From Verifiability to Accountability

Fully Exploiting Distributed Ledgers for Digital Contracts (abbreviated)

Allen L. Brown, Jr., Ph.D.

Deixis, PBC
Seattle, WA

abrown@deixis.digital

W3C TPAC Lyon
October 25, 2018

Outline

In this section:

Trust

Contracts are instruments of trust:

- that the participating parties have a common understanding of the contract's *actionable* obligations and permissions
- that the execution of the contract is verifiable
- that contract defaulters are discoverable and can be held accountable

Trust

Contracts are instruments of trust:

- that the participating parties have a common understanding of the contract's *actionable* obligations and permissions
- that the execution of the contract is verifiable
- that contract defaulters are discoverable and can be held accountable

Trust

Contracts are instruments of trust:

- that the participating parties have a common understanding of the contract's *actionable* obligations and permissions
- that the execution of the contract is verifiable
- that contract defaulters are discoverable and can be held accountable

Trust

Contracts are instruments of trust:

- that the participating parties have a common understanding of the contract's *actionable* obligations and permissions
- that the execution of the contract is verifiable
- that contract defaulters are discoverable and can be held accountable

Sketch of a contract

The parties to the contract are:

- Alice, a health care provider
- Ted, a patient
- Carol, a payer

The terms of the contract are:

- Alice, the provider, treats Ted, the patient, and awaits acknowledgment of treatment from Ted
- Ted accepts treatment from Alice, acknowledges to Alice having received treatment and alerts Carol, the payer, that he has been treated
- Alice, upon being acknowledged by Ted, requests payment from Carol
- Carol upon receiving Ted's alert and Alice's request, pays Alice
- Upon payment from Carol, Alice is once again ready to treat Ted

Sketch of a contract

The parties to the contract are:

- Alice, a health care provider
- Ted, a patient
- Carol, a payer

The terms of the contract are:

- Alice, the provider, treats Ted, the patient, and awaits acknowledgment of treatment from Ted
- Ted accepts treatment from Alice, acknowledges to Alice having received treatment and alerts Carol, the payer, that he has been treated
- Alice, upon being acknowledged by Ted, requests payment from Carol
- Carol upon receiving Ted's alert and Alice's request, pays Alice
- Upon payment from Carol, Alice is once again ready to treat Ted

Sketch of a contract

The parties to the contract are:

- Alice, a health care provider
- Ted, a patient
- Carol, a payer

The terms of the contract are:

- Alice, the provider, treats Ted, the patient, and awaits acknowledgment of treatment from Ted
- Ted accepts treatment from Alice, acknowledges to Alice having received treatment and alerts Carol, the payer, that he has been treated
- Alice, upon being acknowledged by Ted, requests payment from Carol
- Carol upon receiving Ted's alert and Alice's request, pays Alice
- Upon payment from Carol, Alice is once again ready to treat Ted

Sketch of a contract

The parties to the contract are:

- Alice, a health care provider
- Ted, a patient
- Carol, a payer

The terms of the contract are:

- Alice, the provider, treats Ted, the patient, and awaits acknowledgment of treatment from Ted
- Ted accepts treatment from Alice, acknowledges to Alice having received treatment and alerts Carol, the payer, that he has been treated
- Alice, upon being acknowledged by Ted, requests payment from Carol
- Carol upon receiving Ted's alert and Alice's request, pays Alice
- Upon payment from Carol, Alice is once again ready to treat Ted

Sketch of a contract

The parties to the contract are:

- Alice, a health care provider
- Ted, a patient
- Carol, a payer

The terms of the contract are:

- Alice, the provider, treats Ted, the patient, and awaits acknowledgment of treatment from Ted
- Ted accepts treatment from Alice, acknowledges to Alice having received treatment and alerts Carol, the payer, that he has been treated
- Alice, upon being acknowledged by Ted, requests payment from Carol
- Carol upon receiving Ted's alert and Alice's request, pays Alice
- Upon payment from Carol, Alice is once again ready to treat Ted

Sketch of a contract

The parties to the contract are:

- Alice, a health care provider
- Ted, a patient
- Carol, a payer

The terms of the contract are:

- Alice, the provider, treats Ted, the patient, and awaits acknowledgment of treatment from Ted
- Ted accepts treatment from Alice, acknowledges to Alice having received treatment and alerts Carol, the payer, that he has been treated
- Alice, upon being acknowledged by Ted, requests payment from Carol
- Carol upon receiving Ted's alert and Alice's request, pays Alice
- Upon payment from Carol, Alice is once again ready to treat Ted

Sketch of a contract

The parties to the contract are:

- Alice, a health care provider
- Ted, a patient
- Carol, a payer

The terms of the contract are:

- Alice, the provider, treats Ted, the patient, and awaits acknowledgment of treatment from Ted
- Ted accepts treatment from Alice, acknowledges to Alice having received treatment and alerts Carol, the payer, that he has been treated
- Alice, upon being acknowledged by Ted, requests payment from Carol
- Carol upon receiving Ted's alert and Alice's request, pays Alice
- Upon payment from Carol, Alice is once again ready to treat Ted

Sketch of a contract

The parties to the contract are:

- Alice, a health care provider
- Ted, a patient
- Carol, a payer

The terms of the contract are:

- Alice, the provider, treats Ted, the patient, and awaits acknowledgment of treatment from Ted
- Ted accepts treatment from Alice, acknowledges to Alice having received treatment and alerts Carol, the payer, that he has been treated
- Alice, upon being acknowledged by Ted, requests payment from Carol
- Carol upon receiving Ted's alert and Alice's request, pays Alice
- Upon payment from Carol, Alice is once again ready to treat Ted

Sketch of a contract

The parties to the contract are:

- Alice, a health care provider
- Ted, a patient
- Carol, a payer

The terms of the contract are:

- Alice, the provider, treats Ted, the patient, and awaits acknowledgment of treatment from Ted
- Ted accepts treatment from Alice, acknowledges to Alice having received treatment and alerts Carol, the payer, that he has been treated
- Alice, upon being acknowledged by Ted, requests payment from Carol
- Carol upon receiving Ted's alert and Alice's request, pays Alice
- Upon payment from Carol, Alice is once again ready to treat Ted

Sketch of a contract

The parties to the contract are:

- Alice, a health care provider
- Ted, a patient
- Carol, a payer

The terms of the contract are:

- Alice, the provider, treats Ted, the patient, and awaits acknowledgment of treatment from Ted
- Ted accepts treatment from Alice, acknowledges to Alice having received treatment and alerts Carol, the payer, that he has been treated
- Alice, upon being acknowledged by Ted, requests payment from Carol
- Carol upon receiving Ted's alert and Alice's request, pays Alice
- Upon payment from Carol, Alice is once again ready to treat Ted

Sketch of a contract

The parties to the contract are:

- Alice, a health care provider
- Ted, a patient
- Carol, a payer

The terms of the contract are:

