
  }
```

【０１１１】

この制約は表現性を制限しない。したがって、代替的なDAML(商標)の実施形態は、この形へのネストされたawaitの自動的な変換を研究し得る。DAML(商標)の実施形態がこの変換を有すると、DAML(商標)の実施形態は常に、モデルの契約毎テンプレート表の表現とネイティブなASTベースの表現との間で切り替えるために、パターンマッチングを使用することができる。

【０１１２】

代替的な実施形態の提案される言語拡張。以下のセクションは、例示的な実行モデルにおいて現在実装されていない完全に規定された拡張を含む。

【０１１３】

選択必須の義務(must-choose obligation)の例が解析され実行される。多くのモデルは、ある時間の期限内にある選択を行うことを関係者に要求する。DAML(商標)の実施形態は、上の「mustPayIouWithin」の例に示されるような懲罰項を使用することによって、ここまでに説明されたモデル言語を用いてそのような義務を表現することができる。しかしながら、すべてのこれらの懲罰項を追加することはやや煩雑になり、これが、DAML(商標)の実施形態が選択必須の義務を構文上およびセマンティクス上で明確にサポートする理由である。

【０１１４】

DAML(商標)の実施形態は、選択必須の義務に対するサポートを3ステップで説明する。DAML(商標)の実施形態はまず、導入のための例を与える。次いで、DAML(商標)の実

施形態は、選択必須の義務の正式な構文を規定する。最後に、DAML(商標)の実施形態は、それらのセマンティクスを提示する。以下の契約モデルは、ある最大限の時間まで別のモデル上で"sell"の選択肢を実行する義務をモデル化する。

```
    mustPayUntil =
     \(seller :: Party)
      (buyer :: Party)
      (good :: Contract)
      (maxSellTime :: Time)
     ->
      await
       {"pay":
        seller must choose cid :: ContractId until maxSellTime
        such that
         cid ~ good
        then
        {"payment": seller exercises "sell" with buyer on cid
        }
       --DAML(商標)の実施形態は、破られた「選択必須の」義務が契約全体
--の凍結につながることを後の例において示すために、購入者が支払いを受け      -
--る権利を失うことを可能にする。
      ,"forfeit": buyer chooses then {}
      }
    ;
```

【０１１５】

これらの言語要素は、選択必須の義務であり、この義務の履行の時間期限である。以下のscenarioは、mustPayUntil契約テンプレートの使用を示す。

```
    testSuccessfulSale =
     let iouFor =\(owner :: Party) -> iouSellSettle 'Bank' owner 100;
       aliceMustPayBobUntil = mustPayUntil 'Alice' 'Bob' (iouFor 'Alice');
     in
      scenario
       [-- 'Alice'に対するIOUおよび支払義務を作成する
        pass (days 0) ~> tO
       ,'Bank' commits create (iouFor 'Alice') ~> aliceIou
       ,'Alice' commits create (aliceMustPayBobUntil (tO + days 2)) ~
> mustPay
        --Aliceは契約を選択できるので義務を負わない。
        --しかしながら、彼女が選ぶことができる唯一の種類の契約は、彼女が義務を
--負うようになるような契約である。このことは、彼女に事実上義務を負わせ    --る。
        --1日後の支払義務の履行の成功を明示する
       ,pass (days 1)
       ,'Alice' commits 'Alice' exercises "pay" with aliceIou on mustPa
y ~> {"payment": bobIou}
       ,assert (bobIou ~ iouFor 'Bob')
       ]
     ;
```

【０１１６】

任意選択の選択肢と選択必須の義務の両方をサポートする選択ステップに対する正式な構文は次の通りである。

```
    controllingPartyExpr1
```

```
[chooses ¦ must choose]
  valueBinder1_1 :: Type1_1, ..., valueBinder1_L :: Type1_L
 at choiceStepTimeBinder1
 after t0
 until t1
such that
 booleanChoiceStepCondExpr1
```

ここで、t0およびt1は選択肢の限度値を何ら参照しないTime型の表現であり、[chooses ¦ must choose]は、¦の両側の2つのキーワードのうちの1つが使用されなければならないことを意味する。任意選択の選択肢に対しては、after 0およびuntil t1の両方の制約が任意選択である。選択必須の義務に対しては、after 0の制約のみが任意選択である。

【０１１７】

DAML(商標)の実施形態は、2つの部分における拡張された選択ステップのセマンティクスを説明する。after 0およびuntil t1の時間制約が評価される。次いで、この実施形態は、「選択必須の」義務がいつ破られるか、およびそのような違反の結果が何であるかを定義する。

【０１１８】

after tMinの制約を伴うあらゆる選択肢または選択必須の義務は、時間t〉=tMinにおいてのみ選ぶことができる。until tMaxの制約を伴うあらゆる選択肢または選択必須の義務chは、t〈tMaxにおいてのみ選ぶことができる。DAML(商標)の実施形態は、tMaxをchの水平線と呼ぶ。DAML(商標)の実施形態は、after 0 until t1およびafter t1 until t2という形の2つの制約と関連付けられる間隔が分離されることが確実に保証されるように、{t ¦ tmin 〈= t 〈 tMax}という形の時間間隔として、after tMin until tMaxを解釈することを決めた。それによって、DAML(商標)の実施形態は、2つの選択肢が同時に利用可能であるコーナーケースの回避を簡単にする。DAML(商標)の実施形態は、after 0によって制約される選択肢chがafter 0の制約のない選択肢chと同じ挙動を有することを確実にするために、最低限の時間の間その間隔を閉じた状態にすることを決めた。

【０１１９】

選択必須の義務chは、時間tがchの水平線以上である場合にのみ、時間tにおいて破られる。選択必須の義務を含むモデルインスタンスは、その選択必須の義務のうちの1つが時間tにおいて破られる場合にのみ、時間tにおいて破られる。DAML(商標)の実施形態は、すべての関係者に対するすべての選択肢を無効にすることによって、破られたモデルを凍結する。

【０１２０】

破られたモデルを完全に凍結することの動機は、次の通りである。DAML(商標)の実施形態は、違反を解決するためのデフォルトの挙動がないモデルにおいて使用されるように、選択必須の義務を設計した。デフォルトの挙動があれば、DAML(商標)の実施形態は、違反を処理するための任意選択の選択肢を相手の関係者に与えることによって、デフォルトの挙動を符号化でき、そうするであろう。DAML(商標)の実施形態は、モデルcにおける選択必須の義務の違反がモデルcのすべての利害関係者に承認を求め、次いで違反により引き起こされる損害を補償するための他のモデルを作成するとともにモデルcを削除する、解決モデルを提案することによって、モデルcにおける選択必須の義務の違反が解決されることを期待する。以下の例は、そのような選択必須の義務の違反およびその解決を示す。

```
testMustChooseObligationBreachResolution =
 let iouFor = \(owner :: Party) -> iouSellSettle 'Bank' owner 100;
   aliceMustPayBobUntil = mustPayUntil 'Alice' 'Bob' (iouFor 'Alice');
 in
```

```
        scenario
         [--'Alice'に対するIOUおよび支払義務を作成する
          pass (days 0) 〜〉 tO
         ,'Bank' commits create (iouFor 'Alice') 〜〉 aliceIouId
         ,'Alice' commits create (aliceMustPayBobUntil (tO + days 2)) 〜
〉 mustPayId
          --「選択必須の」義務のうちの1つが破られるとすぐに契約が凍結されること
--を明示する
         ,pass (days 2)
          --'Alice'は支払いが遅すぎた。
         ,mustFail ('Alice' exercises "pay" with aliceIouId on mustPayId)
          --「選択必須の」義務が破られた契約は凍結される
         ,mustFail ('Bob' exercises "forfeit" on mustPayId)
          --破られた契約の解決を明示する
          --
          --'Bob'が賠償の提案を受け入れてから2日以内に'Alice'が元の支払いと100
%の          --罰金を支払うことによって'Alice'が破られたmustPayId契約を解決で
きるこ          --とに、'Alice'および'Bob'が台帳の外で合意したと仮定しよう。
         ,'Alice' commits create (
          await
           {"accept": 'Bob' chooses at t then
            {"deleted": delete mustPayId (aliceMustPayBobUntil (tO + da
ys 2))
            ,"payment1": aliceMustPayBobUntil (t + days 2)
            ,"payment2": aliceMustPayBobUntil (t + days 2)
            }
           ,"bob rejects": 'Bob' chooses then {}
           }
         ) 〜〉 proposedResolution
        ,'Bob' commits 'Bob' exercises "accept" on proposedResolution
          〜〉 { "payment1": payment1, "payment2": payment2 }
         --Aliceは最初の1回分の支払いを直ちに行う。
        ,'Alice' commits 'Alice' exercises "pay" with aliceIouId on payment1
        ]
      ;
```

【０１２１】

代替的なDAML(商標)の実施形態は、DAML(商標)の実施形態が代替的な実行環境において追加することを望み得るモデル言語の拡張を含む。それらは、相対時間において1日を表記するための'1d'のような、相対時間のための記述リテラルを含み得る。同様に、相対時間を測定する他の一般的な単位のための。法的合意のためのMustacheテンプレート:法的文書を記述するための明示的な構文は極めて冗長である。Mustacheテンプレートの形式の、軽量な代替物が提案される。たとえば、DAML(商標)の実施形態は、合意を次のように使用し得る。