- Alice, the provider, treats Ted, the patient, and awaits acknowledgment of treatment from Ted
- Ted accepts treatment from Alice, acknowledges to Alice having received treatment and alerts Carol, the payer, that he has been treated
- Alice, upon being acknowledged by Ted, requests payment from Carol
- Carol upon receiving Ted's alert and Alice's request, pays Alice
- Upon payment from Carol, Alice is once again ready to treat Ted

Contracts, smart and otherwise

Contracts	**Analog**	**Digital**	
	Traditional	*Smart*	*Deixis*
Syntax	Late Renaissance English with Latin neologisms	Yet another OO language	Domain specific language for specifying behavior only
Semantics	What your lawyer says	What your programmer says	Mathematically defined categorical, logical and operational semantics
Author	Your lawyer	Your programmer	Your DBA
Performance tracking	Humint	Distributed ledger	Distributed ledger
Guarantees	What your lawyer says	What your programmer says	What your specification says
Violations	When your lawyer says	When your programmer says	When your spec checker says

Contracts, smart and otherwise

Contracts	**Analog**	**Digital**	
	Traditional	*Smart*	*Deixis*
Syntax	Late Renaissance English with Latin neologisms	Yet another OO language	Domain specific language for specifying behavior only
Semantics	What your lawyer says	What your programmer says	Mathematically defined categorical, logical and operational semantics
Author	Your lawyer	Your programmer	Your DBA
Performance tracking	Humint	Distributed ledger	Distributed ledger
Guarantees	What your lawyer says	What your programmer says	What your specification says
Violations	When your lawyer says	When your programmer says	When your spec checker says

Contracts, smart and otherwise

Contracts	**Analog**	**Digital**	
	Traditional	*Smart*	*Deixis*
Syntax	Late Renaissance English with Latin neologisms	Yet another OO language	Domain specific language for specifying behavior only
Semantics	What your lawyer says	What your programmer says	Mathematically defined categorical, logical and operational semantics
Author	Your lawyer	Your programmer	Your DBA
Performance tracking	Humint	Distributed ledger	Distributed ledger
Guarantees	What your lawyer says	What your programmer says	What your specification says
Violations	When your lawyer says	When your programmer says	When your spec checker says

Contracts, smart and otherwise

Contracts	**Analog**	**Digital**	
	Traditional	*Smart*	*Deixis*
Syntax	Late Renaissance English with Latin neologisms	Yet another OO language	Domain specific language for specifying behavior only
Semantics	What your lawyer says	What your programmer says	Mathematically defined categorical, logical and operational semantics
Author	Your lawyer	Your programmer	Your DBA
Performance tracking	Humint	Distributed ledger	Distributed ledger
Guarantees	What your lawyer says	What your programmer says	What your specification says
Violations	When your lawyer says	When your programmer says	When your spec checker says

Contracts, smart and otherwise

Contracts	**Analog**	**Digital**	
	Traditional	*Smart*	*Deixis*
Syntax	Late Renaissance English with Latin neologisms	Yet another OO language	Domain specific language for specifying behavior only
Semantics	What your lawyer says	What your programmer says	Mathematically defined categorical, logical and operational semantics
Author	Your lawyer	Your programmer	Your DBA
Performance tracking	Humint	Distributed ledger	Distributed ledger
Guarantees	What your lawyer says	What your programmer says	What your specification says
Violations	When your lawyer says	When your programmer says	When your spec checker says

Contracts, smart and otherwise

Contracts	**Analog**	**Digital**	
	Traditional	*Smart*	*Deixis*
Syntax	Late Renaissance English with Latin neologisms	Yet another OO language	Domain specific language for specifying behavior only
Semantics	What your lawyer says	What your programmer says	Mathematically defined categorical, logical and operational semantics
Author	Your lawyer	Your programmer	Your DBA
Performance tracking	Humint	Distributed ledger	Distributed ledger
Guarantees	What your lawyer says	What your programmer says	What your specification says
Violations	When your lawyer says	When your programmer says	When your spec checker says

Contracts, smart and otherwise

Contracts	**Analog**	**Digital**	
	Traditional	*Smart*	*Deixis*
Syntax	Late Renaissance English with Latin neologisms	Yet another OO language	Domain specific language for specifying behavior only
Semantics	What your lawyer says	What your programmer says	Mathematically defined categorical, logical and operational semantics
Author	Your lawyer	Your programmer	Your DBA
Performance tracking	Humint	Distributed ledger	Distributed ledger
Guarantees	What your lawyer says	What your programmer says	What your specification says
Violations	When your lawyer says	When your programmer says	When your spec checker says

In this section:

Alice's behavior as algebra

We present Alice's behavior as an aglebraic system of equations of the same sort that gives rise to regular expressions with an associated interpretation as a labeled transitions system (LTS)

$$\begin{aligned}
\texttt{Alice} &\stackrel{\text{def}}{=} \overline{\texttt{dose}}.\texttt{Alice1} \\
\texttt{Alice1} &\stackrel{\text{def}}{=} \texttt{ackdose}.\texttt{Alice2} \\
\texttt{Alice2} &\stackrel{\text{def}}{=} \overline{\texttt{bill}}.\texttt{Alice3} \\
\texttt{Alice3} &\stackrel{\text{def}}{=} \texttt{pay}.\texttt{Alice}
\end{aligned}$$

Alice's behavior as algebra

We present Alice's behavior as an aglebraic system of equations of the same sort that gives rise to regular expressions with an associated interpretation as a labeled transitions system (LTS)

$$\mathtt{Alice} \stackrel{\text{def}}{=} \overline{\mathtt{dose}}.\mathtt{Alice1}$$

$$\mathtt{Alice1} \stackrel{\text{def}}{=} \mathtt{ackdose}.\mathtt{Alice2}$$

$$\mathtt{Alice2} \stackrel{\text{def}}{=} \overline{\mathtt{bill}}.\mathtt{Alice3}$$

$$\mathtt{Alice3} \stackrel{\text{def}}{=} \mathtt{pay}.\mathtt{Alice}$$

Alice's behavior as algebra

We present Alice's behavior as an aglebraic system of equations of the same sort that gives rise to regular expressions with an associated interpretation as a labeled transitions system (LTS)

$$\texttt{Alice} \stackrel{\text{def}}{=} \overline{\texttt{dose}}.\texttt{Alice1}$$

$$\texttt{Alice1} \stackrel{\text{def}}{=} \texttt{ackdose}.\texttt{Alice2}$$

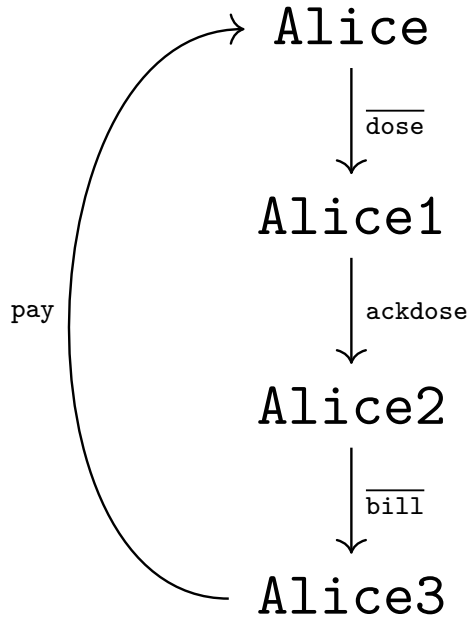
$$\texttt{Alice2} \stackrel{\text{def}}{=} \overline{\texttt{bill}}.\texttt{Alice3}$$

$$\texttt{Alice3} \stackrel{\text{def}}{=} \texttt{pay}.\texttt{Alice}$$

Alice's behavior as algebra

We present Alice's behavior as an aglebraic system of equations of the same sort that gives rise to regular expressions with an associated interpretation as a labeled transitions system (LTS)

$$\begin{aligned}
\texttt{Alice} &\stackrel{\text{def}}{=} \overline{\texttt{dose}}.\texttt{Alice1} \\
\texttt{Alice1} &\stackrel{\text{def}}{=} \texttt{ackdose}.\texttt{Alice2} \\
\texttt{Alice2} &\stackrel{\text{def}}{=} \overline{\texttt{bill}}.\texttt{Alice3} \\
\texttt{Alice3} &\stackrel{\text{def}}{=} \texttt{pay}.\texttt{Alice}
\end{aligned}$$



Carol's behavior as algebra

$$\texttt{Carol} \stackrel{\text{def}}{=} \texttt{bill}.\texttt{Carol1} + \texttt{rcvdose}.\texttt{Carol2}$$

$$\texttt{Carol1} \stackrel{\text{def}}{=} \texttt{rcvdose}.\texttt{Carol3}$$

$$\texttt{Carol2} \stackrel{\text{def}}{=} \texttt{bill}.\texttt{Carol3}$$

$$\texttt{Carol3} \stackrel{\text{def}}{=} \overline{\texttt{pay}}.\texttt{Carol}$$

Carol's behavior as algebra

$$\texttt{Carol} \stackrel{\text{def}}{=} \texttt{bill.Carol1} + \texttt{rcvdose.Carol2}$$

$$\texttt{Carol1} \stackrel{\text{def}}{=} \texttt{rcvdose.Carol3}$$

$$\texttt{Carol2} \stackrel{\text{def}}{=} \texttt{bill.Carol3}$$

$$\texttt{Carol3} \stackrel{\text{def}}{=} \overline{\texttt{pay}}\texttt{.Carol}$$

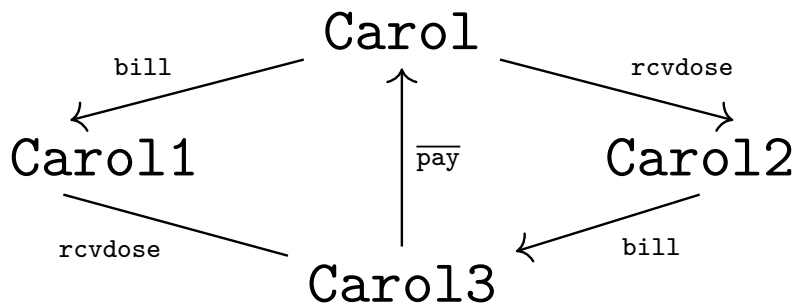
Carol's behavior as algebra

$$\texttt{Carol} \stackrel{\text{def}}{=} \texttt{bill.Carol1} + \texttt{rcvdose.Carol2}$$

$$\texttt{Carol1} \stackrel{\text{def}}{=} \texttt{rcvdose.Carol3}$$

$$\texttt{Carol2} \stackrel{\text{def}}{=} \texttt{bill.Carol3}$$

$$\texttt{Carol3} \stackrel{\text{def}}{=} \overline{\texttt{pay}}\texttt{.Carol}$$



Ted's behavior as algebra

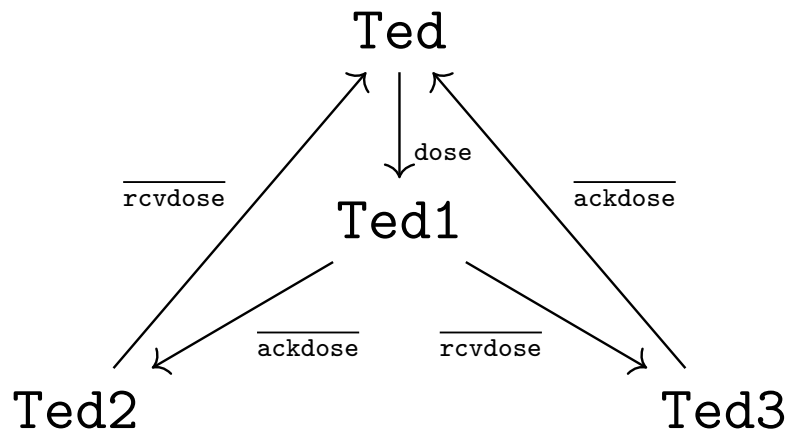
$$\begin{aligned}
\mathtt{Ted} &\stackrel{\mathrm{def}}{=} \mathtt{dose}.\mathtt{Ted1} \\
\mathtt{Ted1} &\stackrel{\mathrm{def}}{=} \overline{\mathtt{ackdose}}.\mathtt{Ted2} + \\
&\qquad \overline{\mathtt{rcvdose}}.\mathtt{Ted3} \\
\mathtt{Ted2} &\stackrel{\mathrm{def}}{=} \overline{\mathtt{rcvdose}}.\mathtt{Ted} \\
\mathtt{Ted3} &\stackrel{\mathrm{def}}{=} \overline{\mathtt{ackdose}}.\mathtt{Ted}
\end{aligned}$$

Ted's behavior as algebra

$$\begin{aligned}
\texttt{Ted} &\stackrel{\texttt{def}}{=} \texttt{dose.Ted1} \\
\texttt{Ted1} &\stackrel{\texttt{def}}{=} \overline{\texttt{ackdose}}\texttt{.Ted2} + \\
&\qquad \overline{\texttt{rcvdose}}\texttt{.Ted3} \\
\texttt{Ted2} &\stackrel{\texttt{def}}{=} \overline{\texttt{rcvdose}}\texttt{.Ted} \\
\texttt{Ted3} &\stackrel{\texttt{def}}{=} \overline{\texttt{ackdose}}\texttt{.Ted}
\end{aligned}$$