```
      seller, buyer agree
       "{{seller} has mown the lawn of {{buyer}} between {{tbought}}
       and {{tbought + 1d}}, and if this was not the case,
       then {{buyer}} can sue {{seller}} according to the Swiss OR."
```

【０１２２】

Public Choicesは、選択肢のためのchoiceTimeBinderプレフィックスにおけるいくつかのpartyMakingChoiceBinderを説明する。DAML(商標)の実施形態は、明示的

な変換機能を定義する軽量な構文を用いたnewtypeの定義を可能にし、表現型にすべての関数の持ち上げられた(lifted)バージョンを導入する。これらのnewtypeは、位置の呼出しの取り決めにおける混乱の機会の数を減らすことが意図されている。

```
newtype Amount = Integer
fromAmount :: Amount -〉 Integer
toAmount :: Integer -〉 Amount
newtype Account = Text
fromAccount :: Account -〉 Text
toAccount :: Text -〉 Account
```

【０１２３】

Type + Operator Extensionsは、DAML(商標)の実施形態が有用であると目論む拡張のグループを含み、これらはすべて、追加の型のセットおよびそれらの型を使用した関数として記述され得る。DAML(商標)の実施形態は現在、次のものを目論んでいる。バンキングモデルは取引位置固有の暦の慣習に大きく依存し得るので、Banking Calendar Support。DAML(商標)の実施形態は、これらの慣習を捉えるために、よく定義されたセマンティクスを習慣の実行者に提供することを望むことがある。Timeは普通、その地域の取引の時間帯の中にある。多くのオプション/先物は、月の第3金曜日の08:30などに期限を迎える。日付が休日の前に移動するような休業日の間には、特別な取り扱いが必要である。したがって、businessDayOrBefore(t,"Europe/Zurich")のような何かの観測可能物を使用することが有用であり得る。

【０１２４】

Simple Observableが有用であり、それは、多くのモデルが、どのように測定されるか客観的な定義があるような、市場データおよび物理的な現実の他の観測可能な値にアクセスする必要があるからである。モデルが観測可能な値の一般的な定義を共有する関係者間で実行される場合、obs :: Text -〉 Time -〉 Integerという形の演算子が有用であり、特定の時間において観測可能物の値を探すには十分である。未来の時間に対しては、この探索は失敗することに留意されたい。モデルの関係者が観測可能な値の公表を第三者に委託することを望む場合、DAML(商標)の実施形態は、次のセクションにおいて説明されるcryptographic receiptを使用することができる。

【０１２５】

Cryptographic Receiptは、たとえば認定された市場データのための、台帳外の暗号ステートメントを用いたモデルの進展の拘束をサポートするために使用され得る。Simple Key-based Receiptは、選択肢の条件において署名を直接使用する。Policy-based Receiptは、Proof-Carrying Authorizationに基づき、選択肢の条件において提供される受領証を確認するために精巧なポリシーが使用されることを可能にする。

【０１２６】

図2を見ると、回転可能な2x2x2の多面のパズルキューブが、参照番号200によって全体的に示されている。説明を簡単にするために、このキューブは、モデルのあり得る状態を表現するためにスマートモデルといくらか類似している例として与えられる。キューブの状態は、3つの見える面の各々に現れる4つのブロックの順序付けられた色として定義される。初期状態では、キューブの6つの面の各々のそれぞれの面のすべての4つのセクションが、赤(R)、白(W)、青(B)、黄(Y)、緑(G)、および黒(B)などの6つの色のうちのそれぞれ1つである。最初の取引は、上を90度時計回りに回転すること通じてキューブを遷移させる。2番目の取引は、右を90度時計回りに回転することを通じてキューブを遷移させる。

【０１２７】

ここで図3を見ると、図2のキューブについて記録されるべき状態の表現の例が、参照番号300によって全体的に示されている。初期状態は{B,B,B,B,R,R,R,R,Y,Y,Y,Y,...}であり、上の90度の時計回りの回転の後の状態は{R,R,B,B,G,G,R,R,Y,Y,Y,Y,...}であり、右の90度の時計回りの回転の後の状態は{R,W,B,W,R,G,R,G,Y,B,Y,R,...}である。

【０１２８】
図4に示されるように、時系列の順序で付記専用台帳に記録されるべきキューブの状態が、参照番号400によって全体的に示されている。ここで、{B,B,B,B,R,R,R,R,Y,Y,Y,Y,...}という初期状態が最初に付記され、{R,R,B,B,G,G,R,R,Y,Y,Y,Y,...}という2番目の状態が次に付記され、{R,W,B,W,R,G,R,G,Y,B,Y,R,...}という3番目の状態が最後に付記される。
【０１２９】
図5を見ると、付記専用台帳のための変更セマンティクス、セキュリティおよび認証、対策が、参照番号500によって全体的に示されている。そのような対策は、スマートモデルの以前に記録された状態を置き換えられるものとしてマークすること、スマートモデルの各々の新しい状態と結び付けられた暗号ハッシュおよび/または署名を作成すること、ならびに、台帳にコミットされる各台帳が、以前の台帳エントリへの参照の証明可能な起源から構築され、以前に定義された検証規則への検証可能なコミットメントに従い、隠れた証明可能なプロパティを後で共有することを可能にし、改竄帽子がされていることを確実にするために、マークル木またはブロックチェーンを使用することを含み得る。
【０１３０】
ここで図6を見ると、台帳の複製されたコピーが、参照番号600によって全体的に示されている。複製されたコピーは、局所的な障害に対して台帳を堅牢にするために使用され得る。
【０１３１】
図7に示されるように、台帳600の複製されたコピーの間での承認およびコンセンサスの分散を伴う台帳が、参照番号700によって全体的に示されている。コンセンサスは、分散型台帳の単一のバージョンの保守を容易にする。
【０１３２】
図8を見ると、最新の承認されたエントリが付記されている台帳700が、参照番号800によって全体的に示されている。示されるように、台帳800は、キューブの前面に対する明らかな気付かれていないエラーを含む。これは受け入れられる状態ではないが、台帳エントリのセマンティクス上の正しさを検証する追加のアルゴリズムなしでは除去されない。そのようなエラーまたは矛盾状態が台帳に含まれる状況は、DAML(商標)台帳ストレージの現在の例示的な実施形態ではない。
【０１３３】
ここで図9を見ると、エラーのある台帳が参照番号900によって全体的に示されている。ここで、明らかに気付かれてコミットされた同じエラー状態が、すべてのノード上での分散した合意になっている。これは受け入れられる状態ではないが、台帳エントリのセマンティクス上の正しさを検証する追加のアルゴリズムなしでは排除されない。そのようなエラーまたは矛盾状態が台帳に含まれる状況は、DAML(商標)台帳ストレージの現在の例示的な実施形態ではない。
【０１３４】
図10に示されるように、不正確な台帳エントリのエントリを避けるための台帳エントリのセマンティクス上の正しさの検証を伴う台帳が、参照番号1000によって全体的に示されている。ここで、より高次元の規則またはマスターモデルに対する検証が無効という結論を生んでおり、1つまたは複数の関係者が、この検証の失敗が原因で、台帳エントリが台帳に入ることを阻止する。正しい検証の場合、台帳エントリは、台帳ログ内に「検証論理」が含まれた状態でコミットされ得る。DAML(商標)のストレージおよび台帳論理の現在の例示的な実施形態では、1つまたは複数の関係者が、誤りのあるまたはセマンティクス上正しくない提案された台帳エントリの取引を拒絶する。DAML(商標)のストレージおよび台帳論理の現在の例示的な実施形態では、監査の目的で各台帳の取引の検証の完全な履歴が再検討され再確認されることを可能にするブロックチェーンの形で、関係者がハッシュ木の基本論理を保持する。
【０１３５】

図11を見ると、検証アルゴリズムがその中で定義され、台帳エントリの検証の能力を強化および拡張するような台帳が、参照番号1100によって全体的に示されている。ここで、分散型台帳の各インスタンスに適用される検証アルゴリズムは、以前の台帳エントリ内で定義されるアルゴリズムに依存することがあり、結果として得られるアクションおよび検証が適用される。これは検証をより強力かつ拡張可能にする。DAML(商標)のストレージの現在の例示的な実施形態は、DAML(商標)の正式なプロパティを使用してDAML(商標)の関数のより後の実行を検証し、DAML(商標)の実行論理を検証し新しいDAMLベースの台帳エントリの包含を検証するための基礎として、以前に記憶されたDAML(商標)のデジタル資産ベースの台帳エントリを使用する。

【０１３６】

ここで図12を見ると、DAML(商標)のコードのインスタンスが参照番号1200によって全体的に示されている。このDAML(商標)のコードは、6つの面の各々の4つのセクションの各々(全部で24個)に対する色という型の変数を定義し(ある例示的な関係者が定義される型を定義する)、これは各セクションに対して6つの色のうちの1つを示す。このコードはさらに、DAML(商標)の「await」関数を含み、これはたとえば、上側または右側のいずれかを回転させるという関係者の選択を待機する。ここで、各選択は古い「キューブ」を消費しながら新しい「キューブ」を産生し、cube2の定義はその検証規則を含む。

【０１３７】

図13に示されるように、DAML(商標)のコードのインスタンスが参照番号1300によって全体的に示されている。このコードは、各側が単一の色である場合に、「そうすると(then)」関係者が「キューブが解決される(Cube is solved)」ことに「合意する(agree)」「ような(such that)」、party1の選択(「chooses」)を待機するDAML(商標)の「await」関数を含む。ここで、DAML(商標)は、関係者がステートメントおよびそれらの解釈について台帳の外部で合意することを可能にする。

【０１３８】

図14を見ると、DAML(商標)のコードのインスタンスが参照番号1400によって全体的に示されている。このコードは、各側が単一の色である場合に、「そうすると(then)」関係者が「キューブが解決される(Cube is solved)」ことおよび価値(prize)がparty2からparty1に移転される(「transferAsset」)ことに「合意する(agree)」「ような(such that)」、party1の選択(「chooses」)を待機するDAML(商標)の「await」関数を含む。したがって、DAML(商標)は、ネイティブのファンジブルデジタル資産(たとえば、HyperLedgerにおける資産)を関係者が操作することを可能にする。

【０１３９】

ここで図15を見ると、2つの関係者による交互の動きを許容するDAML(商標)のインスタンスが参照番号1500によって全体的に示されている。このDAML(商標)の2つの関係者による交互の動きは、1つだけの参加者が2つの示されている動きのうちの1つを選ぶことができる図12のものとは異なる。

【０１４０】

図16に示されるように、複数のステップを組み合わせるDAML(商標)コードのインスタンスが参照番号1600によって全体的に示されている。これは、検証が受け入れられて更新の発生が許可されるために満たされなければならない「such that」条件を伴う、複数の動きの例である。

【０１４１】

図17を見ると、2つの関係者の間での資産の交換のための例示的なDAML(商標)のコードが参照番号1700によって全体的に示されている。これは、複数の関係者の間でのキューブの交換への再帰を介して拡張され得る。

【０１４２】