Ted's behavior as algebra

$$\begin{aligned}
\mathtt{Ted} &\stackrel{\mathtt{def}}{=} \mathtt{dose.Ted1} \\
\mathtt{Ted1} &\stackrel{\mathtt{def}}{=} \overline{\mathtt{ackdose}}\mathtt{.Ted2} + \\
&\qquad \overline{\mathtt{rcvdose}}\mathtt{.Ted3} \\
\mathtt{Ted2} &\stackrel{\mathtt{def}}{=} \overline{\mathtt{rcvdose}}\mathtt{.Ted} \\
\mathtt{Ted3} &\stackrel{\mathtt{def}}{=} \overline{\mathtt{ackdose}}\mathtt{.Ted}
\end{aligned}$$



Alice | Ted joint behavior

Thus far we have considered the behaviors of each of Alice, Carol and Ted as 1-body problems

We may extend behavior to 2-body problems and beyond by demanding that an LTS be "closed" under certain rules, such as

$$\frac{x \xrightarrow{\overline{a}} x' \qquad y \xrightarrow{a} y'}{x \mid y \xrightarrow{\tau} x' \mid y'}$$

which (when instantiated) allow Alice and Ted to *interact* via

$$\frac{\texttt{Alice} \xrightarrow{\overline{\texttt{dose}}} \texttt{Alice1} \qquad \texttt{Ted} \xrightarrow{\texttt{dose}} \texttt{Ted1}}{\texttt{Alice} \mid \texttt{Ted} \xrightarrow{\tau} \texttt{Alice1} \mid \texttt{Ted1}}$$

and via

$$\frac{\texttt{Ted1} \xrightarrow{\overline{\texttt{ackdose}}} \texttt{Ted2} \qquad \texttt{Alice1} \xrightarrow{\texttt{ackdose}} \texttt{Alice2}}{\texttt{Ted1} \mid \texttt{Alice1} \xrightarrow{\tau} \texttt{Ted2} \mid \texttt{Alice2}}$$

Alice | Ted joint behavior

Thus far we have considered the behaviors of each of Alice, Carol and Ted as 1-body problems

We may extend behavior to 2-body problems and beyond by demanding that an LTS be "closed" under certain rules, such as

$$\frac{x \xrightarrow{\overline{a}} x' \qquad y \xrightarrow{a} y'}{x \mid y \xrightarrow{\tau} x' \mid y'}$$

which (when instantiated) allow Alice and Ted to *interact* via

$$\frac{\texttt{Alice} \xrightarrow{\overline{\texttt{dose}}} \texttt{Alice1} \qquad \texttt{Ted} \xrightarrow{\texttt{dose}} \texttt{Ted1}}{\texttt{Alice} \mid \texttt{Ted} \xrightarrow{\tau} \texttt{Alice1} \mid \texttt{Ted1}}$$

and via

$$\frac{\texttt{Ted1} \xrightarrow{\overline{\texttt{ackdose}}} \texttt{Ted2} \qquad \texttt{Alice1} \xrightarrow{\texttt{ackdose}} \texttt{Alice2}}{\texttt{Ted1} \mid \texttt{Alice1} \xrightarrow{\tau} \texttt{Ted2} \mid \texttt{Alice2}}$$

Alice | Ted joint behavior

Thus far we have considered the behaviors of each of Alice, Carol and Ted as 1-body problems

We may extend behavior to 2-body problems and beyond by demanding that an LTS be "closed" under certain rules, such as

$$\frac{x \xrightarrow{\overline{a}} x' \qquad y \xrightarrow{a} y'}{x \mid y \xrightarrow{\tau} x' \mid y'}$$

which (when instantiated) allow Alice and Ted to *interact* via

$$\frac{\texttt{Alice} \xrightarrow{\overline{\texttt{dose}}} \texttt{Alice1} \qquad \texttt{Ted} \xrightarrow{\texttt{dose}} \texttt{Ted1}}{\texttt{Alice} \mid \texttt{Ted} \xrightarrow{\tau} \texttt{Alice1} \mid \texttt{Ted1}}$$

and via

$$\frac{\texttt{Ted1} \xrightarrow{\overline{\texttt{ackdose}}} \texttt{Ted2} \qquad \texttt{Alice1} \xrightarrow{\texttt{ackdose}} \texttt{Alice2}}{\texttt{Ted1} \mid \texttt{Alice1} \xrightarrow{\tau} \texttt{Ted2} \mid \texttt{Alice2}}$$

Alice | Ted joint behavior

Thus far we have considered the behaviors of each of Alice, Carol and Ted as 1-body problems

We may extend behavior to 2-body problems and beyond by demanding that an LTS be "closed" under certain rules, such as

$$\frac{x \xrightarrow{\overline{a}} x' \qquad y \xrightarrow{a} y'}{x \mid y \xrightarrow{\tau} x' \mid y'}$$

which (when instantiated) allow Alice and Ted to *interact* via

$$\frac{\texttt{Alice} \xrightarrow{\overline{\texttt{dose}}} \texttt{Alice1} \qquad \texttt{Ted} \xrightarrow{\texttt{dose}} \texttt{Ted1}}{\texttt{Alice} \mid \texttt{Ted} \xrightarrow{\tau} \texttt{Alice1} \mid \texttt{Ted1}}$$

and via

$$\frac{\texttt{Ted1} \xrightarrow{\overline{\texttt{ackdose}}} \texttt{Ted2} \qquad \texttt{Alice1} \xrightarrow{\texttt{ackdose}} \texttt{Alice2}}{\texttt{Ted1} \mid \texttt{Alice1} \xrightarrow{\tau} \texttt{Ted2} \mid \texttt{Alice2}}$$

Alice | Ted joint behavior

Thus far we have considered the behaviors of each of Alice, Carol and Ted as 1-body problems

We may extend behavior to 2-body problems and beyond by demanding that an LTS be "closed" under certain rules, such as

$$\frac{x \xrightarrow{\overline{a}} x' \qquad y \xrightarrow{a} y'}{x \mid y \xrightarrow{\tau} x' \mid y'}$$

which (when instantiated) allow Alice and Ted to *interact* via

$$\frac{\texttt{Alice} \xrightarrow{\overline{\texttt{dose}}} \texttt{Alice1} \qquad \texttt{Ted} \xrightarrow{\texttt{dose}} \texttt{Ted1}}{\texttt{Alice} \mid \texttt{Ted} \xrightarrow{\tau} \texttt{Alice1} \mid \texttt{Ted1}}$$

and via

$$\frac{\texttt{Ted1} \xrightarrow{\overline{\texttt{ackdose}}} \texttt{Ted2} \qquad \texttt{Alice1} \xrightarrow{\texttt{ackdose}} \texttt{Alice2}}{\texttt{Ted1} \mid \texttt{Alice1} \xrightarrow{\tau} \texttt{Ted2} \mid \texttt{Alice2}}$$

Alice | Ted joint behavior

Thus far we have considered the behaviors of each of Alice, Carol and Ted as 1-body problems