したがって、DAML(商標)台帳エントリは、合意、関係者が1つまたは複数の選択を行うのを待機する「await」(アクティブモデル)、および/またはネイティブのファンジブル資産に対する操作であり得る。DAML(商標)の合意は、関係者を「現実世界の」データ

または「現実世界の」活動および/もしくは約束と結び付ける。DAML(商標)のアクティブモデルおよびデジタル資産に対する操作は、それらがデータ中心の合意を「含み」、アルゴリズム的であるアクティブモデルの実行と資産の操作とを可能にするという点で、データとアルゴリズムの両方である。アクティブモデルの実行のための検証規則は各々のアクティブモデル「内で」定義され、ここでそれらの検証規則は、ブロックチェーンのスクリプトのようであるが事実上より洗練された言語によるものであり、新しいアクティブモデルが新しい検証規則を作成することができる。

【０１４３】

ここで図18を見ると、例示的なDAML(商標)コードのエントリ型が参照番号1800によって全体的に示されている。アクティブモデルは、実行されると非アクティブモデルになる。すなわち、ある選択肢の選択において1つの選択肢しか選ぶことができず、選ばれると、その選択肢の選択はもはや利用可能ではない。

【０１４４】

図19に示されるように、例示的なDAML(商標)コードの選択肢が参照番号1900によって全体的に示されている。合成によって、各モデルのステップを望み通りに洗練されたものにすることができる。ここで、図16の「Top 180」のための複数の動きの待機に加えて、「Swap cubes」のための別の待機が提示される。更新セクションは、アクティブな交換されるキューブを作成して、以前のキューブを非アクティブにすることでそれらを削除することによって、2つのキューブを交換する。

【０１４５】

図20を見ると、順序付けられた台帳エントリを伴う例示的なDAMLベースの台帳が参照番号2000によって全体的に示されている。台帳は、ここでX、Y、Zとして示されている台帳エントリの順序付けられたセットである。DAML(商標)台帳のストレージの現在の例示的な実施形態は、厳密な「1次元」の順序が維持される付記専用の連続する順序付けである。DAML(商標)のストレージの代替的な実施形態は、台帳エントリのより厳密ではないがそれでも論理的に正しい並べ替え(たとえば、有向非巡回グラフの使用を通じた)を使用することによって、より良いスケーリングを達成するために使用され得る。

【０１４６】

ここで図21を見ると、順序付けられた台帳エントリを伴う別の例示的なDAMLベースの台帳が参照番号2100によって全体的に示されている。図20の厳密な1次元の順序付けは連続的であるが、ローカルの順序付けと対比したグローバルな順序付けは、他の連続的な順序付けの可能性を許容する。ここで、トポロジカルソートまたは有向非巡回グラフ(DAG)技法が、より大きなシナリオおよびより複雑な可能性に対してスケーリングするために利用され得る。

【０１４７】

図22を見ると、順序付けられタイムスタンプが付された台帳エントリを伴う例示的なDAMLベースの台帳が参照番号2200によって全体的に示されている。この台帳は、ここではX、Y、Zとして示されている台帳エントリの順序付けられタイムスタンプが付されたセットを格納する。DAML(商標)台帳のストレージの現在の例示的な実施形態は、時間情報が各々の単調に増加するタイムスタンプが付された台帳エントリとともに維持される、付記専用台帳である。

【０１４８】

図23を見ると、複数の関係者にわたる2つの例示的なDAML(商標)のストレージおよび台帳論理の展開が参照番号2300によって全体的に示されている。ここで、一方(右)は図22のような集中型論理を有するが、他方(左)は分散型論理を有する。DAML(商標)のストレージおよび台帳論理の集中型の実施形態は、重要な商取引関係者(たとえば、取引所)を中心とし、関係者はそれでも自身固有のバージョンのDAML(商標)のストレージおよび台帳論理を維持することができる。DAML(商標)のストレージおよび台帳論理の展開の分散型の実施形態は、重要な商取引関係者を中心としない商取引プロセスをサポートするために分散型である。DAML(商標)のストレージのこの実施形態および同様の実施形態

は、広義単調増加するタイムスタンプ(たとえば、関係しない台帳エントリが時間的な依存性を有しない)を使用することによって、より大きなスケーリングを達成するために使用され得る。

【０１４９】

ここで図24を見ると、データ構造として記憶される台帳エントリおよび機密性を伴う例示的なDAML(商標)の取引が参照番号2400によって全体的に示されている。集中型(左)と分散型(右)の両方の台帳の実施形態が示されている。ここで、関係者CおよびDはある台帳エントリを見ることができないが、関係者AおよびBは見ることができる。ある関係者に見える台帳は、機密性をサポートすることなどのために、たとえば台帳取引が認可された関係者だけに見える台帳エントリを示すように、既知の暗号技法を使用して実質的に隠され得る。ここで、台帳取引は、「グループ」として追加される複数の台帳エントリを含むことがあり、取引論理はグループに対する「オールオアナッシング」の実行論理を確実にする。取引は取引の参加者の部分集合に対して非公開の台帳エントリを含むことがあるので、取引に関与するすべての関係者がその取引のすべての台帳エントリを見る必要はなく、または見ることを認められないので、機密性および部分的な可視性がもたらされる。取引が入力またはコミットされることを複数の関係者が認可する必要がある、取引およびエントリの挿入/付記の認可がサポートされる。

【０１５０】

ここで図25を見ると、台帳エントリおよび機密性を伴う例示的なDAML(商標)の取引が参照番号2500によって全体的に示されている。ここで、関係者Bはある台帳エントリを見ることができないが、関係者Aは見ることができる。ある関係者に対して可視の台帳は、機密性をサポートすることなどのために、台帳取引が認可された関係者だけに見える台帳エントリを示すように実質的に隠され得る。ここで、台帳取引は、「グループ」として追加される複数の台帳エントリを含むことがあり、取引論理はグループに対する「オールオアナッシング」の実行論理を確実にする。取引は取引の参加者の部分集合に対して非公開の台帳エントリを含むことがあるので、取引に関与するすべての関係者がその取引のすべての台帳エントリを見る必要はなく、または見ることを認められないので、機密性および部分的な可視性がもたらされる。取引が入力またはコミットされることを複数の関係者が認可する必要がある、取引およびエントリの挿入/付記の認可がサポートされる。

【０１５１】

図26に示されるように、新しい台帳エントリの取引のコミットメントの複数の関係者による認可を含む例示的なDAMLベースの台帳が、参照番号2600によって全体的に示されている。ここで、認可は影響を受ける関係者を検証することによってもたらされる。1つまたは複数の検証側の関係者からの認可は、台帳にコミットされたすべての新しい台帳エントリが適切な以前に合意された台帳論理を満たすことを保証し得る。影響を受ける関係者からの認可は、台帳エントリ内で定義された未来の商取引の変形およびシナリオに合意すると約束することであり得る。すなわち、1つまたは複数の関係者は、提案されたDAMLベースの台帳取引の、即刻のおよび場合によっては未来の台帳の実行の影響に同意する。同様に、1つまたは複数の関係者は、提案されたDAMLベースの台帳取引の関係者間の関係に対する即刻のおよび場合によっては未来の影響に同意する。DAML(商標)台帳の現在の例示的な実施形態では、1つまたは複数の関係者がDAML(商標)台帳の実行論理に同意し、以前に合意された外部の法的合意の文脈の範囲内での提案されるDAML(商標)台帳取引の解釈に従った2つ以上の関係者の間での法的な権利および義務に対する変更に、1つまたは複数の関係者が同意する。DAML(商標)台帳の代替的な実施形態は、DAML(商標)台帳取引が、以前の外部の合意の枠組みの中で機能する必要なく2つ以上の関係者の間の法的拘束力のある関係を定義することを可能にするために、法的権威により証明された認可を有し得る。DAML(商標)台帳の代替的な実施形態は、DAML(商標)台帳取引が2つ以上の法的管轄区域にまたがる2つ以上の関係者の間での法的拘束力のある関係を定義することを可能にするために、複数の法的権威により証明される認可を有し得る。DAML(商標)台帳の現在の例示的な実施形態では、1つまたは複数の関係者がDAML(商標)

台帳の実行論理の正しさを検証する。

【０１５２】

図27に示されるように、ハッシュと対比されたエントリおよび詳細を含む例示的なDAMLベースの2層の台帳が参照番号2700によって全体的に示されている。ここで、すべての関係者が取引およびハッシュ値を匿名化しているが、関係者Bは関係者Aが有する一部の詳細を有しない。ハッシュ鍵は台帳エントリと結び付けられる。ハッシュは各台帳エントリと関連付けられるが、ハッシュは台帳エントリの詳細を明らかにしない。ハッシュは、台帳エントリの詳細を明らかにすることなくそれらの証明を提供するのを助ける。2層の台帳は、ハッシュデータの台帳またはログ、および台帳エントリの台帳を含む。両方の台帳の層が、取引構造および/または任意選択のブロック構造を含む。

【０１５３】

図28に示されるように、例示的なDAMLベースのハッシュ中心公開台帳層または公開ログが参照番号2800によって全体的に示されている。ここで、公開台帳は証明可能な認可された参照を提供する。この公開台帳は、台帳エントリデータを隠す「ハッシュ」の台帳である。各台帳エントリのハッシュは、その台帳エントリに対する匿名化された参照である。これは、取引のコミットメントのステータスを提供し、関係者は自身の非公開の台帳に基づいて取引を認可するが、取引のコミットは公開台帳の中にある。非公開台帳のコミットは公開台帳のコミットに依存する。この特定の実施形態はハッシュだけを実装するが、有限のライフサイクルもエントリ間の依存性も有しない。すなわち、公開台帳は、エントリが非公開台帳に記憶されている詳細な台帳エントリと関連付けられるハッシュ値であるようなログであり、公開台帳は詳細な非公開のエントリに対する匿名化された参照である。台帳取引のコミットメントは、公開台帳内で1つまたは複数の関係者によって認可され、この公開台帳のコミットメントのステータスは、非公開台帳の取引のコミットメントのステータスを決定する。DAML(商標)台帳の現在の例示的な実施形態は、そのエントリ(ハッシュ値)のライフサイクルもエントリ間の依存性も含まないので、ログとして見なすことができる。DAML(商標)台帳の代替的な実施形態は、いくつかの台帳の詳細の同型の暗号化などの追加の論理を含むことがあり、さらなる台帳論理を実装することがある。

【０１５４】

図29に示されるように、例示的なDAMLベースの非公開の、かつ共有可能な非公開台帳層が、参照番号2900によって全体的に示されている。ここで、非公開台帳は、認可された非公開の、かつ共有可能な非公開台帳を含む。台帳エントリデータは、「知る必要がある者にしか知らせないという原則」で共有され得る。関係者は、自身が「参加者」ではない台帳エントリを受け取らず、見ず、記憶もしないので、プライバシーが保たれる。非公開台帳と公開台帳は、台帳エントリのハッシュを通じてリンクされる。関係者(たとえば、検証側の関係者)によって構築および共有され、安全に分散され、暗号化されるマークル木上の「マークル木ベースの証明」を含めて、認可および取引のコミットメントが両方の台帳に適用される。ここで、知る必要がある者にしか知らせないという原則が、非公開台帳を部分的に見ることができる関係者によって使用され得る。

【０１５５】

非公開台帳は、要求された非公開台帳のプロパティに対する要求側の関係者の関係に基づいて、知る必要がある者にしか知らせないという原則で、公開台帳に対して関連付け可能なプロパティの証明書または「正しさの証明」を他の関係者に提供する。非公開のデータプロパティを通じた公開台帳のエントリの証明書および証明スキームの現在の例示的な実施形態は、マークル署名スキームに基づく。証明書および証明スキームの代替的な実施形態が可能であり、たとえば、PKIおよびproof carryingスキームに基づく。

【０１５６】

図30に示されるように、関連する台帳エントリの特定および台帳エントリの活発さの追跡を伴う、例示的な2層のDAML(商標)台帳が、参照番号3000によって全体的に示されている。これらの2つの台帳エントリのプロパティの現在の例示的な実施形態は、非公開台帳層の中に存在するハッシュ構造の上に構築され、ハッシュ木/グラフが、固有の関