We may extend behavior to 2-body problems and beyond by demanding that an LTS be "closed" under certain rules, such as

$$\frac{x \xrightarrow{\overline{a}} x' \qquad y \xrightarrow{a} y'}{x \mid y \xrightarrow{\tau} x' \mid y'}$$

which (when instantiated) allow Alice and Ted to *interact* via

$$\frac{\texttt{Alice} \xrightarrow{\overline{\texttt{dose}}} \texttt{Alice1} \qquad \texttt{Ted} \xrightarrow{\texttt{dose}} \texttt{Ted1}}{\texttt{Alice} \mid \texttt{Ted} \xrightarrow{\tau} \texttt{Alice1} \mid \texttt{Ted1}}$$

and via

$$\frac{\texttt{Ted1} \xrightarrow{\overline{\texttt{ackdose}}} \texttt{Ted2} \qquad \texttt{Alice1} \xrightarrow{\texttt{ackdose}} \texttt{Alice2}}{\texttt{Ted1} \mid \texttt{Alice1} \xrightarrow{\tau} \texttt{Ted2} \mid \texttt{Alice2}}$$

Alice | Ted joint behavior

Thus far we have considered the behaviors of each of Alice, Carol and Ted as 1-body problems

We may extend behavior to 2-body problems and beyond by demanding that an LTS be "closed" under certain rules, such as

$$\frac{x \xrightarrow{\overline{a}} x' \qquad y \xrightarrow{a} y'}{x \mid y \xrightarrow{\tau} x' \mid y'}$$

which (when instantiated) allow Alice and Ted to *interact* via

$$\frac{\texttt{Alice} \xrightarrow{\overline{\texttt{dose}}} \texttt{Alice1} \qquad \texttt{Ted} \xrightarrow{\texttt{dose}} \texttt{Ted1}}{\texttt{Alice} \mid \texttt{Ted} \xrightarrow{\tau} \texttt{Alice1} \mid \texttt{Ted1}}$$

and via

$$\frac{\texttt{Ted1} \xrightarrow{\overline{\texttt{ackdose}}} \texttt{Ted2} \qquad \texttt{Alice1} \xrightarrow{\texttt{ackdose}} \texttt{Alice2}}{\texttt{Ted1} \mid \texttt{Alice1} \xrightarrow{\tau} \texttt{Ted2} \mid \texttt{Alice2}}$$

Alice | Ted joint behavior

Thus far we have considered the behaviors of each of Alice, Carol and Ted as 1-body problems

We may extend behavior to 2-body problems and beyond by demanding that an LTS be "closed" under certain rules, such as

$$\frac{x \xrightarrow{\overline{a}} x' \qquad y \xrightarrow{a} y'}{x \mid y \xrightarrow{\tau} x' \mid y'}$$

which (when instantiated) allow Alice and Ted to *interact* via

$$\frac{\texttt{Alice} \xrightarrow{\overline{\texttt{dose}}} \texttt{Alice1} \qquad \texttt{Ted} \xrightarrow{\texttt{dose}} \texttt{Ted1}}{\texttt{Alice} \mid \texttt{Ted} \xrightarrow{\tau} \texttt{Alice1} \mid \texttt{Ted1}}$$

and via

$$\frac{\texttt{Ted1} \xrightarrow{\overline{\texttt{ackdose}}} \texttt{Ted2} \qquad \texttt{Alice1} \xrightarrow{\texttt{ackdose}} \texttt{Alice2}}{\texttt{Ted1} \mid \texttt{Alice1} \xrightarrow{\tau} \texttt{Ted2} \mid \texttt{Alice2}}$$

Mechanization (contracts as software)

The contract to which Alice, Carol and Ted are parties may be specified by a system of algebraic equations in the *Calculus of Concurrent Systems* (CCS), the "solution" to this system being the previous LTS

$$\texttt{ACT1} \stackrel{\text{def}}{=} \texttt{Alice} \mid \texttt{Carol} \mid \texttt{Ted}$$

$$\texttt{Alice} \stackrel{\text{def}}{=} \overline{\texttt{dose}}.\texttt{Alice1}$$

$$\texttt{Alice1} \stackrel{\text{def}}{=} \texttt{ackdose}.\texttt{Alice2}$$

$$\texttt{Alice2} \stackrel{\text{def}}{=} \overline{\texttt{bill}}.\texttt{Alice3}$$

$$\texttt{Alice3} \stackrel{\text{def}}{=} \texttt{pay}.\texttt{Alice}$$

$$\texttt{Carol} \stackrel{\text{def}}{=} \texttt{bill}.\texttt{Carol1} + \texttt{rcvdose}.\texttt{Carol2}$$

$$\texttt{Carol1} \stackrel{\text{def}}{=} \texttt{rcvdose}.\texttt{Carol3}$$

$$\texttt{Carol2} \stackrel{\text{def}}{=} \texttt{bill}.\texttt{Carol3}$$

$$\texttt{Carol3} \stackrel{\text{def}}{=} \overline{\texttt{pay}}.\texttt{Carol}$$

$$\texttt{Ted} \stackrel{\text{def}}{=} \texttt{dose}.\texttt{Ted1}$$

$$\texttt{Ted1} \stackrel{\text{def}}{=} \overline{\texttt{ackdose}}.\texttt{Ted2} + \overline{\texttt{rcvdose}}.\texttt{Ted3}$$

$$\texttt{Ted2} \stackrel{\text{def}}{=} \overline{\texttt{rcvdose}}.\texttt{Ted}$$

$$\texttt{Ted3} \stackrel{\text{def}}{=} \overline{\texttt{ackdose}}.\texttt{Ted}$$

Mechanization (contracts as software)

The contract to which Alice, Carol and Ted are parties may be specified by a system of algebraic equations in the *Calculus of Concurrent Systems* (CCS), the "solution" to this system being the previous LTS

$$\mathtt{ACT1} \stackrel{\text{def}}{=} \mathtt{Alice} \mid \mathtt{Carol} \mid \mathtt{Ted}$$

$$\mathtt{Alice} \stackrel{\text{def}}{=} \overline{\mathtt{dose}}.\mathtt{Alice1}$$

$$\mathtt{Alice1} \stackrel{\text{def}}{=} \mathtt{ackdose}.\mathtt{Alice2}$$

$$\mathtt{Alice2} \stackrel{\text{def}}{=} \overline{\mathtt{bill}}.\mathtt{Alice3}$$

$$\mathtt{Alice3} \stackrel{\text{def}}{=} \mathtt{pay}.\mathtt{Alice}$$

$$\mathtt{Carol} \stackrel{\text{def}}{=} \mathtt{bill}.\mathtt{Carol1} + \mathtt{rcvdose}.\mathtt{Carol2}$$

$$\mathtt{Carol1} \stackrel{\text{def}}{=} \mathtt{rcvdose}.\mathtt{Carol3}$$

$$\mathtt{Carol2} \stackrel{\text{def}}{=} \mathtt{bill}.\mathtt{Carol3}$$

$$\mathtt{Carol3} \stackrel{\text{def}}{=} \overline{\mathtt{pay}}.\mathtt{Carol}$$

$$\mathtt{Ted} \stackrel{\text{def}}{=} \mathtt{dose}.\mathtt{Ted1}$$

$$\mathtt{Ted1} \stackrel{\text{def}}{=} \overline{\mathtt{ackdose}}.\mathtt{Ted2} + \overline{\mathtt{rcvdose}}.\mathtt{Ted3}$$

$$\mathtt{Ted2} \stackrel{\text{def}}{=} \overline{\mathtt{rcvdose}}.\mathtt{Ted}$$

$$\mathtt{Ted3} \stackrel{\text{def}}{=} \overline{\mathtt{ackdose}}.\mathtt{Ted}$$