連する暗号ハッシュ値を台帳エントリと関連付けながら、DAML(商標)(および場合によっては非DAML)ベースの非公開台帳のエントリ間での非巡回の依存性を捉えるために使用され、一方、マークル木構造を付加的に更新することが、各台帳エントリの活発さ(アクティブな状態対非アクティブな状態)を追跡するために使用される。台帳エントリの特定および台帳エントリの活発さの追跡の代替的な実施形態は、ハッシュ値割当て論理内で、および活発さを追跡する構造内で、物理区分モデルのプロパティを捉えることを介して、台帳エントリデータの不均一な性質に対処することによって、スケーラビリティを狙うことができる。

【０１５７】

図31に示されるように、ブロック指向の論理を伴い、関連する台帳エントリの特定およびブロック中心の活発さの追跡を伴う、例示的な2層のDAML(商標)台帳が、参照番号3100によって全体的に示されている。台帳エントリの活発さを追跡するための現在の例示的な実施形態は、複数の取引のブロックとして台帳取引を再グループ化し、取引の各々の新しいブロックを用いて非公開台帳のマークル木を付加的に更新することである。ブロック指向の台帳のエントリの活発さを追跡することの代替的な実施形態は、ハッシュ値割当て論理内で、およびブロック指向の活発さを追跡する構造内で、物理区分モデルのプロパティを捉えることを介して、台帳エントリデータの不均一な性質に対処することによって、スケーラビリティを狙うことができる。

【０１５８】

図32に示されるように、非公開台帳のエントリのプロパティの証明書または証明可能なプロパティの、例示的な関係者間での共有が、参照番号3200によって全体的に示されている。非公開台帳のエントリのプロパティの証明書または証明可能なプロパティを共有するための現在の例示的な実施形態は、台帳エントリハッシュ値と結び付けられた部分的なより深い非公開台帳データによって任意選択でサポートされる、特定の台帳エントリのマークル署名スキームベースの証明書を、知る必要がある者だけに知らせる原則で提供することである。ここで、台帳の活発さの認可は「マークル署名」に基づく。

【０１５９】

図33を見ると、分散型台帳のホストされたコピーおよびホストされないコピーが参照番号3300によって全体的に示されている。ここで、公開のバージョンは左の列にあり、非公開のバージョンは右の列にある。公開のバージョンと非公開のバージョンの両方のホストされたコピーは、関係者A1、A2などのための一番上の行の中にあり、ホストされないコピーは、それぞれ、関係者Bおよび関係者Cのための下2行の中にある。すなわち、関係者の台帳はホストされることがあり、またはそれぞれの関係者によって管理されることがある。

【０１６０】

ここで図34を見ると、Agree、Await、およびDeleteコマンドのための例示的なDAML(商標)台帳エントリの型が、参照番号3400によって全体的に示されている。DAML(商標)台帳エントリには、3つの主な型がある。ここで、Agreeは一般に、現実世界の変更または指定された関係者の間での現実世界の約束に1つまたは複数の関係者が合意することを意味する。Awaitは一般に、台帳に対する「あらかじめ定められた」変更の、より後のまたは未来の許可もしくは約束を、実行が待機することを意味する。Deleteは一般に、以前に入力されたアクティブモデルが非アクティブ化される(inactivated)(非アクティブ化される(deactivate)/廃止される/無効にされる)ことを意味する。台帳エントリはまた、根拠となる台帳によって定義されるネイティブの暗号資産を含み得る。await型のDAML(商標)台帳エントリによって参照と作成の両方が行われ得る、別々の台帳エントリがある(図14参照)。動作において、Awaitは申し出として機能することがあり、一方でChoosesおよび/またはAgreeは受け入れとして機能することがある。

【０１６１】

図35に示されるように、DAML(商標)のAgreeコマンドの例示的な使用が参照番号3500によって全体的に示されている。ここで、構文およびセマンティクスは完全に、「台

帳外で」合意する関係者次第であるが、合意のすべての関係者がそれを認可する必要がある。より詳細には、Agreeは1つまたは関係者が現実世界の変更に合意することを意味する。構文などは完全に、「台帳外で」合意する関係者次第である。合意のすべての関係者がそれを認可する必要がある。いくつかの状況では、そのような合意またはアクティブモデルは、その認可に法的な地位があり他の法定の要件が満たされる場合、法的強制力のある契約を具現化することができる。たとえば、そのような合意に法的な地位を与えるために、親契約が使用され得る。そうではない場合、親契約は他の特徴を合意に授け得る。合意またはアクティブモデルの履歴を実際に削除することはできないが、それでもある合意が以前の合意に取って代わり、それにより以前の合意を非アクティブにすることができる。したがって、関数Deleteはモデルを非アクティブにするが、その履歴を削除する必要はない。「台帳論理」は、現実世界の影響の範囲を自動的には知らない。

【０１６２】

図36を見ると、台帳エントリは、参照番号3600によって全体的に示される「インスタンス化された」エントリである。ここで、左の列の中のコードはキューブのためのインスタンス化されていないモデルテンプレートを表し、中の列はテンプレートへとインスタンス化されるべきデータ値を表し、右の列はデータ値を含むインスタンス化されたアクティブモデルを表す。

【０１６３】

ここで図37を見ると、関係者がある事象の通知をその事象のためのフォーマットで行い受け取ることに合意する、エクイティモデル内での外部への通知に対する合意の例示的なDAML(商標)コードが、参照番号3700によって全体的に示されている。代替的な実施形態では、合意は多くの他のDAML(商標)の適用例において使用され得る。

【０１６４】

図38に示されるように、キューブモデル内でのAwaitコマンドを使用した例示的なDAML(商標)コードが参照番号3800によって全体的に示されている。ここで、Awaitは台帳に対するより後の/未来の「あらかじめ定められあらかじめ認可された」変更の許可または約束を待機する。これらは、awaitエントリによってもたらされる論理「choices(選択肢)」である。各choiceは、新しい「キューブ」またはアクティブモデルを産生しながら古いものを消費し、その古いものを非アクティブモデルにする。この例では、cube2モデルの定義はその検証規則セットでもある。

【０１６５】

図39を見ると、deleteコマンドを伴うDAML(商標)台帳が参照番号3900によって全体的に示されている。deleteコマンドは、非アクティブ化/廃止/無効化をあらゆる以前のアクティブエントリに適用することなどによって、アクティブモデルを非アクティブモデルに変換するために使用される。ここで、より前の(上の)deleteはawaitを参照するが、より後の(下の)deleteはagreeを参照する。

【０１６６】

取引のバンドルまたはブロックが、以下の3つの部分を含む単一のステップとして、オールオアナッシングで実行される。

1. バンドル内の取引が、以下のものを格納するように作成され、定義され、または導出される:

a. 以下のうちの1つまたは複数のセット:

i. 以前の台帳エントリの非アクティブ化;

ii. 同じ取引内で作成され非アクティブ化される過渡エントリ;および/または

iii. 新しい台帳エントリの作成

b. 各エントリに対して

i. どの関係者がエントリを認可する必要があるか(影響を受ける関係者);

ii. どの関係者がこのエントリを見るか(どの関係者に対してエントリが見えるか);および/または

iii. 各関係者の検証確認の健全性をサポートするために必要とされるようなb(i

)およびb(ii)の推論の正当化/証明
2．内容に対して影響を受ける関係者からの認可を得る
3．認可を用いて取引を台帳にコミットする

【０１６７】

第1の部分において、取引および/または台帳エントリは、単一の関係者によって提案および/もしくは作成され、または、複数の関係者によって共同で提案および/もしくは作成され得る。取引は、取引エントリのデータ中心セットとして、または、任意選択の根拠となるソルトコードを伴う取引エントリを生成するDAML(商標)スクリプトとして通信され得る。各取引エントリは、過去とは独立であり、かつ以前の台帳エントリとは無関係であることがあり、または、1つまたは複数の以前の台帳エントリに依存することがある。提案側の関係者は、提案される台帳取引の検証の基礎を、それが影響を受ける関係者に対して以前は見えるようになっていなかった場合に提供する。たとえば、過去に依存しないエントリは正当化のために以前のエントリに対する参照を必要としない。

【０１６８】

ここで図40および図41を見ると、役割中心のフローを伴う台帳アルゴリズムが参照番号4000および4100によって全体的に示されている。台帳アルゴリズムは分散型であり、順番に、または任意選択で並列に、以下の動作ステップを反復する。

【０１６９】

ステップ1および/またはノード1において、台帳取引が開始される。台帳取引の提案が作成され、または任意選択で、既存の台帳エントリを解釈することから導出される。ある関係者が、予想される取引を指定する「台帳エントリ更新」を作成する。任意選択で、2つ以上の関係者が、台帳エントリ更新を作成することについて一緒に合意し得る。「Core Transaction」とも呼ばれ得る、台帳エントリ更新によって引き起こされる台帳への影響が、関係者のDAML(商標)エンジンシステムによって計算される。

【０１７０】

ステップ2および/またはノード2において、提案される取引は、出力、入力、認可、および証明/正当化を含み得る。出力は、1つまたは複数の新しい台帳エントリのセットを含み得る。台帳エントリの型の数は任意であり得るが、主なものは台帳エントリの破壊(DAML(商標)のdelete)、choice論理を伴うDAML(商標)のagreementまたはDAML(商標)のawaitなどの新しい台帳エントリの作成である。入力は、既存のアクティブな台帳エントリに対する0個以上の参照のセットを含み得る。各取引台帳エントリに対して、認可情報は、どの関係者がこの取引内でエントリを認可する必要があるか(影響を受ける、または検証側の関係者)、どの関係者がこのエントリを見るか(どの関係者に対してエントリが見えるか)、任意選択でどの関係者が導出されたアクションを認可するために必要であるか、および取引を正当化するために必要な台帳エントリと結び付けられた根拠となるマークル署名を含み得る。

【０１７１】

ステップ3および/またはノード3において、台帳取引が認可される。ここで、認可に対する要求が作成され、関係する関係者に送られる。要求は、作成された後、その認可が必要な関係者だけに送られ、各要求は、関係者が見えないものを除く、取引台帳エントリのコピーを格納する。

【０１７２】

ステップ4および/またはノード4において、台帳取引が調整される。認可関係者からのデジタル署名が提供される。これらは、台帳取引ごとに、または、台帳取引のマークル木内の、もしくは場合によっては取引のブロックのマークル木内の台帳エントリと結び付けられるマークル署名ごとに、生成される。各関係者は、認可を与える前に、取引、台帳エントリ、および見ることができるマークル木ハッシュの正しさを確認する。

【０１７３】

ステップ5および/またはノード5において、台帳取引が記憶される。取引に関与する各関係者は、自身の台帳の非公開の層に新しい台帳エントリを記憶するが、それはこれらを

非公開のストレージにまだコミットすることなく行われる。

【０１７４】

ステップ6および/またはノード6において、台帳取引がコミットされる。分散型コンセンサスアルゴリズムを伴う台帳(たとえば、HyperLedger)の共有される公開の層は、新しい取引をそのマークル木ハッシュおよび認可とともに、しかし取引エントリの詳細なしで、記憶およびコミットする。

【０１７５】

ステップ7および/またはノード7において、台帳取引は台帳ストレージの非公開の層にコミットされる。新しい取引に関与する各関係者は、以前に記憶された取引エントリの詳細を台帳の公開の層にコミットすることができる。

【０１７６】

DAML(商標)台帳の現在の例示的な実施形態は、各々の新しいDAML(商標)台帳のエントリ取引の提案を検証する専用の関係者(たとえば、取引所エンティティ)を有する。DAML(商標)台帳の現在の例示的な実施形態では、新しいDAML(商標)台帳エントリの取引が1つの関係者によって提案され、台帳および他の関係者との関係に対する新しい取引の影響について提案側の関係者および追加の専用の関係者が同意する。複数の関係者の同意は、複数の関係者の同意を組み合わせるDAML(商標)の論理を構築するような方式で、異なる関係者によるDAML(商標)台帳取引の連続するエントリによって行われる。DAML(商標)台帳の代替的な実施形態は、台帳および他の関係者との関係に対する新しいDAML(商標)の取引の影響に、任意の数の関係者が同意することを可能にし得る。

【０１７７】

図42を見ると、関係者中心の台帳アルゴリズムのフローが参照番号4200によって全体的に示されている。ここで、関数ブロック4210は、非公開共有層4212および公開層4214を含む台帳を開始し、制御権を決定ブロック4220に渡す。決定ブロック4220は、関係者のアクションを待機し、次いで制御権を関数ブロック4230および/または4240に渡す。関数ブロック4230は、非公開共有台帳4212上で台帳取引を開始し、制御権を決定ブロック4220に戻してさらなる関係者のアクションを待機する。関数ブロック4240は、公開台帳4214上で台帳取引を認可し、制御権を決定ブロック4220に戻してさらなる関係者のアクションを待機する。

【０１７８】

ここで図43を見ると、台帳中心の台帳アルゴリズムが参照番号4300によって全体的に示されている。ここで、関数ブロック4310は、非公開共有層4312および公開層4314を含む台帳を開始し、制御権を決定ブロック4320に渡す。決定ブロック4320は、台帳イベントに基づいて関数ブロック4330、4340、4350、4360、および/または4370のうちの1つまたは複数に制御権を渡し、これらの関数が各々、特定の関数を実施して、制御権を決定ブロック4320に戻し次の台帳イベントを待機する。ここで、関数ブロック4330は台帳取引を認可し、関数ブロック4340は台帳取引を調整し、関数ブロック4350は非公開共有台帳4312に台帳を記憶し、関数ブロック4360は関係者の台帳の公開層4314に台帳取引をコミットし、かつ/または、関数ブロック4370は関係者の台帳の非公開層にストレージ状態をコミットする。

【０１７９】

図44に示されるように、台帳取引を開始するための例示的な関数が参照番号4400によって全体的に示されている。入力ブロック4410において、ある関係者が、商取引意図メッセージ(BIT:business intent message)に基づいて新しいDAML(商標)の更新表現を開始し、制御権を関数ブロック4420に渡す。関数ブロック4420は、コア取引(CT)4412の形で提案されたDAML(商標)の更新の台帳に対する影響を計算するためにDAML(商標)エンジンを使用し、制御権を関数ブロック4430に渡す。関数ブロック4430は、台帳上でコア取引を開始し調整する。コア取引4412は、予想される変更を数え上げ、台帳エントリにデジタル的に署名するためにフックを提供する。基本的な提案される取引の詳細は、新しい台帳エントリ(アクティブモデルになるであろう出力)、および削除される

べき台帳エントリ(非アクティブモデルになるであろう入力)を含み得る。加えて、各取引は、どの関係者が各台帳エントリを認可しなければならないか、およびどの関係者が各台帳エントリを「見る」ことができるか(どの関係者に対して台帳エントリが見えるか)、ならびに、BITの暗号ハッシュの形の正当化、および根拠となるマークル証明を指定する。

【０１８０】

DAML(商標)エンジンはさらに、DAML(商標)モデリング活動を観察してそれと対話するための能力を提供する、DAML(商標)シミュレータを含み得る。DAML(商標)は、新しいタイプの契約規定言語(CSL:conract specification language)である。ユーザは、scenarioを実行し、DAML(商標)モデルがあるscenarioにおいてどのように使用されて進行するかを観察することによって、開発されたDAML(商標)コードを調査することができる。ユーザは、この可視性を使用して、たとえばユーザの入力がDAMLベースの台帳にどのように吸収されるかを観察することができる。

【０１８１】

DAML(商標)シミュレータは、ユーザがアクセス可能なホストへとDAML(商標)エンジンとともに展開され得る。DAML(商標)シミュレータは、ソースコードとしてDAML(商標)モデリングコードを伴うバイナリとして展開され得る。このソースコードはシミュレータにおいて可視であり得る。したがって、ユーザは、DAML内でユーザの商取引プロセスのモデリングを見ることができるが、このインストールを変更することはできない。DAML(商標)シミュレータはユーザインターフェース(UI)部分を含む。UIへのアクセスは、HTTP、HTTPSなどを介したものに限定され得る。

【０１８２】

動作において、DAMLベースの台帳のプロパティは、連続する台帳の動作の間のプロセスの連続性を確実にする。プロパティは、固定であり関係者の間であらかじめ合意されるが、付加的に管理されることも可能である。現在の例示的な実施形態では、プロパティは「コアプリミティブの台帳」を用いて実装され、次いで1つまたは複数の重要な関係者(たとえば、取引所エンティティ)によるバージョン管理と同様に展開し、台帳セマンティクスの後方互換性を可能にする。可変のセマンティクスが企図されるが、例示的な実施形態は既存の台帳エントリに対して固定されたセマンティクスを使用し、それは、これらの規則の新しいバージョンが、以前のバージョンから新しいバージョンに台帳エントリを移行する義務を関係者に課し得るからである。新しいバージョンの関数セマンティクスは、未来の台帳エントリがより古い台帳エントリの既存のセマンティクスを維持する限り、未来の台帳エントリに対して導入され得る。そのような新しいバージョンの関数は、たとえば、バージョンまたは日付によって追跡され得る。しかしながら、例示的な実施形態は、変更を受ける可能性がある外部の法的合意に依存することがあるので、改訂の日付によって追跡されるべきであることに留意されたい。

【０１８３】

関係者は、台帳エントリおよびデータとしてのその表現に合意する。台帳エントリは、台帳の媒体を通じて関係者が何に合意できるかを定義する。これらは、字句、文法、バイナリ、および他の表現で定義される。DAML(商標)の例示的な実施形態は、agree、await、delete、transfer、および他のHyperLedger(商標)互換のデジタル資産台帳プリミティブを使用することができる。以前および未来の台帳エントリの解釈のセマンティクス、および現実世界の合意。台帳エントリの解釈のセマンティクスは、台帳エントリをアルゴリズム的に解釈または評価するための決定論的な方法を定義する。台帳エントリのセマンティクスは、以前および未来の台帳エントリに影響することがある。

【０１８４】

DAMLにおいて、「delete」エントリは、DAMLにおける以前の台帳エントリを無効化または非アクティブ化するが、監査可能な履歴は残る。「await選択肢」の実行の解釈は、「選ばれたawaitエントリ」を削除するための「権利」と、台帳エントリの更新をサポートする制約を満たす0個以上の新しい台帳エントリを作成する権利とを提供する。

【０１８５】

合意において、台帳エントリのセマンティクスは未来の台帳エントリに影響し得る。DAMLにおいて、「await」台帳エントリは、所与のプロパティに一致する台帳エントリをより後の「choice」が作成することを認可する。さらなる解釈は、未来の台帳エントリデータに依存し得る。DAMLにおいて、await選択肢の選択構成は、1つまたは複数の未来の台帳エントリおいて提供されることになる0個以上のパラメータに任意選択で依存するテンプレートを含み得る。

【０１８６】

台帳エントリのセマンティクスは、物理的な世界に影響し得る。DAMLにおいて、「agree」台帳エントリは、「現実世界」の合意に複数の関係者を結び付ける。台帳エントリのセマンティクスは、台帳エントリの更新の宣言から新しい台帳エントリをアルゴリズム的に作成するための決定論的な方法を定義する。DAMLでは、更新ステートメントおよび一般化されたフォローアップはそのようなセマンティクスを有する。

【０１８７】

認可規則が、新しい台帳エントリの作成にどの関係者が合意する必要があるかを計算するアルゴリズムにおいて使用され得る。これは、1つまたは複数の既存の台帳エントリの解釈から台帳エントリが導出されることの影響を、または、台帳エントリが過去と無関係に作成されるかどうかを考慮する。これは、「機密性の顔」の連続性が、信頼される関係者を検証側の認可関係者として指定することによって、かつアクティブな台帳エントリのマークル木上でマークル署名を使用することによって保証され得ることを考慮する。

【０１８８】

検証側の関係者のセマンティクスが実施される。台帳エントリの解釈のセマンティクスは、台帳論理の正しさおよびマークル木ベースの論理の「連続性」を保証することによって、「二重取引がないこと」を保証する目的で検証側の関係者を定義することができる。中心の概念は、検証側の関係者がDAML内で定義され得るということである。例示的な実施形態では、それはまた、以前にそうなったことがない関係者が検証側の関係者になるとすれば、かつこの関係者が検証を保証するための十分な台帳の詳細を有していないとすれば、その関係者は別の関係者から追加の台帳の詳細を受け取ることが必要であり、制定された機密性の規則の範囲内で台帳に対するその関係者の一貫した新しい見方を示すことが可能であることが必要であり、さもなければその関係者は提案された取引を認可することが許可されないことを意味する。

【０１８９】

機密性のセマンティクスおよび規則は、新しい台帳エントリをどの関係者が見ることができるかを計算するアルゴリズムにおいて使用される。これは、台帳エントリの認可の必要性を考慮する。その上、ソルトおよびマークル木の実装規則は、台帳エントリまたは台帳エントリ更新の1つまたは複数のハッシュ鍵を計算するための決定論的な方法を定義する。これらの規則は、同じ台帳エントリおよびソルト値が与えられると同じハッシュ値を産生し、同じ台帳エントリ更新および同じソルト値の列を与えられると同じハッシュの列を産生する。台帳エントリごとに1つのハッシュが更新によって定義される。

【０１９０】

暗号プロパティを伴うハッシュは、それらのハッシュが、それらの由来である台帳エントリ上の情報を導出するために使用されることを許可しない。たとえば、DAML(商標)エンジンの実装形態は、有効性の確認のためだけに使用可能なハッシュによって危殆化されない。

【０１９１】

取引において、台帳エントリは「すべて成功」または「すべて失敗」の不可分性プロパティを用いて「グループ」として追加される。これは、DAML(商標)台帳取引の概念である。複数の関係者が、取引が入力(コミット)されることを認可する必要があるが、取引に関与するすべての関係者が取引のすべての台帳エントリを見る必要はない。これは、取引が、取引参加者の部分集合に対して非公開の台帳エントリを含み得るからである。関係者は取引に合意している間、それらの関係者は技術的に、その取引内の台帳エントリを認可

している。したがって、関係者のデジタル署名は個々の台帳エントリと結び付けられる。「取引」論理はさらに、「オールオアナッシング」の実行論理を確実にする。

【０１９２】

DAMLを使用すると、3つの型の台帳エントリがある。delete型は、以前のエントリを非アクティブモデルに移行することによって、以前のエントリを非アクティブ化する/廃止する/無効化するものと言える。await型は、台帳に対するより後のまたは未来の「あらかじめ定められた」変更に対する許可もしくは約束を具現化する。agree型は、複数の関係者が各々、1つまたは複数のステップ、条件、または結果に合意することを要求する。

【０１９３】

awaitエントリは、認可側の関係者が「awaitの選択肢の結果を許容するように彼ら自身を拘束する」という、「選択肢」を提供する。agreeエントリは、現実世界の変更について1つまたは複数の関係者が合意することを要求するが、その結果は非アクティブ化することができず、合意自体を削除することはできない。「台帳論理」は範囲を知らないので、そのような範囲が「削除され」得るかどうかは決定できない。

【０１９４】