Mechanization (contracts as software)

The contract to which Alice, Carol and Ted are parties may be specified by a system of algebraic equations in the *Calculus of Concurrent Systems* (CCS), the "solution" to this system being the previous LTS

$$\texttt{ACT1} \stackrel{\text{def}}{=} \texttt{Alice} \mid \texttt{Carol} \mid \texttt{Ted}$$

$$\texttt{Alice} \stackrel{\text{def}}{=} \overline{\texttt{dose}}.\texttt{Alice1}$$

$$\texttt{Alice1} \stackrel{\text{def}}{=} \texttt{ackdose}.\texttt{Alice2}$$

$$\texttt{Alice2} \stackrel{\text{def}}{=} \overline{\texttt{bill}}.\texttt{Alice3}$$

$$\texttt{Alice3} \stackrel{\text{def}}{=} \texttt{pay}.\texttt{Alice}$$

$$\texttt{Carol} \stackrel{\text{def}}{=} \texttt{bill}.\texttt{Carol1} + \texttt{rcvdose}.\texttt{Carol2}$$

$$\texttt{Carol1} \stackrel{\text{def}}{=} \texttt{rcvdose}.\texttt{Carol3}$$

$$\texttt{Carol2} \stackrel{\text{def}}{=} \texttt{bill}.\texttt{Carol3}$$

$$\texttt{Carol3} \stackrel{\text{def}}{=} \overline{\texttt{pay}}.\texttt{Carol}$$

$$\texttt{Ted} \stackrel{\text{def}}{=} \texttt{dose}.\texttt{Ted1}$$

$$\texttt{Ted1} \stackrel{\text{def}}{=} \overline{\texttt{ackdose}}.\texttt{Ted2} + \overline{\texttt{rcvdose}}.\texttt{Ted3}$$

$$\texttt{Ted2} \stackrel{\text{def}}{=} \overline{\texttt{rcvdose}}.\texttt{Ted}$$

$$\texttt{Ted3} \stackrel{\text{def}}{=} \overline{\texttt{ackdose}}.\texttt{Ted}$$

Mechanization (contracts as software)

The contract to which Alice, Carol and Ted are parties may be specified by a system of algebraic equations in the *Calculus of Concurrent Systems* (CCS), the "solution" to this system being the previous LTS

$$\texttt{ACT1} \stackrel{\text{def}}{=} \texttt{Alice} \mid \texttt{Carol} \mid \texttt{Ted}$$

$$\texttt{Alice} \stackrel{\text{def}}{=} \overline{\texttt{dose}}.\texttt{Alice1}$$

$$\texttt{Alice1} \stackrel{\text{def}}{=} \texttt{ackdose}.\texttt{Alice2}$$

$$\texttt{Alice2} \stackrel{\text{def}}{=} \overline{\texttt{bill}}.\texttt{Alice3}$$

$$\texttt{Alice3} \stackrel{\text{def}}{=} \texttt{pay}.\texttt{Alice}$$

$$\texttt{Carol} \stackrel{\text{def}}{=} \texttt{bill}.\texttt{Carol1} + \texttt{rcvdose}.\texttt{Carol2}$$

$$\texttt{Carol1} \stackrel{\text{def}}{=} \texttt{rcvdose}.\texttt{Carol3}$$

$$\texttt{Carol2} \stackrel{\text{def}}{=} \texttt{bill}.\texttt{Carol3}$$

$$\texttt{Carol3} \stackrel{\text{def}}{=} \overline{\texttt{pay}}.\texttt{Carol}$$

$$\texttt{Ted} \stackrel{\text{def}}{=} \texttt{dose}.\texttt{Ted1}$$

$$\texttt{Ted1} \stackrel{\text{def}}{=} \overline{\texttt{ackdose}}.\texttt{Ted2} + \overline{\texttt{rcvdose}}.\texttt{Ted3}$$

$$\texttt{Ted2} \stackrel{\text{def}}{=} \overline{\texttt{rcvdose}}.\texttt{Ted}$$

$$\texttt{Ted3} \stackrel{\text{def}}{=} \overline{\texttt{ackdose}}.\texttt{Ted}$$

ACT1

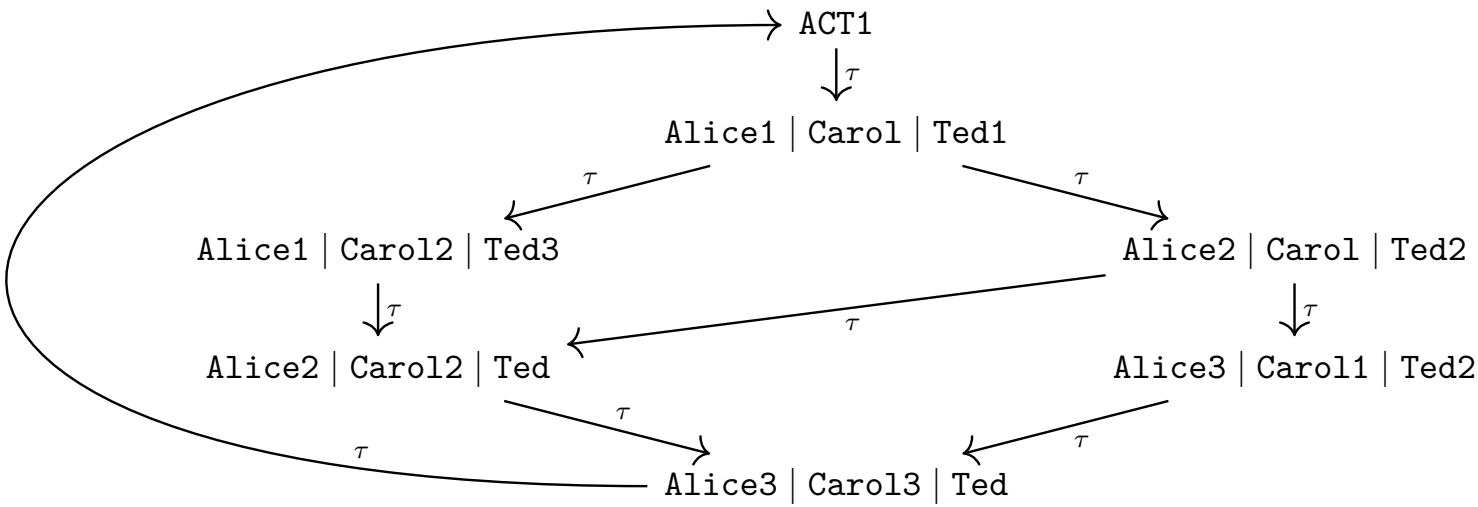
$$\texttt{ACT1} \stackrel{\texttt{def}}{=} \texttt{Alice} \mid \texttt{Carol} \mid \texttt{Ted}$$

ACT1

$$\texttt{ACT1} \stackrel{\texttt{def}}{=} \texttt{Alice} \mid \texttt{Carol} \mid \texttt{Ted}$$

ACT1

$$\texttt{ACT1} \stackrel{\texttt{def}}{=} \texttt{Alice} \mid \texttt{Carol} \mid \texttt{Ted}$$



In this section:

Why distributed ledgers?

- As we have just illustrated, contracts can be described as a dynamical system
  - whose state is constantly changing
  - involving a number of stakeholders
  - some of whom may be actors in the system
  - for which there may be a shared, explicit specification – AKA a contract
- There is a high degree of desire among the stakeholders for a shared chronicle – a kind of trace – of an evolving system state
- Such a chronicle constitutes a shared database
- Considerations of both security and administrative complexity render each of the stakeholders averse to hosting a shared database *system*

Note that the poster child application for a distributed ledger is a cryptocurrency such as Bitcoin

Why distributed ledgers?

- As we have just illustrated, contracts can be described as a dynamical system
  - whose state is constantly changing
  - involving a number of stakeholders
  - some of whom may be actors in the system
  - for which there may be a shared, explicit specification – AKA a contract
- There is a high degree of desire among the stakeholders for a shared chronicle – a kind of trace – of an evolving system state
- Such a chronicle constitutes a shared database
- Considerations of both security and administrative complexity render each of the stakeholders averse to hosting a shared database *system*

Note that the poster child application for a distributed ledger is a cryptocurrency such as Bitcoin

Why distributed ledgers?

- As we have just illustrated, contracts can be described as a dynamical system
  - whose state is constantly changing
  - involving a number of stakeholders
  - some of whom may be actors in the system
  - for which there may be a shared, explicit specification – AKA a contract
- There is a high degree of desire among the stakeholders for a shared chronicle – a kind of trace – of an evolving system state
- Such a chronicle constitutes a shared database
- Considerations of both security and administrative complexity render each of the stakeholders averse to hosting a shared database *system*

Note that the poster child application for a distributed ledger is a cryptocurrency such as Bitcoin

Why distributed ledgers?

- As we have just illustrated, contracts can be described as a dynamical system
  - whose state is constantly changing
  - involving a number of stakeholders
  - some of whom may be actors in the system
  - for which there may be a shared, explicit specification – AKA a contract
- There is a high degree of desire among the stakeholders for a shared chronicle – a kind of trace – of an evolving system state
- Such a chronicle constitutes a shared database
- Considerations of both security and administrative complexity render each of the stakeholders averse to hosting a shared database *system*

Note that the poster child application for a distributed ledger is a cryptocurrency such as Bitcoin

Why distributed ledgers?

- As we have just illustrated, contracts can be described as a dynamical system
  - whose state is constantly changing
  - involving a number of stakeholders
  - some of whom may be actors in the system
  - for which there may be a shared, explicit specification – AKA a contract
- There is a high degree of desire among the stakeholders for a shared chronicle – a kind of trace – of an evolving system state
- Such a chronicle constitutes a shared database
- Considerations of both security and administrative complexity render each of the stakeholders averse to hosting a shared database *system*

Note that the poster child application for a distributed ledger is a cryptocurrency such as Bitcoin

Why distributed ledgers?

- As we have just illustrated, contracts can be described as a dynamical system
  - whose state is constantly changing
  - involving a number of stakeholders
  - some of whom may be actors in the system
  - for which there may be a shared, explicit specification – AKA a contract
- There is a high degree of desire among the stakeholders for a shared chronicle – a kind of trace – of an evolving system state
- Such a chronicle constitutes a shared database
- Considerations of both security and administrative complexity render each of the stakeholders averse to hosting a shared database *system*

Note that the poster child application for a distributed ledger is a cryptocurrency such as Bitcoin

Why distributed ledgers?

- As we have just illustrated, contracts can be described as a dynamical system
  - whose state is constantly changing
  - involving a number of stakeholders
  - some of whom may be actors in the system
  - for which there may be a shared, explicit specification – AKA a contract
- There is a high degree of desire among the stakeholders for a shared chronicle – a kind of trace – of an evolving system state
- Such a chronicle constitutes a shared database
- Considerations of both security and administrative complexity render each of the stakeholders averse to hosting a shared database *system*

Note that the poster child application for a distributed ledger is a cryptocurrency such as Bitcoin

Why distributed ledgers?

- As we have just illustrated, contracts can be described as a dynamical system
  - whose state is constantly changing
  - involving a number of stakeholders
  - some of whom may be actors in the system
  - for which there may be a shared, explicit specification – AKA a contract
- There is a high degree of desire among the stakeholders for a shared chronicle – a kind of trace – of an evolving system state
- Such a chronicle constitutes a shared database
- Considerations of both security and administrative complexity render each of the stakeholders averse to hosting a shared database *system*

Note that the poster child application for a distributed ledger is a cryptocurrency such as Bitcoin

Why distributed ledgers?

- As we have just illustrated, contracts can be described as a dynamical system
  - whose state is constantly changing
  - involving a number of stakeholders
  - some of whom may be actors in the system
  - for which there may be a shared, explicit specification – AKA a contract
- There is a high degree of desire among the stakeholders for a shared chronicle – a kind of trace – of an evolving system state
- Such a chronicle constitutes a shared database
- Considerations of both security and administrative complexity render each of the stakeholders averse to hosting a shared database *system*

Note that the poster child application for a distributed ledger is a cryptocurrency such as Bitcoin

Why distributed ledgers?