デフォルトで、合意のすべての関係者が合意を認可する必要がある。合意は以前の合意に取って代わり得る。合意は典型的には「多事(eventful)」であるが、そうである必要はない。合意の構文および解釈は完全に、「台帳外で」合意する関係者次第である。例示的な実施形態の台帳は、そのような台帳外の合意を記録するが、それらを解釈しようとはしない。特定の状況のもとでは、アクティブモデルにつながるそのような合意は、所与の法的管轄区域における法的拘束力のある契約の要件を、その契約が関係者の意図でありそれぞれの認可が法的な地位を有していた場合、満たすことができる。一般に、台帳は、所与の合意に法的強制力があるかどうかを気にせず、例示的な実施形態は、一般的な合意と、法的強制力のある契約の基準を満たす契約とを区別しない。望まれる場合、本発明の概念は、親契約が特定の法的管轄区域における契約としての法的地位をDAML(商標)の合意に与えるために使用され得ることを想定する。

【０１９５】

上で論じられたすべての台帳エントリ、コード、宣言ステートメント、データ構造などは、非一時的コンピュータ可読記憶媒体に記憶され得る。本明細書で説明される機能ステップは、プロセッサ上で実行されるコンピュータコードによって遂行され得る。上で説明された様々なデータ操作は、異なる方式でコンピュータプロセッサによって処理される変換されたデータ構造を作成するために、記憶されているデータ構造上で達成され得る。await関数などの、実施形態の様々な関数は、コンピューティングシステムが取引を遂行するために新しい方式で動作することを可能にし、従来技術では見出されない利点を提供する。様々なフローチャートのステップは、コンピュータプロセッサ上で実行されるソフトウェアモジュールによって遂行され得る。図面に示される円柱は、記録を記憶するデータベースなどのデータ構造を表現することができ、これらは、コンピューティングシステムがデータを操作してデータを変換することを可能にするために、説明された方式で操作される。

【０１９６】

本明細書で説明されたコンピュータで実施される方法の1つまたは複数を実行するデジタルシステム500の例が図45に示されている。システム5000は、デジタル資産と、複数の関係者の関連する権利を含むデジタル資産の展開のモデリングおよび/または記録を容易にし得る。ある例として、複数の関係者は第1のユーザ5001および第2のユーザ5003を含み得る。第1のユーザ5001は第1のノード5002と関連付けられ、第2のユーザ5003は第2のノード5004と関連付けられる。第1のノード5002および第2のノード5004は第3のノード5007と通信していることがある。これは、第1のノードおよび第2のノード(およびそれぞれのユーザによって操作される)から第3のノード5007へのメッセージ、通知、指示、選択、認可などを可能にする。

【０１９７】
第1のノード5002、第2のノード5004、および第3のノード5007は、コンピュータ、タブレットコンピュータ、モバイル通信デバイス、コンピュータサーバ、コンピュータ端末などの電子デバイスであり得る。そのような電子デバイスは、処理デバイス、データストア、およびユーザインターフェースを含み得る。ユーザインターフェースの例は、キーボード、マウス、モニタ、タッチスクリーンディスプレイなどを含む。
【０１９８】
処理デバイス5008を有する第3のノード5007は、少なくとも1つのストレージデバイス5009と通信している。いくつかの例では、ストレージデバイス5009は、付記専用台帳のための、ハードディスク、磁気ディスク、ソリッドステートストレージなどのデータストア5011である。いくつかの例では、付記専用台帳のためのストレージデバイスは、ピアツーピア分散型台帳5013の形態である。ピアツーピア分散型台帳5013は、データを受信して記録するために、1つまたは複数の処理デバイス5015と関連付けられ得る。いくつかの例では、ピアツーピア分散型台帳5013はブロックチェーンを含む。
【０１９９】
システムはまた、関与する関係者のうちの1つであり得る別のユーザ5017を含み得る。別のユーザ5017は、システム5000と対話するために、第2のユーザ5003(および対応する第2のノード5004)などの代表者を使用し得る。
【０２００】
第1のノード5002、第2のノード5004、第3のノード5007、およびデータストレージ5009は、ローカルエリアネットワーク、ワイドエリアネットワークのうちの1つまたは複数を含み得る1つまたは複数のネットワークを通じて互いに通信していることがあることを理解されたい。いくつかの例では、これはセキュアなローカルエリアネットワークを含み得る。いくつかの例では、通信はインターネットを含む通信ネットワークを含み得る。
【０２０１】
図46は、デジタル資産およびその展開をモデリングおよび/または記録するためにデータ構造を操作する方法100の例を示す。この方法の少なくとも一部は、第3のノード5007によって実行され得る。これは、複数の関係者のうちの少なくとも1つの権利に関するデジタル資産の譲渡のために、await関数インスタンスの中で定義される少なくとも1つの選択肢のうちの1つを少なくとも一度使用して、一度だけ実行されるawait関数インスタンスを提供すること(1200)を含み、前記await関数インスタンスは、少なくとも1つの選択肢と関連付けられるconfigured関数インスタンスを履行するという影響を受ける関係者の同意のもとで組み込まれる。
【０２０２】
いくつかの例では、このステップ1200は、関係者5001、5017と関連付けられるノード5002、5003から送信される(5100)同意の指示を受け取ると開始され得る。
【０２０３】
方法100は、実行するために複数の関係者のうちの少なくとも1つの同意を必要とするagree関数インスタンスを提供すること(1300)を含み得る。いくつかの例では、このステップ1300は、関係者5001、5017と関連付けられるノード5002、5003から送信される(5100)agree関数と関連付けられる同意の指示を受け取ると開始され得る。
【０２０４】
方法100はまた、実行された関数インスタンスの結果を記憶するための付記専用台帳に対するアクセスを提供すること(4000)を含み得る。そして、少なくとも1つのストレージデバイス5009は付記専用台帳に結果を記憶し得る(5200)。
【０２０５】
いくつかの例では、システム5000は、モデリングされたデジタル資産および複数の関係者の権利に関するデジタル資産の展開を解釈する方法を実行し得る。これは、上で説明されたawait関数1200のawait関数インスタンスを実行するステップと、上で説明され

たagree関数1300のagree関数インスタンスを実行するステップとを含み得る。これらの実行された関数インスタンスの結果が次いで、付記専用台帳に記憶され得る。

【０２０６】

システム5000によって実行される方法6000の別の例が、ここで図47を参照して説明される。この方法6000の少なくとも一部は第3のノード5007によって実行され得る。これは、複数の関係者の関連する権利を含む、デジタル資産およびその展開をモデリングおよび/または記録するためにデータ構造を操作する、コンピュータで実施される方法を示す。方法6000は、デジタル資産の譲渡のためにawait関数の中で定義される少なくとも1つの選択肢を含むawait関数を決定する(6100)ステップを含み、少なくとも1つの選択肢は関連するconfigured関数およびそれぞれの選択ステップ条件を有する。

【０２０７】

方法6000はさらに、await関数インスタンスを一度だけ実行する(6200)ステップを含み、少なくとも1つの選択肢によって必要とされる関係者の同意の指示を受け取り、少なくとも1つの選択肢から有効な選択肢の選択を受け取ると、方法はさらに、少なくとも1つの選択肢の関連するconfigured関数インスタンスを実行するステップを含み、await関数は、有効な選択肢の選択の選択ステップ条件が満たされるという決定で終了する。

【０２０８】

ステップ6100と6200のいずれかおよび/または両方が、少なくとも1つの選択肢によって要求される1つまたは複数の関係者の同意の指示を受け取ることから、および/または、関係者5001、5003と関連付けられるノード5002、5004によって送信される(6250)有効な選択肢の選択を受け取ることから開始され得ることを理解されたい。いくつかの例では、同意の指示は、明示的にまたは暗黙的に、選択肢の選択を含むことがある(選択肢の選択が同意の指示を含むこともある)。

【０２０９】

方法6000はさらに、デジタル資産と関連付けられる合意記録を作成するためにagree関数を決定するステップを含み、方法はさらに、合意に必要な関係者の同意を受け取ったことに基づいて、agree関数インスタンスを実行するステップを含む。このagree関数は、ノード5002、5004から送信される(6350)合意に必要な同意の指示を受け取ることによって開始され得る。

【０２１０】

方法6000はさらに、実行された関数インスタンスの結果を、記憶のために付記専用台帳に送信する(6400)ステップを含む。そして、ストレージデバイス5009は付記専用台帳にその結果を記憶する(6450)。

【０２１１】

方法100、6000のいくつかのさらなる例はさらに、実行されるときに、agree関数を無効にするために、または実行されないawait関数を無力化するために影響を受ける関係者の同意を必要とする、delete関数を決定する(6700)ステップを含み得る。このステップ6700は、関係者5001、5017と関連付けられるノード5002、5003から送信される(6750)、agree関数を無効化するための、または実行されないawait関数を無効にするための、影響を受ける関係者の同意の指示を受け取ったことに応答するものであり得る。方法はさらに、実行されるawait関数、agree関数、およびdelete関数の結果を、記憶のために付記専用台帳に送信する(6800)ステップを含む。そして、少なくとも1つのストレージデバイス5009がその結果を付記専用台帳に記憶し得る。

【０２１２】

上の方法およびシステムは、様々なデジタル資産に適用され得る。いくつかの例では、そのようなデジタル資産は、有形資産(たとえば、鉄鉱石、小麦、コーヒー、ゴールドなど)を表すことがあり、またはそれと関連付けられることがある。たとえば、ファンジブルなコモディティなどの、高度に取引されるコモディティの現代的な交換は、デジタル資産として表現される。例として、ゴールドは、ゴールド自体の物理的な交換を伴うことなく、(デジタル)トークンを使用して関係者の間でデジタル的に取引され得る。他の例では

、デジタル資産は、譲渡可能な証券(紙幣など)での支払いの権利などの、他の権利を表すものであり得る。封印、透かし、磁気帯などの偽造防止手段を組み込み得る従来の紙幣と同様に、デジタル資産も、デジタル資産の整合性を維持するためのセキュリティ機能を必要とする。これは、二重消費を防ぎ、監査に使用できる変更不可能な記録をとるための手段を組み込むことを含み得る。

【０２１３】

デジタル資産がどのようにモデリングされ展開し得るかの非限定的な例がここで説明される。この例では、デジタル資産は関連する権利(これは関係者間の関係の条件の中にあり得る)を含む。いくつかの例では、これらの条件および合意のいくつかは、(少なくとも部分的に)標準化されることがあり、または関係者間の合意のセットの中にあることがある。ある取引の状況において、たとえば、主合意が、取引者とその証券保管機関との関係を詳述し、証券保管業者がいつどのように行動するかを説明し、各々の責任を記述することがある。医療においては、主合意は、病院従事者と外来患者看護業者との間でのリスク共有の合意であり得る。主合意の運用は一部、この方法(たとえば、デジタル資産の展開)によって容易にされ得る。

【０２１４】

いくつかの例では、関係者間の関係は契約と見なされ得る(契約条件を含み得る合意に対して関係者の申し出および受け入れがあるという意味で)。大量のデジタル資産があることがあり、一部のデジタル資産は互いに類似していることがあり、または類似の特性を共有することがあるので、デジタル資産に関連する情報(契約条件など)を少なくとも一部含むテンプレートを含むことが有益であり得る。契約テンプレートは、合意ステートメントと、すべての適用可能なパラメータと、そのデータに基づいて行動する際に行われ得る選択とを含む。契約テンプレートは、受け入れ可能な入力および得られる出力を規定する。したがって、契約テンプレートは、契約が実装されるときに発生する状態の変化を記述する。契約テンプレートは、それが関与する関係者によって署名されるまで効力がない。契約テンプレートは、実際の名前、量、日付などではなく、プレースホルダーを含む。

【０２１５】

図49は、契約テンプレート7001を表す概略図を示す。1つの関係者(所有者7003など)は何らかの通貨を有する。所有者7003は、以下のことを行うための選択を行うことができる。

i. 銀行7007に金を預ける(第1の選択肢7011);または

ii. それを第2の関係者7005に売る(恐らく商品またはサービスの交換において)(第2の選択肢7013)

【０２１６】

第1の選択肢7011において、金は銀行7007の銀行口座に行き、第2の選択肢7013において、金は第2の関係者7005に行く。上で述べられたように、契約テンプレートは、実際の名前、量、日付などではなくプレースホルダーを含むことがあり、それは、これらが、契約者が選択および/または同意を提供する後の段階で決定され得るからである。このことは、契約テンプレートが複数の回数使用されること(および/または適宜変更されること)を可能にする。契約テンプレートは、取出しのためにデータストアの中のライブラリに記憶され得ることを理解されたい。いくつかの例では、契約テンプレートは、(関係者が下で論じられるように契約に入る場合)関係者の権利の少なくとも一部を形式化または定義し得る。

【０２１７】

アクティブな契約は、プレースホルダーが実際のデータと置き換えられた契約のインスタンスであり得る。これは受け入れられており、効力があり、参加者の少なくとも1つを拘束している。アクティブな契約は、上で説明された契約テンプレートの変更および/または実行される例であり得る。

【０２１８】

図50は、アクティブな契約7101の例の概略図を示し、所有者7003の詳細は特定の人

物Alice 7103と置き換えられている。同様に、第2の関係者7005はBob 7105として指定される。銀行7007は契約テンプレートにおいて定義されることがあり、または、特定の銀行がアクティブな契約7101において定義されることがある。したがって、アクティブな契約において、Aliceは金を持っており彼女には2つの選択肢がある。

i. 銀行7007に金を預ける(第1の選択肢7011);または

ii. それをBob 7105に売る(第2の選択肢7013)

【0219】

上で説明された方法では、アクティブな契約のこの状態は、await関数(またはより具体的にはawait関数のインスタンス)によってもたらされ得る。これは、Alice 7103に対する2つの選択肢7011、7013を含むawait関数インスタンス7201を示す、図51において示されている。

【0220】

Alice 7103が第1の選択肢7011(すなわち、「支払い」のオプション)を選ぶ場合、コードは、次に起きることが関連するconfigured関数7211の実行であると決定する。Aliceの選択および、Aliceの口座に金を預けることに対する銀行の合意(Aliceの銀行口座を開設したときの以前の合意または一時的な合意のいずれかによる)は、Aliceの選択に対する関係者の同意を示すものであり得る。この例では、関連するconfigured関数は、銀行がAliceの口座に金を預けるためのものである。それぞれの選択ステップ条件は、銀行7007がAliceの口座に預け入れを行ったという確認を受け取るという条件を含み得る。

【0221】

Alice 7103が第2の選択肢7103(すなわち、「売却」のオプション)を選ぶ場合、コードは、契約の新しいインスタンス7223が契約テンプレートに基づいて作成されると決定するが、今回は、Bobが通貨および預け入れと売却を行うための選択肢を持っている者である。すなわち、第2の選択肢7013に対する関連するconfigured関数は、契約7223の新しいインスタンスを作成することである。それぞれの選択ステップ条件は、契約の新しいインスタンス7223が作成されることの指示を含み得る。代わりに、または組み合わせて、別のそれぞれの選択ステップ条件は、Bob7105が有効な選択を行ったこと、または銀行7007が金をBobの口座に預けたことを含み得る。

【0222】

契約が使用され選択が行われたとき、それはもはや使用可能ではない。すなわち、ユーザは同じ資産を再び売ることはできず、または同じ通貨を2回預けることはできない。これは、有効な選択肢の選択ステップ条件が満たされるという決定で終了するawait関数によって達成され得る。

【0223】

選択ステップ条件は、二重消費を防ぎながら並列な選択ステップを可能にし得る。重要なことに、これは、より前の選択ステップのより前の選択ステップ条件が満足されない、もしくは満たされない場合、またはそのことが可能ではない場合、後続の並列の選択肢が有効に選択されることを可能にし得る。たとえば、Alice 7103が第1の選択肢7011を最初に選ぶが、銀行7007がAliceの口座に金を預けることが不可能である(たとえば、Aliceの口座が解約されていた)とする。したがって、Aliceは選択を行ったが、選択ステップ条件を満たすことができない。したがって、このことは、Aliceが第2の選択肢7013などの代替的な選択を行うことができるように、アクティブな契約を残す(およびアクティブなawait関数を通じて実行される)。これは、満足することができない選択にAliceが行き詰まる状況を防ぐ。

【0224】

契約が「使用される」と、await関数インスタンス7201は終了する(すなわち、有効な選択が行われるとき、それぞれの選択ステップ条件が満たされる)。したがって、その使用された契約は非アクティブな契約になり、もはやそれに基づいて行動することができない(すなわち、Aliceは選択を再び行うことができない)。これは二重消費を防ぐ。この

システムの特徴は、項目が削除されないので、契約(または契約と関連付けられる台帳に記憶されている情報)が付記専用台帳から削除されないということである。これは、システムの整合性を保ち、見直しおよび監査を助け得る。図52は、Aliceが有効な選択7013を行ったのでもはや他の選択を行うことができない、非アクティブな契約7101'(これは終了したawait関数インスタンスに相当する)を示す。この結果は、Bobが、Charlie 7106に売ることまたは銀行7007に預けることなどの、新しいawait関数インスタンス7223において選択を行うことができるというものである。

【０２２５】

しかしながら、定義された選択肢のうちの1つまたは複数が望ましくない、またはそれらを選ぶことができないいくつかの状況がある。これは、予測されなかった、またはモデリングできなかった事象に関連して、デジタル資産のための関連情報を記録することが必要である状況を含み得る。これは、デジタル資産に影響する「現実世界の」事象であり得る。したがって、システムは、現実世界の事象が記録されることを可能にするためのステップも含む。

【０２２６】

これは、デジタル資産と関連付けられる合意記録が付記専用台帳に記録されることを可能にするための、agree関数(またはそのインスタンス)を含み得る。agree関数は、合意に必要な関係者の同意を受け取ると実行され得る。たとえば、必要とされる関係者は、その権利が影響を受ける、または未解決である関係者であり得る。

【０２２７】

いくつかの例では、agree関数は、agree関数インスタンスおよび/または選択ステップ条件と関連付けられ得る。たとえば、選択肢の一部は、「現実世界の」または「台帳外の」アクションを実行することを含むことがあり、この実行には、agree関数インスタンスがこれを記録することが必要であり得る。上の例を参照すると、銀行がAliceまたはBobの口座に金を預けるとき、これは、システムの外部でその支払いに対する預金伝票または受領証をもたらし得る。したがって、Alice 7103および銀行7007、またはBob 7105および銀行7007は、預け入れが行われたことに彼らがともに同意することの指示を提供する必要があることがあり、この指示が、台帳上で合意記録としてこれを記録するための同意として使用されることがある。デジタル資産が有形商品を表す他の例では、agree関数は、ある量の石炭が届けられたことに両方の関係者が合意することなどの、商品の配達を記録するために使用され得る。

【０２２８】

delete関数インスタンス(およびdelete関数インスタンス)は、影響を受ける関係者の同意とともに、付記専用台帳に記録されている以前のエントリを無力化し、または無効にするために使用され得る。この関数は、agree関数とともに、付記専用台帳の記録から情報を削除することなく誤りを正すためのいくらかの柔軟性をシステムに与え得る。

【０２２９】

上の方法およびシステムは、ユーザが、特定の契約または契約テンプレートのためにコーディングを行い、様々な選択の結果を見るために契約を実行することを可能にし得る。このことは、ユーザが様々な契約のオプション(すなわち選択肢、これが次いでさらなる契約および追加の選択肢につながり得る)を見るためにドリルダウンを行うことを可能にし得る。このことは、ユーザが実質的にあらゆる経路をたどって、あらゆる有効な方法でモデリングされた契約と対話して結果を見ることを可能にする。いくつかの例では、このことはまた、1人または複数のユーザの契約のシミュレーションおよび対話を可能にし得る。したがって、いくつかの例では、このシステムおよび方法は分析ツールとなるように適合され得る。

【０２３０】

図53は、ノード5002、5004、5007と関連付けられ得る処理デバイス5008の例を示す。処理デバイス5008は、プロセッサ9310、メモリ9320、およびバス9330を介して互いに通信するインターフェースデバイス9340を含む。メモリ9320は、上で説明

された方法100、3900、6000を実施するための命令およびデータを記憶し、プロセッサ9310は、方法を実施するためにメモリ9320からの命令(コンピュータプログラムなど)を実行する。インターフェースデバイス9340は、通信ネットワークとの通信を、およびいくつかの例では、データストア5009ならびに他のノードなどのユーザインターフェースおよび周辺機器との通信を支援する、通信モジュールを含み得る。処理デバイス5008は独立のネットワーク要素であり得るが、処理デバイス5008は別のネットワーク要素の一部でもあり得ることに留意されたい。さらに、処理デバイス5008によって実行されるいくつかの機能は、複数のネットワーク要素に分散されてよい。たとえば、複数の処理デバイス5008は、セキュアなローカルエリアネットワークにおいて方法を実行し得る。

【０２３１】

本発明の概念は限定しない例示的な実施形態に関して例として説明されたが、他の代替、変更、および変形が、本明細書で開示される教示に基づいて関連技術分野の当業者には明らかになるであろう。したがって、添付の特許請求の範囲は、本明細書に記載される例示的な実施形態に対するすべてのそのような代替、変更、および変形、ならびに、本開示の範囲および趣旨内に入るそれらの均等物を含むことが意図されている。

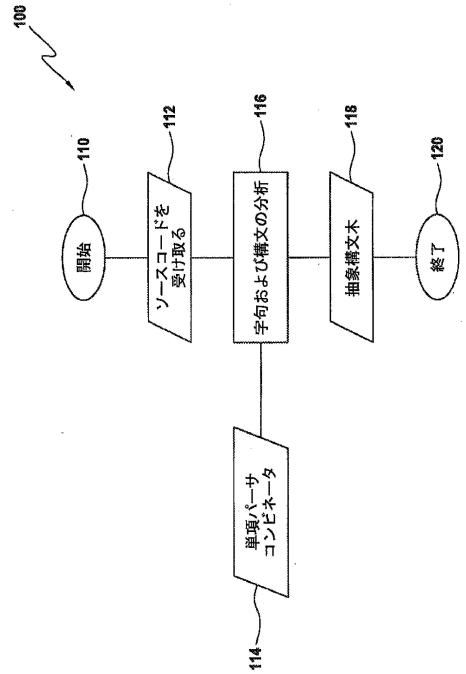
【符号の説明】

【０２３２】

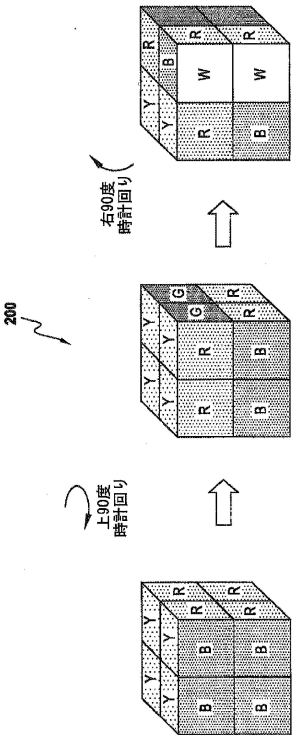
114 単項パーサコンビネータ
118 抽象構文木
4212 非公開共有台帳
4214 公開台帳
4312 非公開共有台帳層
4314 公開台帳層
4412 コア取引
5001 第1のユーザ
5002 第1のノード
5003 第2のユーザ
5004 第2のノード
5007 第3のノード
5008 処理デバイス
5009 ストレージデバイス
5011 データストア
5013 ピアツーピア分散型台帳
5015 処理デバイス
5017 別のユーザ
7001 契約テンプレート
7003 所有者
7005 第2の関係者
7007 銀行
7011 第1の選択肢
7013 第2の選択肢
7103 Alice
7105 Bob
9310 プロセッサ
9320 メモリ
9322 データ
9324 命令
9330 バス
9340 インターフェース

【図面】

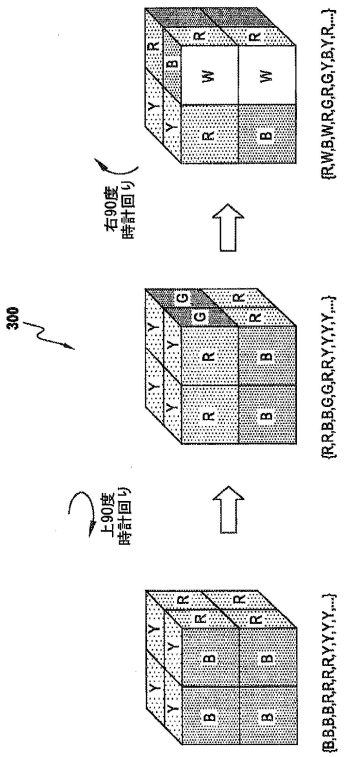
【図１】


100
110
開始
112
ソースコードを受け取る
114
単項パーサコンビネータ
116
字句および構文の分析
118
抽象構文木
120
終了

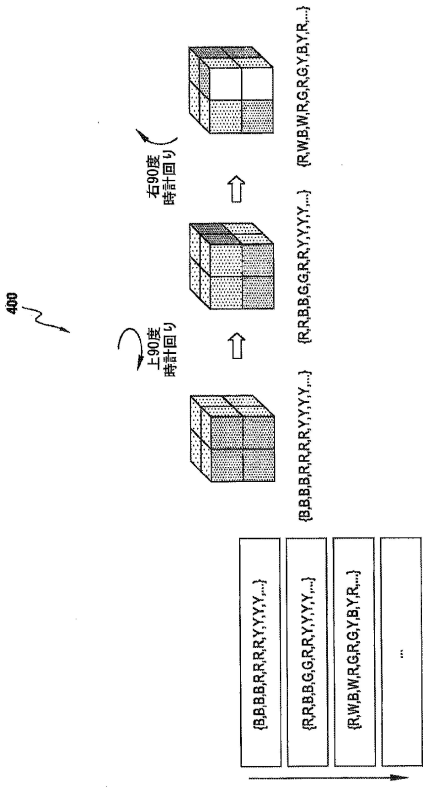
【図２】


200
上90度時計回り
右90度時計回り

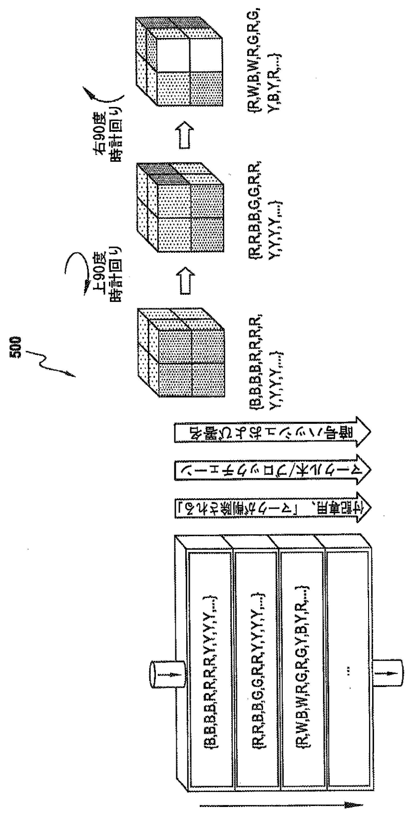
【図３】


300
上90度時計回り
右90度時計回り
{B,B,B,B,R,R,R,R,Y,Y,Y,Y,...}
{R,R,B,B,G,G,R,R,Y,Y,Y,Y,...}
{R,W,B,W,R,G,R,G,Y,B,Y,R,...}

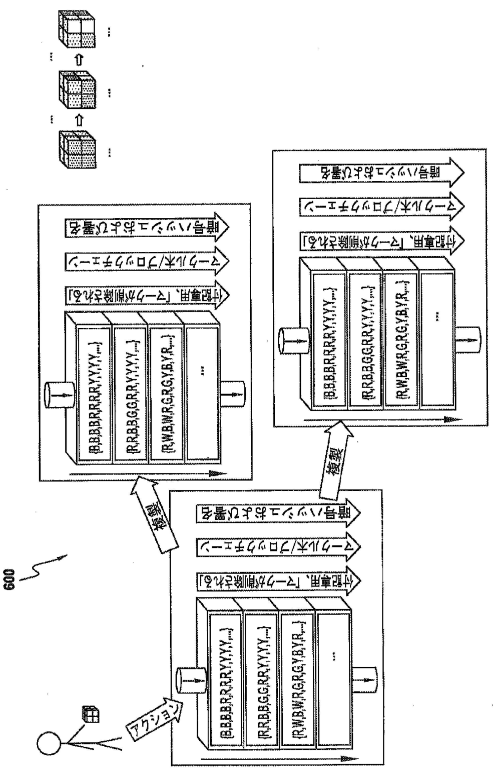
【図４】


400
上90度時計回り
右90度時計回り
{B,B,B,B,R,R,R,R,Y,Y,Y,Y,...}
{R,R,B,B,G,G,R,R,Y,Y,Y,Y,...}
{R,W,B,W,R,G,R,G,Y,B,Y,R,...}
{B,B,B,B,R,R,R,R,Y,Y,Y,Y,...}
{R,R,B,B,G,G,R,R,Y,Y,Y,Y,...}
{R,W,B,W,R,G,R,G,Y,B,Y,R,...}
...

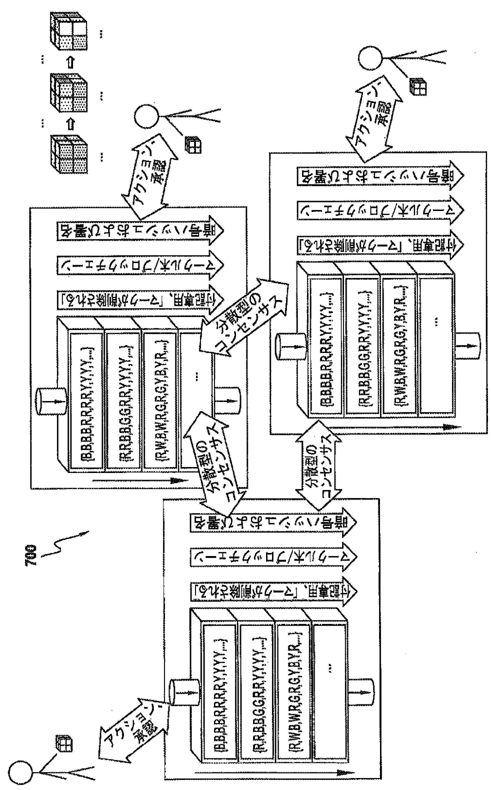
【図５】



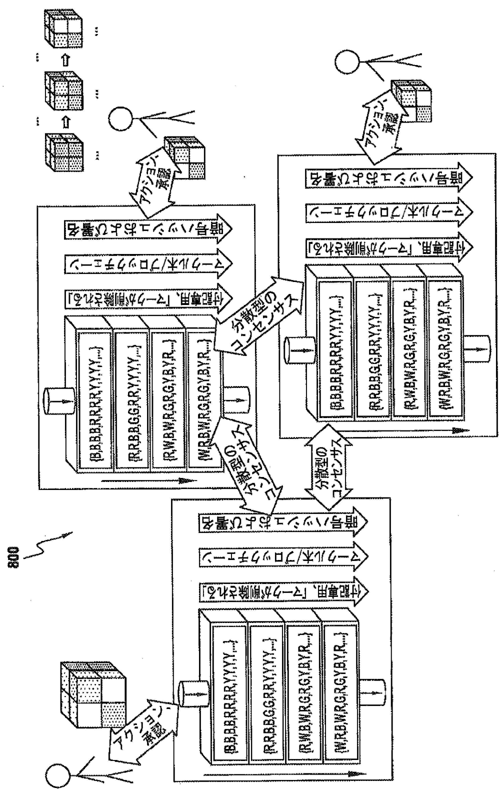
【図６】



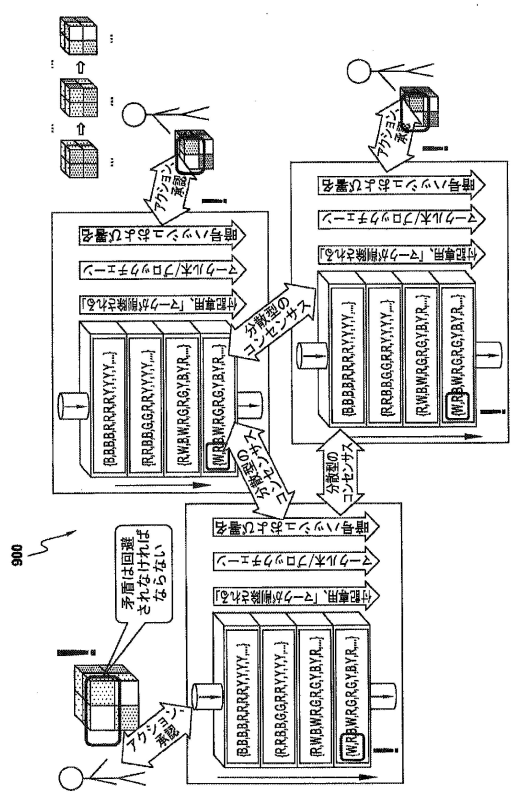
【図７】



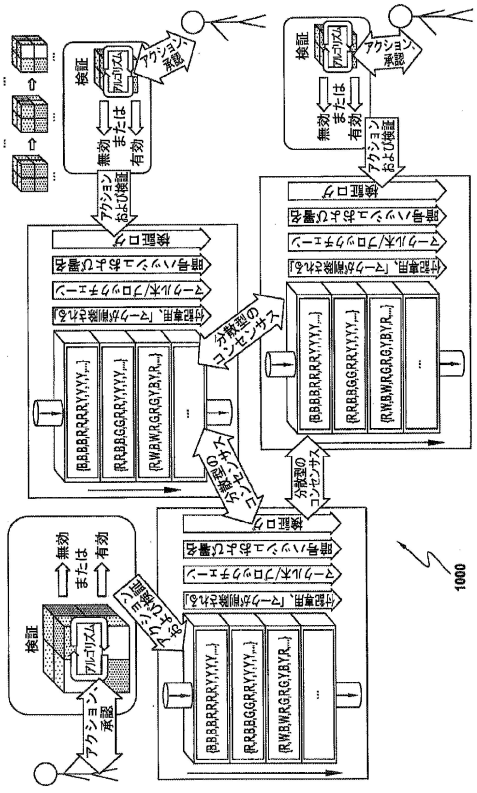
【図８】



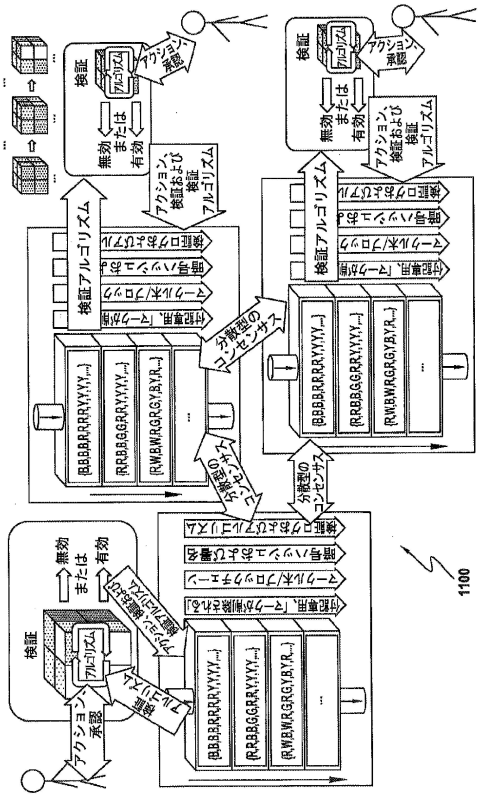
【図９】



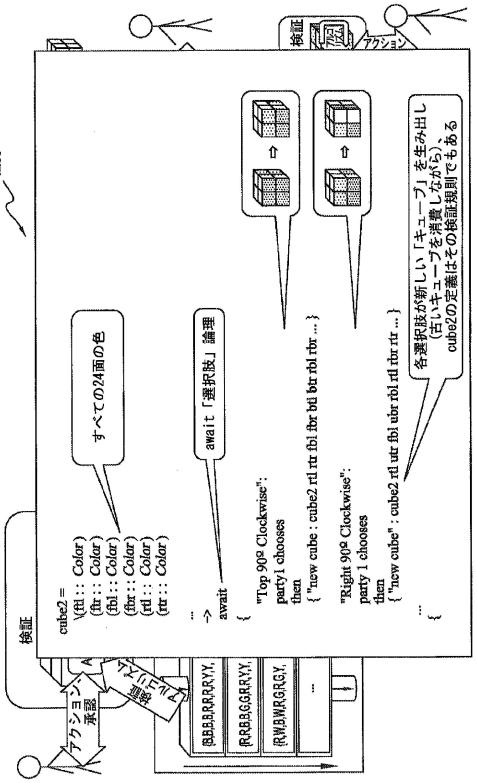
【図１０】



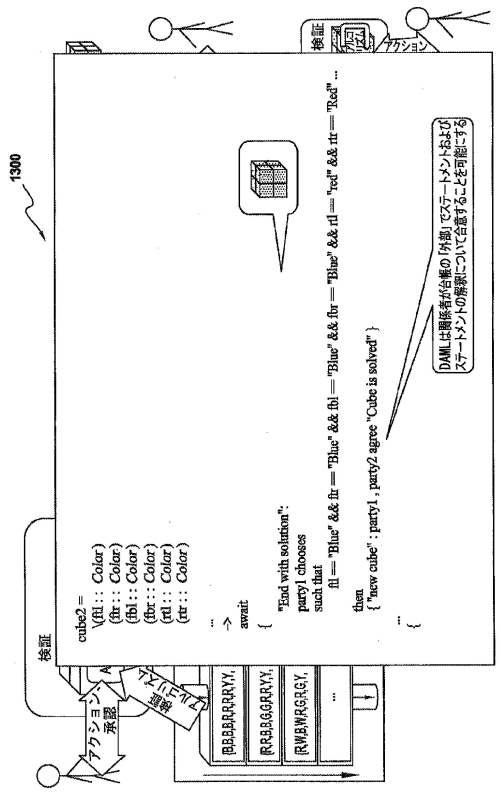
【図１１】



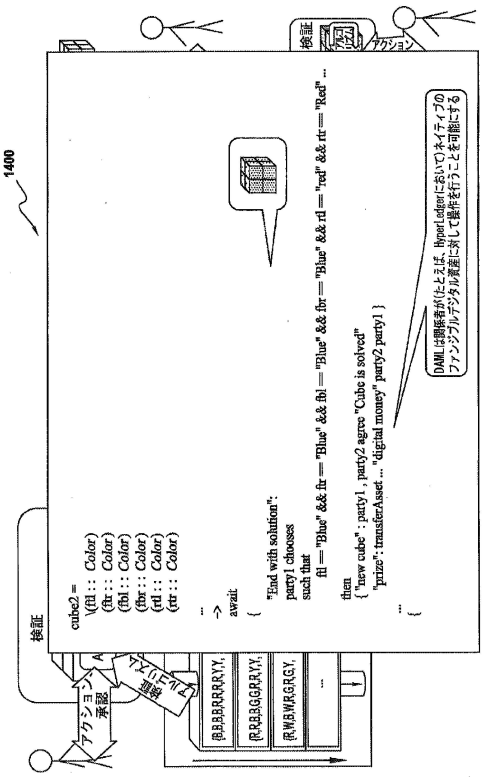
【図１２】



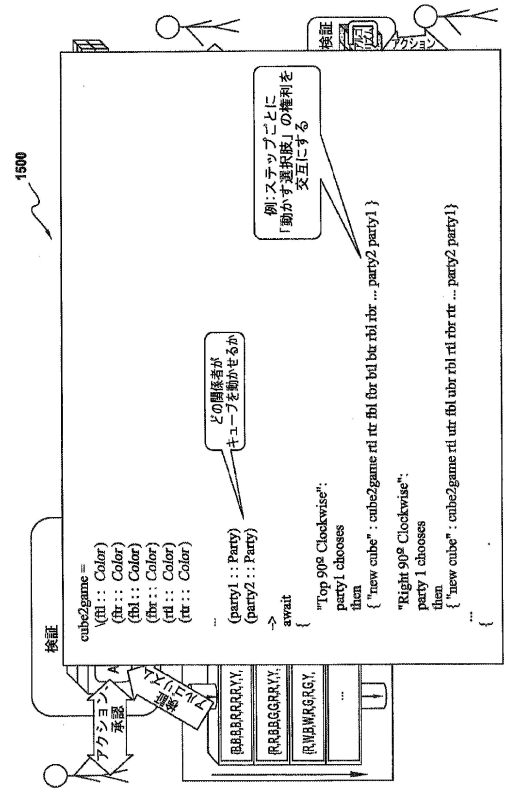
【図１３】



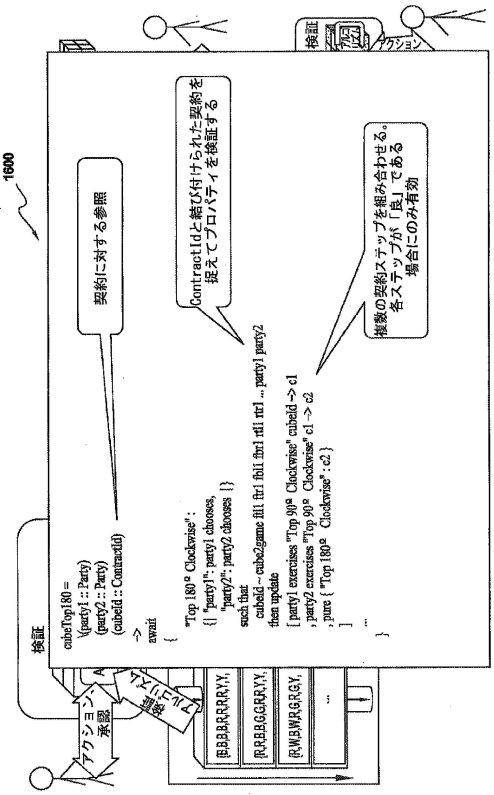
【図１４】



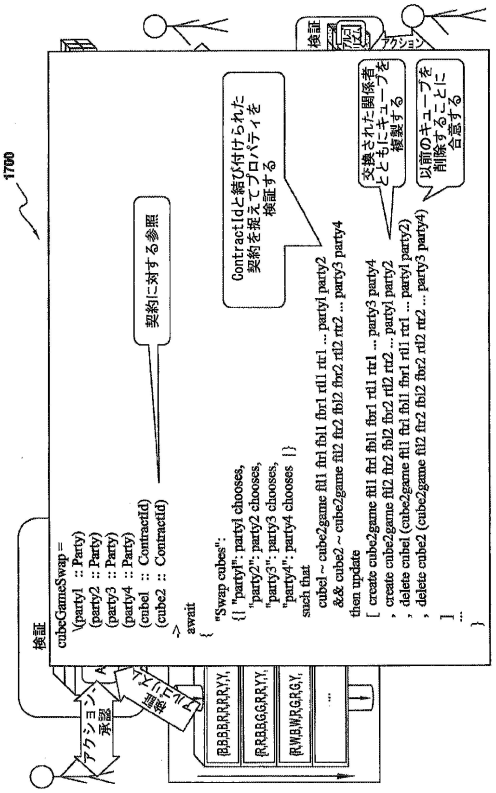
【図１５】



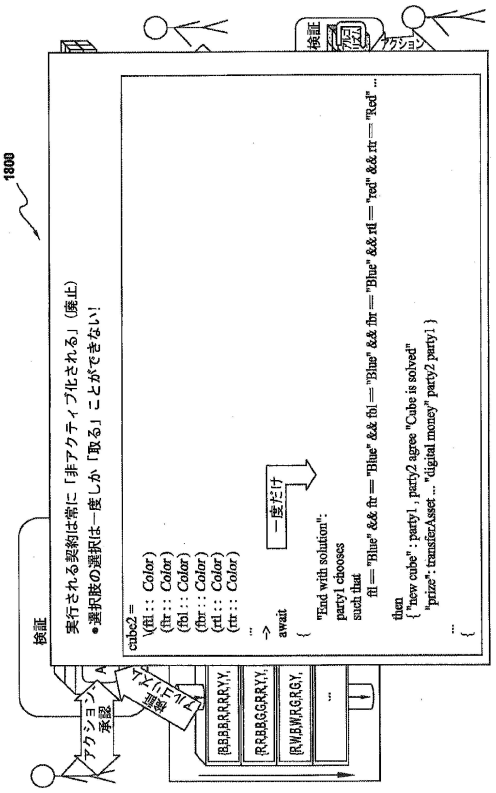
【図１６】



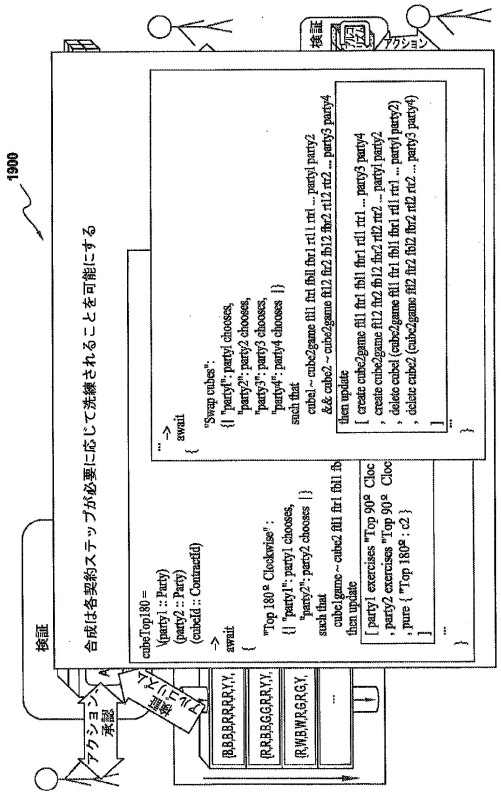
【図１７】



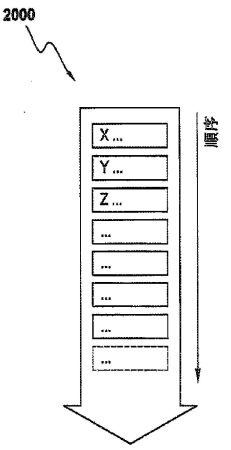
【図１８】



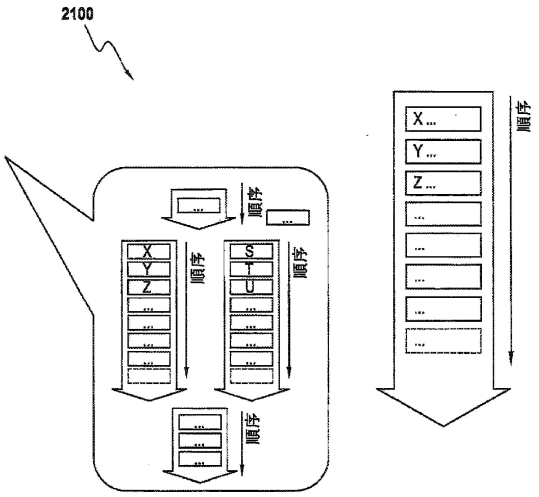
【図１９】



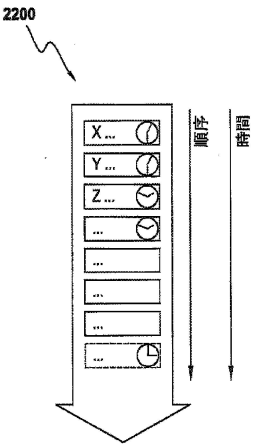
【図２０】



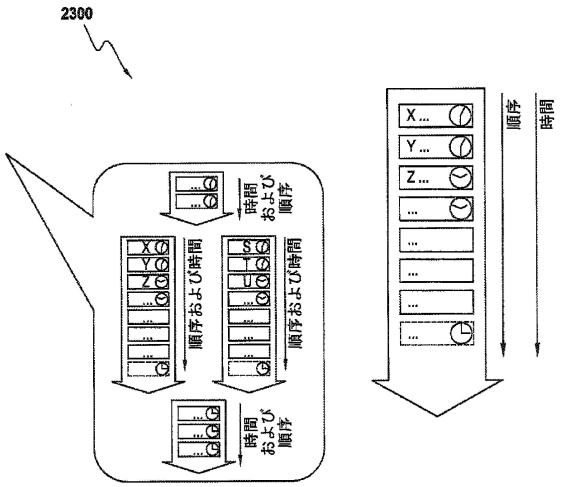
【図２１】



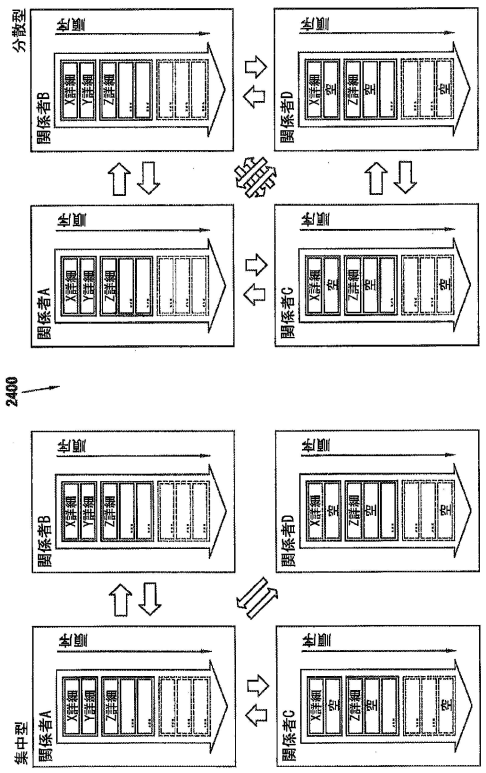
【図２２】



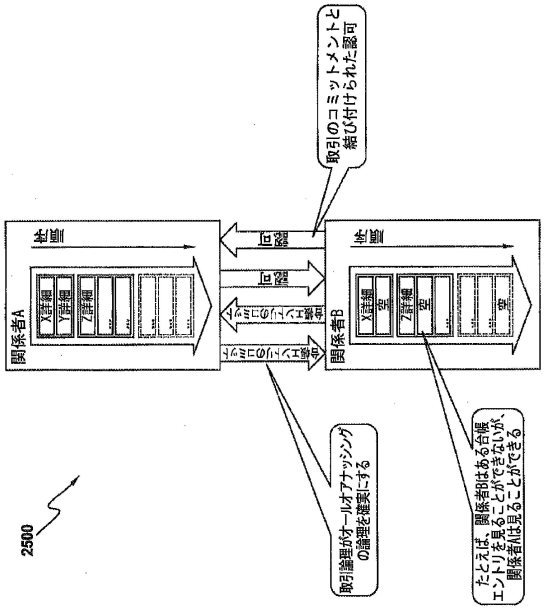
【図２３】



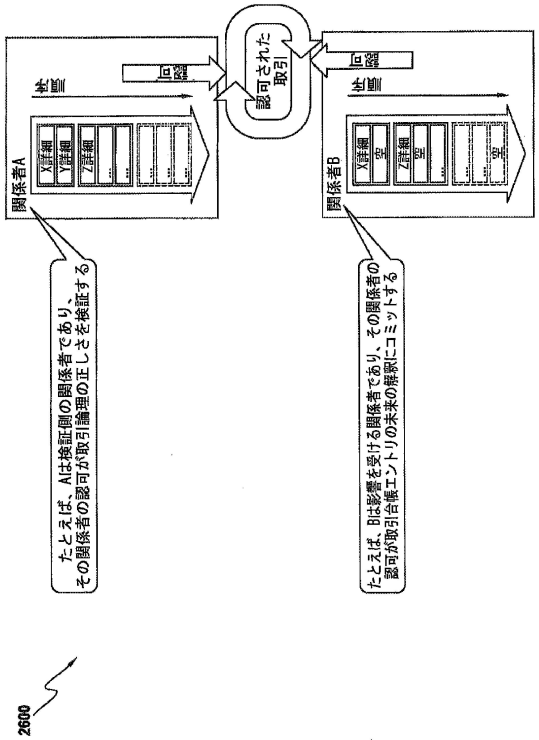
【図２４】



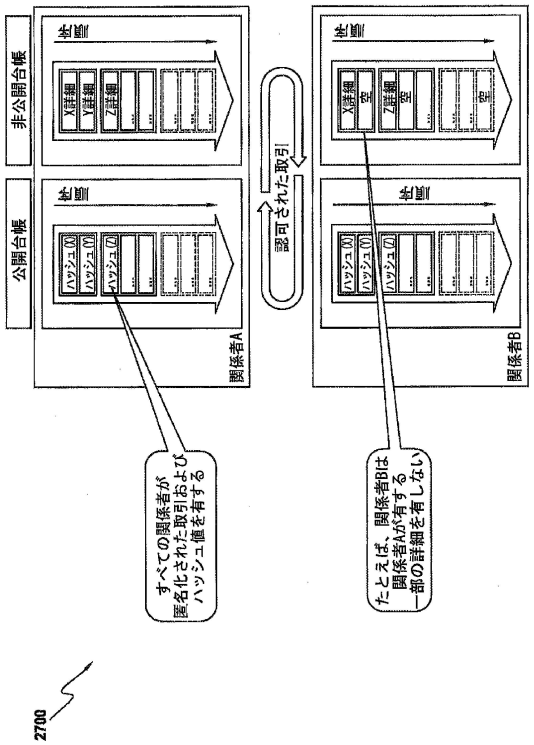
【図２５】



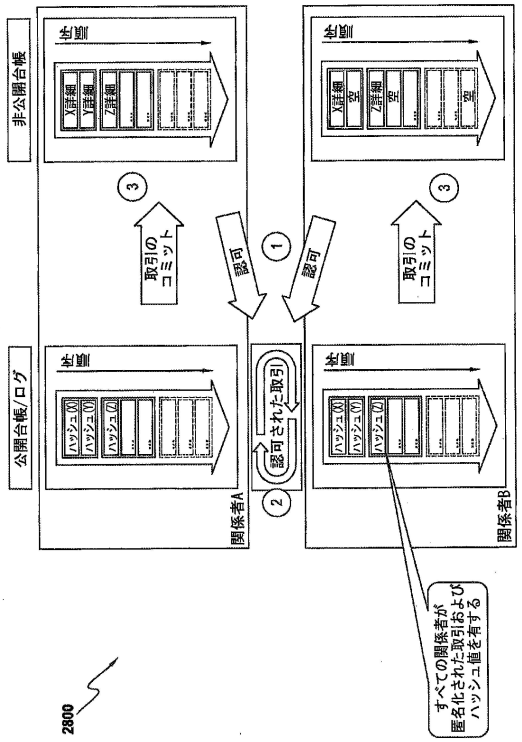
【図２６】



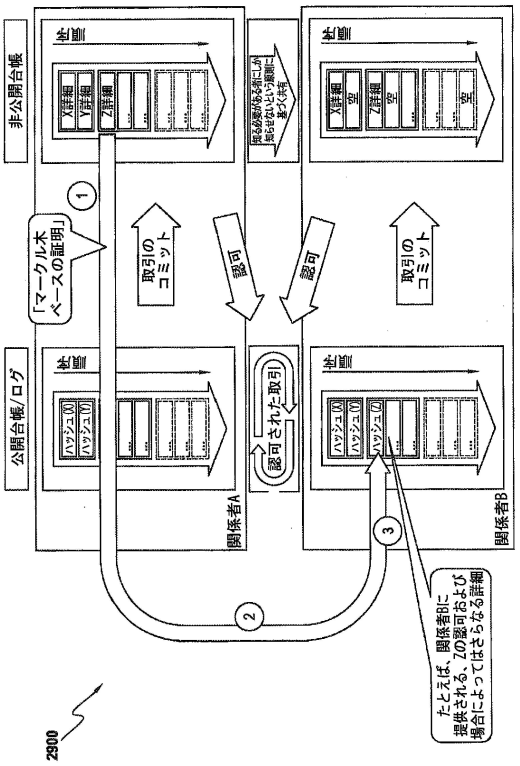
【図２７】



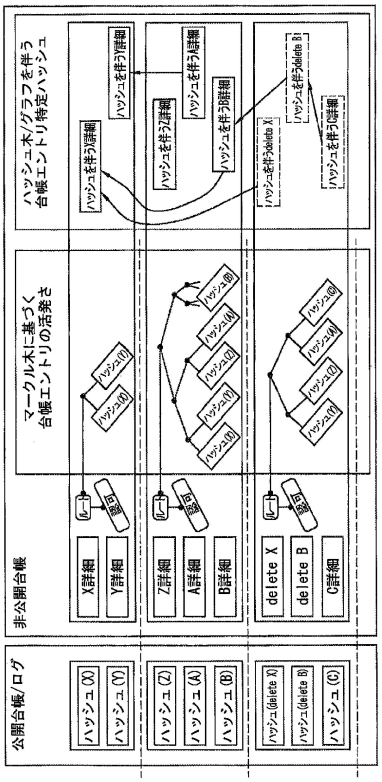
【図２８】



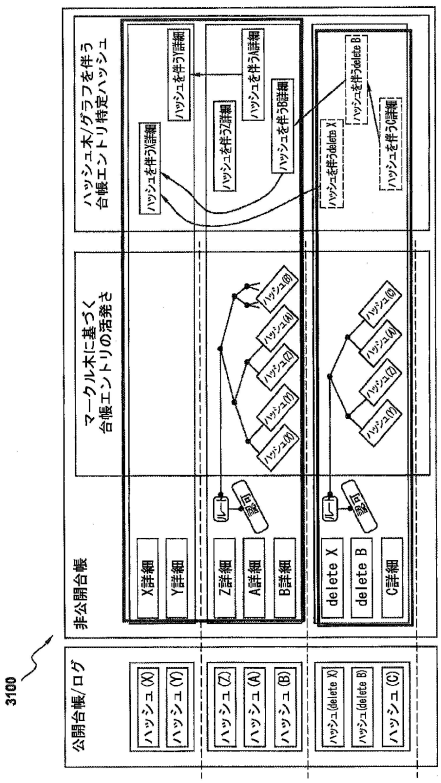
【図２９】



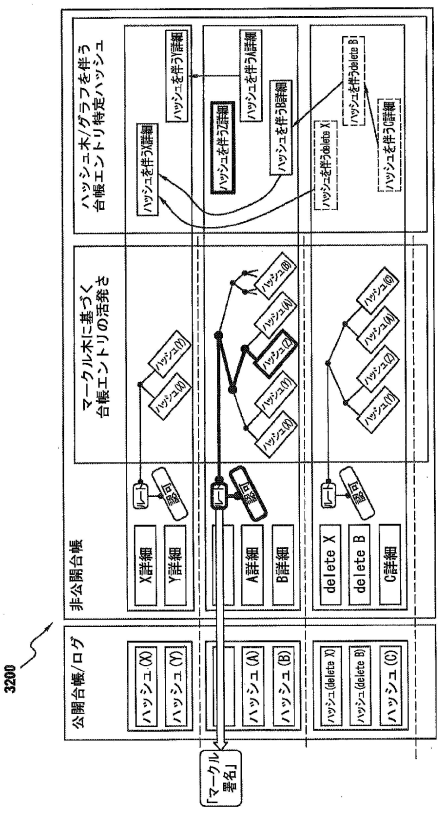
【図３０】



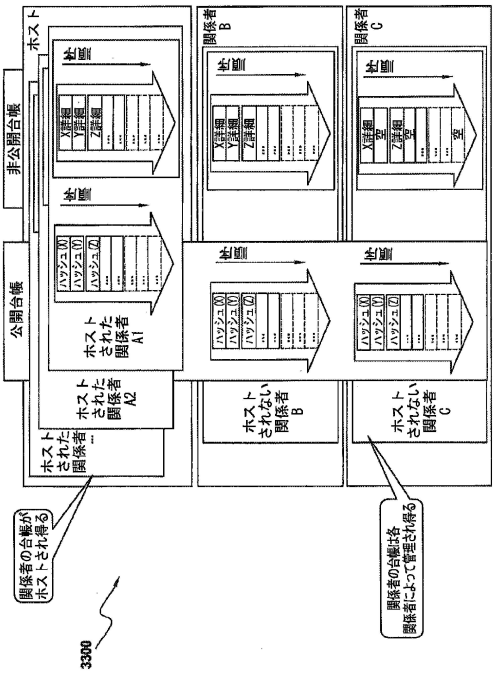
【図３１】



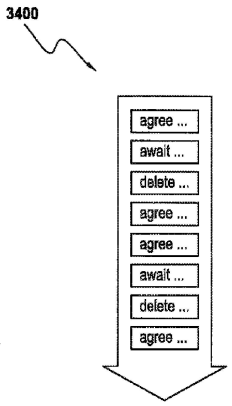
【図３２】



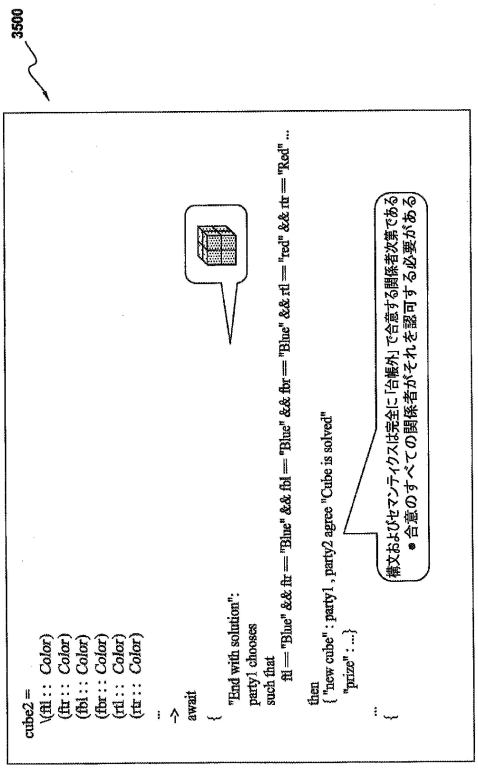
【図３３】



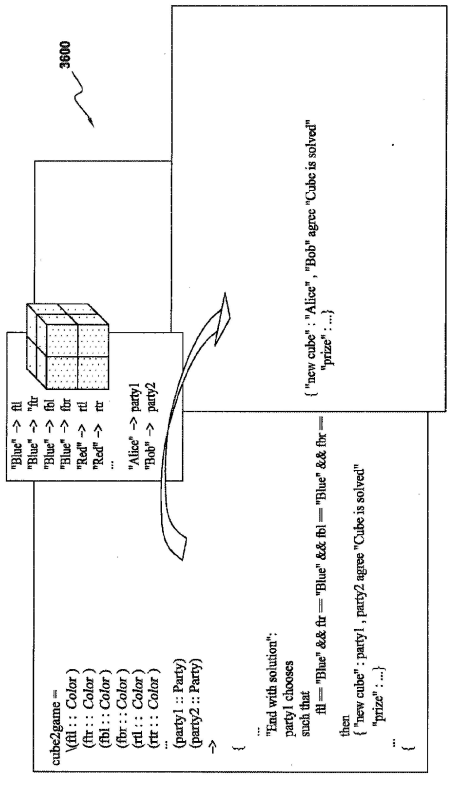
【図３４】



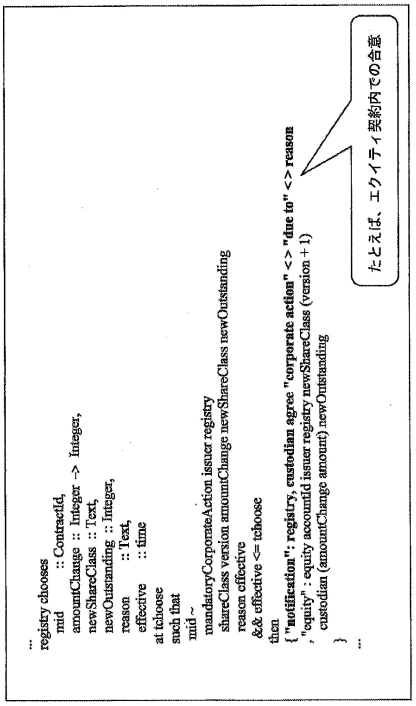
【図３５】



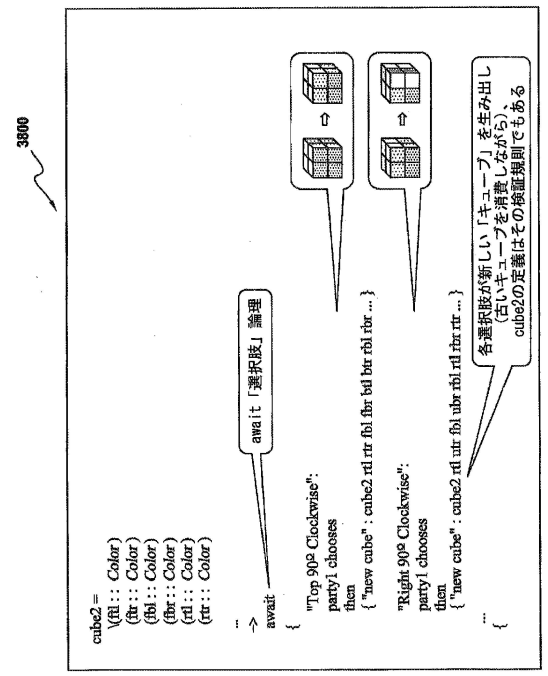
【図３６】



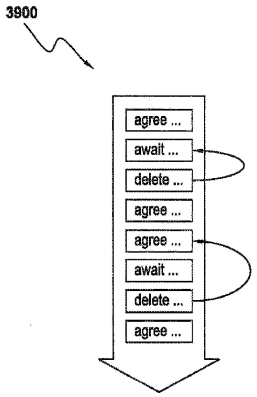
【図３７】



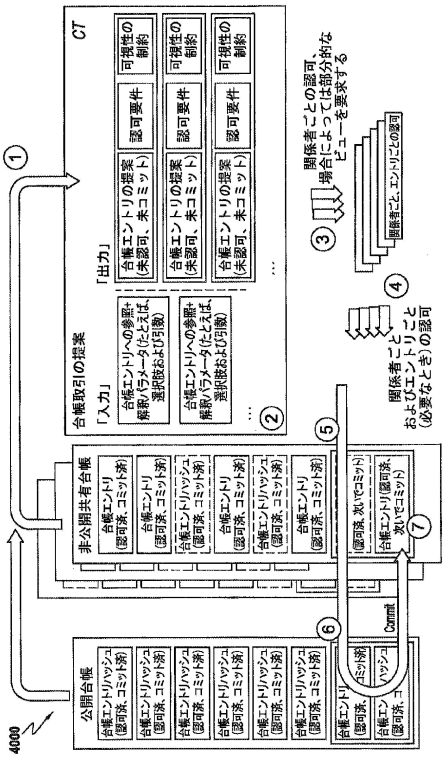
【図３８】



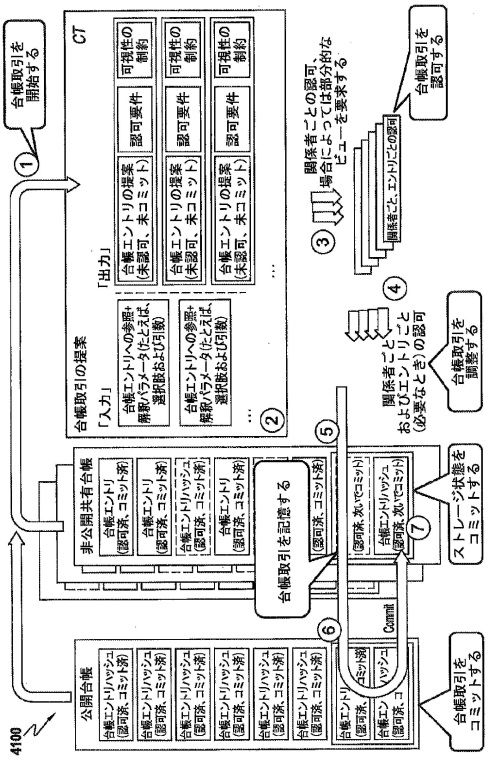
【図３９】



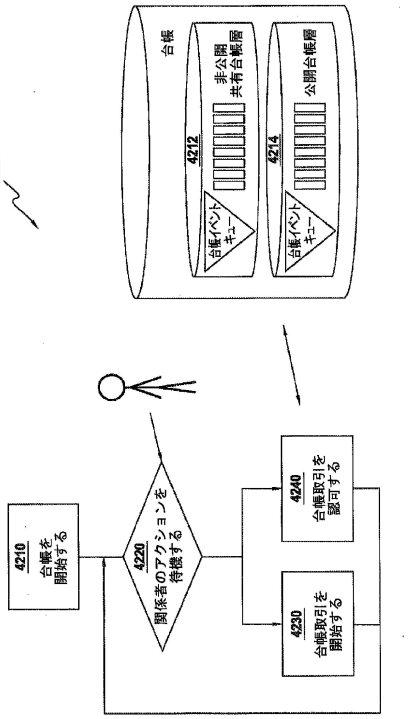
【図４０】



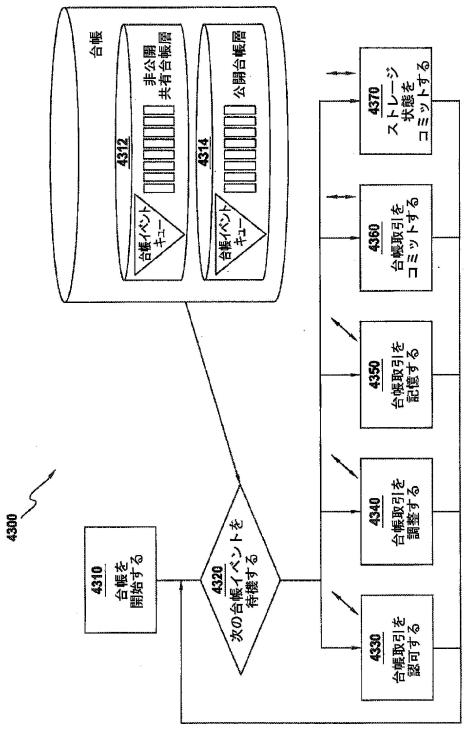
【図４１】



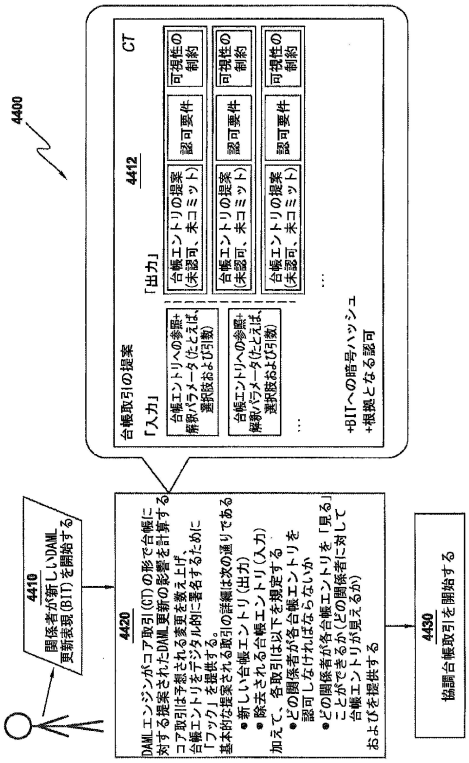
【図４２】



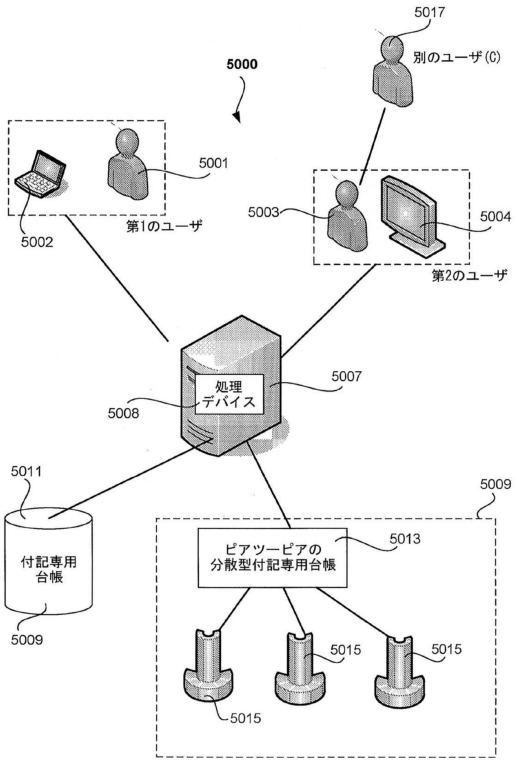
【図４３】



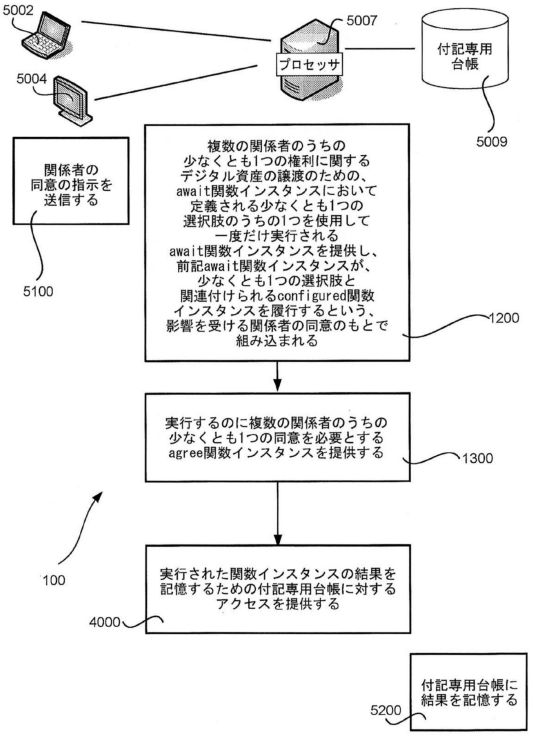
【図４４】



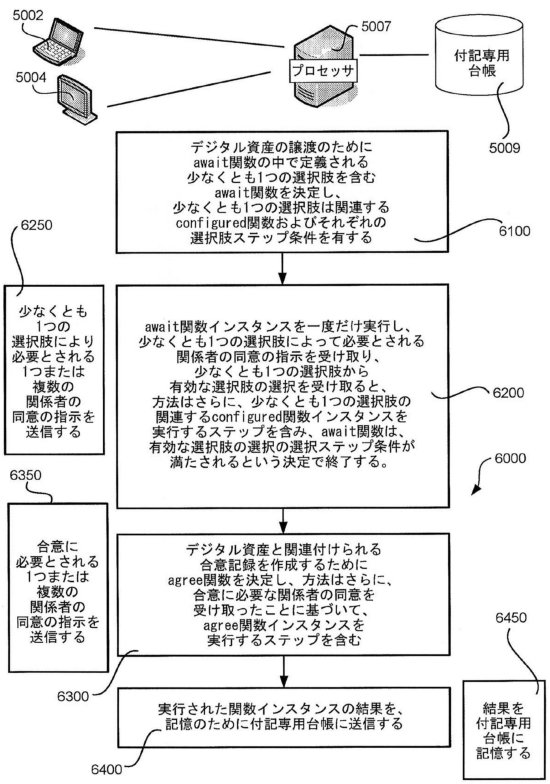
【図４５】



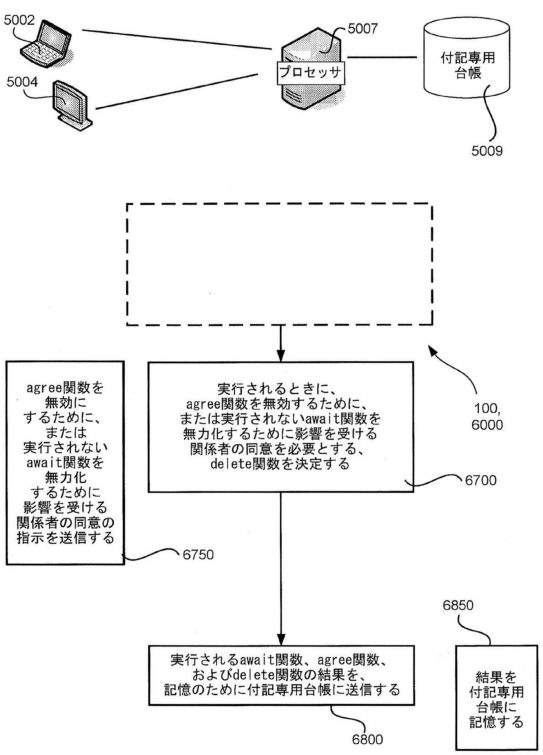
【図４６】



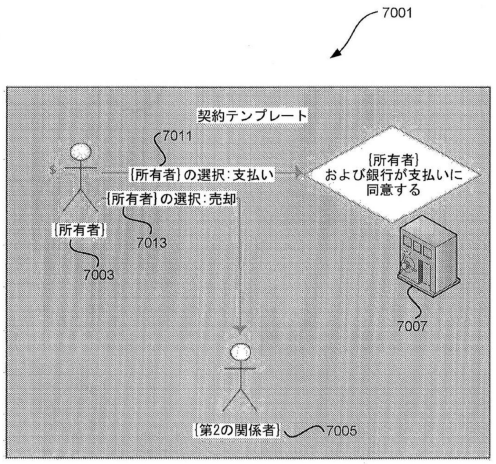
【図４７】



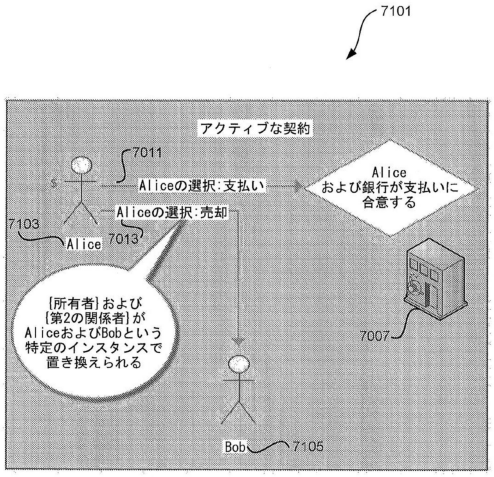
【図４８】



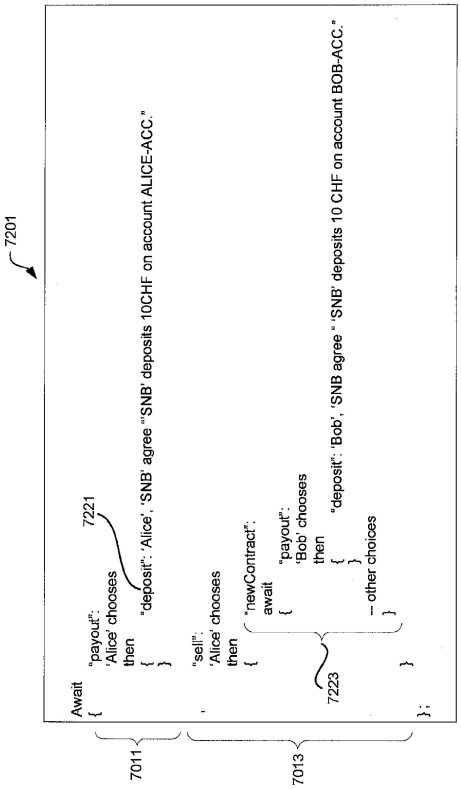
【図４９】



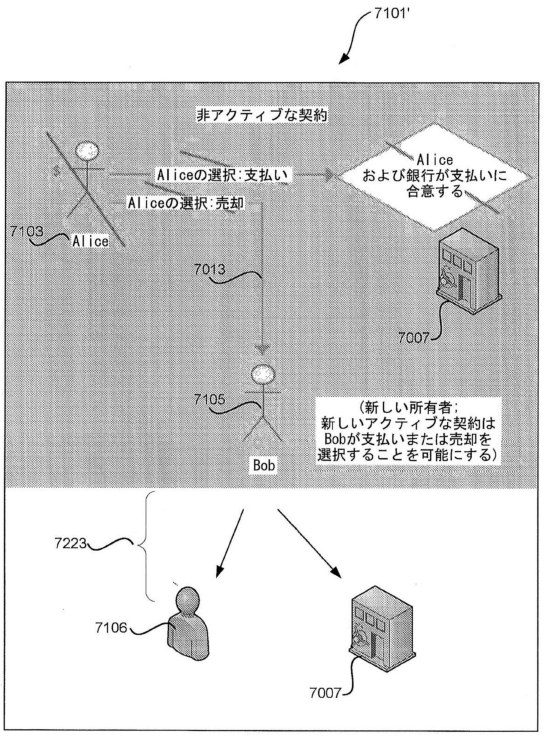
【図５０】



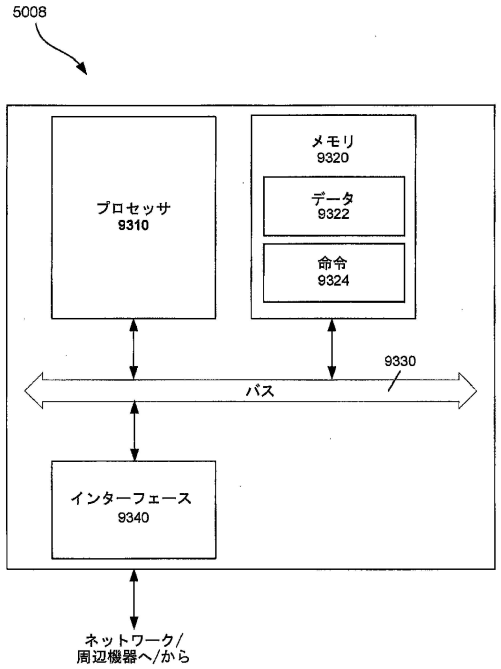
【図５１】



【図５２】



【図５３】


5008
プロセッサ
9310
メモリ
9320
データ
9322
命令
9324
9330
バス
インターフェース
9340
ネットワーク/
周辺機器へ/から

【手続補正書】
【提出日】令和5年3月31日(2023.3.31)
【手続補正１】
【補正対象書類名】特許請求の範囲
【補正対象項目名】全文
【補正方法】変更
【補正の内容】
【特許請求の範囲】
【請求項１】
デジタル資産と、複数の関係者の権利に関する前記デジタル資産の展開とをモデリングするための、データ構造を操作する、コンピュータで実施される方法であって、
１つまたは複数のコンピューティングデバイスによって、前記デジタル資産に関する前記複数の関係者の前記権利を決定するステップであって、前記デジタル資産は、第１の関数インスタンスの実行によって譲渡され、前記第１の関数インスタンスは、前記デジタル資産の処分のためにそこに定義された少なくとも第１の選択肢を含む、ステップと、
１つまたは複数のコンピューティングデバイスによって、コンピュータネットワークの複数の異なるノードによって維持される付記専用台帳を更新するように構成された提案された取引を生成するステップであって、前記提案された取引は、
（ｉ）前記複数の異なるノードのうちの第１のノードによって、前記提案された取引の即時および将来の台帳実行の影響を認可する、暗号的な認可と、
（ｉｉ）前記提案された取引の即時および将来の台帳実行の影響を暗号的に認可することができる、複数のノードの認可する部分集合を識別するデータと、
（ｉｉｉ）前記取引に関連する情報を受信する権利を有する前記複数のノードの秘密部分集合を識別するデータと、
を含む、生成するステップと、
前記１つまたは複数のコンピューティングデバイスによって、認可ノードのそれぞれが提案された取引の暗号的な認可を提供したことを検証するステップと、
前記１つまたは複数のコンピューティングデバイスによって、前記第１の関数インスタンスを実行し、実行された第１の関数インスタンスの結果を、コミットされた取引として付記専用台帳に格納するステップと、を含む、コンピュータで実施される方法。
【請求項２】
前記提案された取引は、前記提案された取引内で作成および非アクティブ化される過渡台帳エントリを含む、請求項１に記載のコンピュータで実施される方法。
【請求項３】
前記提案された取引は、１つまたは複数の台帳エントリを含む出力を含む、請求項１に記載のコンピュータで実施される方法。
【請求項４】
前記提案された取引は、台帳エントリの部分集合のみを第１のノードが暗号的に認可することをさらに含む、請求項３に記載のコンピュータで実施される方法。
【請求項５】
前記台帳エントリの部分集合は、マークル署名を用いて暗号的に認可される、請求項４に記載のコンピュータで実施される方法。
【請求項６】
各台帳エントリは、付記専用台帳に記録された既存の台帳エントリを破棄するコマンドか、または新しい台帳エントリを作成して当該新しい台帳エントリを付記専用台帳に記録するコマンドのいずれかである、請求項３に記載のコンピュータで実施される方法。
【請求項７】
請求項１の（ｉｉ）は、前記提案された取引の出力に含まれるそれぞれの台帳エントリについて、それぞれの台帳エントリを暗号的に認可しなければならない１つ以上の台帳エントリ認可ノードを識別する台帳エントリごとの認可データを含む、請求項３に記載のコン

ピュータで実施される方法。
【請求項８】
前記方法は、前記台帳エントリ認可ノードにのみ前記台帳エントリごとの認可データを送信するステップであって、前記コンピュータネットワーク内の他のノードには送信しないステップをさらに含む、請求項７に記載のコンピュータで実施される方法。
【請求項９】
前記方法は、前記提案された取引の前記出力に含まれる各台帳エントリが、各台帳エントリ認可ノードによって暗号的に認可されていることを検証するステップをさらに含む、請求項７に記載のコンピュータで実施される方法。
【請求項１０】
前記方法は、前記提案された取引の暗号的に保護されたバージョンを、前記提案された取引によって影響を受けない前記コンピュータネットワークの前記複数の異なるノードの部分集合に送信するステップをさらに含む、請求項１に記載のコンピュータで実施される方法。
【請求項１１】
前記方法は、前記提案された取引の可読バージョンを、取引に係る情報を受信する権利を有する秘密ノードのみに送信するステップであって、前記コンピュータネットワーク内の他のノードには送信しないステップをさらに含む、請求項１に記載のコンピュータで実施される方法。
【請求項１２】
前記付記専用台帳は、少なくとも前記秘密ノード間で共有される秘密台帳を含み、前記秘密台帳はコミットされた取引を格納する、請求項１に記載のコンピュータで実施される方法。
【請求項１３】
前記付記専用台帳は、前記コンピュータネットワークの前記複数のノード間で共有される公開台帳を含み、前記公開台帳は、前記コミットされた取引の暗号化技術を用いた表現を格納する、請求項１２に記載のコンピュータで実施される方法。
【請求項１４】
前記コミットされた取引の前記暗号化技術を用いた表現は、前記コミットされた取引のハッシュを含む、請求項１３に記載のコンピュータで実施される方法。
【外国語明細書】
2023078372000055.pdf

フロントページの続き

１．ＪＡＶＡＳＣＲＩＰＴ

(72)発明者　ラトコ・ヴェプレク
アメリカ合衆国・ニューヨーク・１００１０・ニュー・ヨーク・フィフス・アヴェニュー・１６２・スイート・９０２内
(72)発明者　ヨハン・ショディン
アメリカ合衆国・ニューヨーク・１００１０・ニュー・ヨーク・フィフス・アヴェニュー・１６２・スイート・９０２内
(72)発明者　ジェームズ・リトシオ
アメリカ合衆国・ニューヨーク・１００１０・ニュー・ヨーク・フィフス・アヴェニュー・１６２・スイート・９０２内
(72)発明者　アレクサンダー・バ－ナウアー
アメリカ合衆国・ニューヨーク・１００１０・ニュー・ヨーク・フィフス・アヴェニュー・１６２・スイート・９０２内
(72)発明者　ダルコ・ピラヴ
アメリカ合衆国・ニューヨーク・１００１０・ニュー・ヨーク・フィフス・アヴェニュー・１６２・スイート・９０２内
(72)発明者　ロビン・クロム
アメリカ合衆国・ニューヨーク・１００１０・ニュー・ヨーク・フィフス・アヴェニュー・１６２・スイート・９０２内
(72)発明者　サイモン・マイヤー
アメリカ合衆国・ニューヨーク・１００１０・ニュー・ヨーク・フィフス・アヴェニュー・１６２・スイート・９０２内
(72)発明者　シャウル・クフィル
アメリカ合衆国・ニューヨーク・１００１０・ニュー・ヨーク・フィフス・アヴェニュー・１６２・スイート・９０２内