- As we have just illustrated, contracts can be described as a dynamical system
  - whose state is constantly changing
  - involving a number of stakeholders
  - some of whom may be actors in the system
  - for which there may be a shared, explicit specification – AKA a contract
- There is a high degree of desire among the stakeholders for a shared chronicle – a kind of trace – of an evolving system state
- Such a chronicle constitutes a shared database
- Considerations of both security and administrative complexity render each of the stakeholders averse to hosting a shared database *system*

Note that the poster child application for a distributed ledger is a cryptocurrency such as Bitcoin

Executions of contracts

- An LTS, being a species of mathematical graph, naturally, has paths through it
- A path in the LTS defined by `ACT1`, such as

$$\texttt{ACT1} \xrightarrow{\tau}$$
$$\texttt{Alice1} \mid \texttt{Carol} \mid \texttt{Ted1} \xrightarrow{\tau}$$
$$\texttt{Alice2} \mid \texttt{Carol} \mid \texttt{Ted2} \xrightarrow{\tau}$$
$$\texttt{Alice3} \mid \texttt{Carol1} \mid \texttt{Ted2} \xrightarrow{\tau}$$
$$\texttt{ACT1}$$

  is called a *execution* of `ACT1`
- An execution may be understood as an alternation of behaviors and states connecting two states of the LTS

Executions of contracts

- An LTS, being a species of mathematical graph, naturally, has paths through it
- A path in the LTS defined by `ACT1`, such as

$$\texttt{ACT1} \xrightarrow{\tau}$$
$$\texttt{Alice1} \mid \texttt{Carol} \mid \texttt{Ted1} \xrightarrow{\tau}$$
$$\texttt{Alice2} \mid \texttt{Carol} \mid \texttt{Ted2} \xrightarrow{\tau}$$
$$\texttt{Alice3} \mid \texttt{Carol1} \mid \texttt{Ted2} \xrightarrow{\tau}$$
$$\texttt{ACT1}$$

  is called a *execution* of `ACT1`
- An execution may be understood as an alternation of behaviors and states connecting two states of the LTS

Executions of contracts

- An LTS, being a species of mathematical graph, naturally, has paths through it
- A path in the LTS defined by `ACT1`, such as

$$\texttt{ACT1} \xrightarrow{\tau} \texttt{Alice1} \mid \texttt{Carol} \mid \texttt{Ted1} \xrightarrow{\tau} \texttt{Alice2} \mid \texttt{Carol} \mid \texttt{Ted2} \xrightarrow{\tau} \texttt{Alice3} \mid \texttt{Carol1} \mid \texttt{Ted2} \xrightarrow{\tau} \texttt{ACT1}$$

  is called a *execution* of `ACT1`

- An execution may be understood as an alternation of behaviors and states connecting two states of the LTS

Executions of contracts

- An LTS, being a species of mathematical graph, naturally, has paths through it
- A path in the LTS defined by `ACT1`, such as

$$\texttt{ACT1} \xrightarrow{\tau} \texttt{Alice1} \mid \texttt{Carol} \mid \texttt{Ted1} \xrightarrow{\tau} \texttt{Alice2} \mid \texttt{Carol} \mid \texttt{Ted2} \xrightarrow{\tau} \texttt{Alice3} \mid \texttt{Carol1} \mid \texttt{Ted2} \xrightarrow{\tau} \texttt{ACT1}$$

  is called a *execution* of `ACT1`
- An execution may be understood as an alternation of behaviors and states connecting two states of the LTS

Distributed ledgers as carriers of contract executions

blocks

Transition records might be assembled into consensus blocks as follows

Distributed ledgers as carriers of contract executions

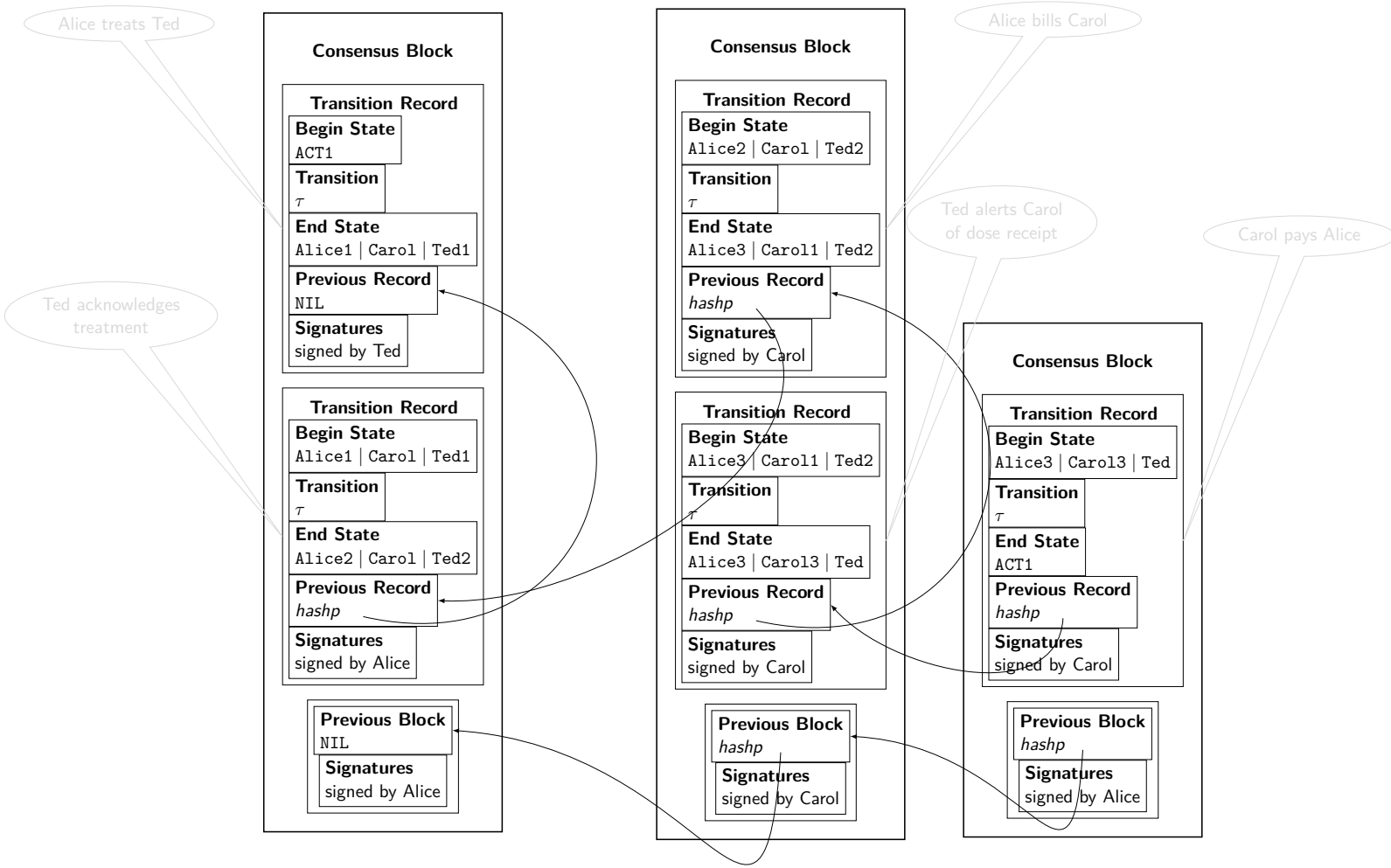
blocks

Transition records might be assembled into consensus blocks as follows

Distributed ledgers as carriers of contract executions

blocks

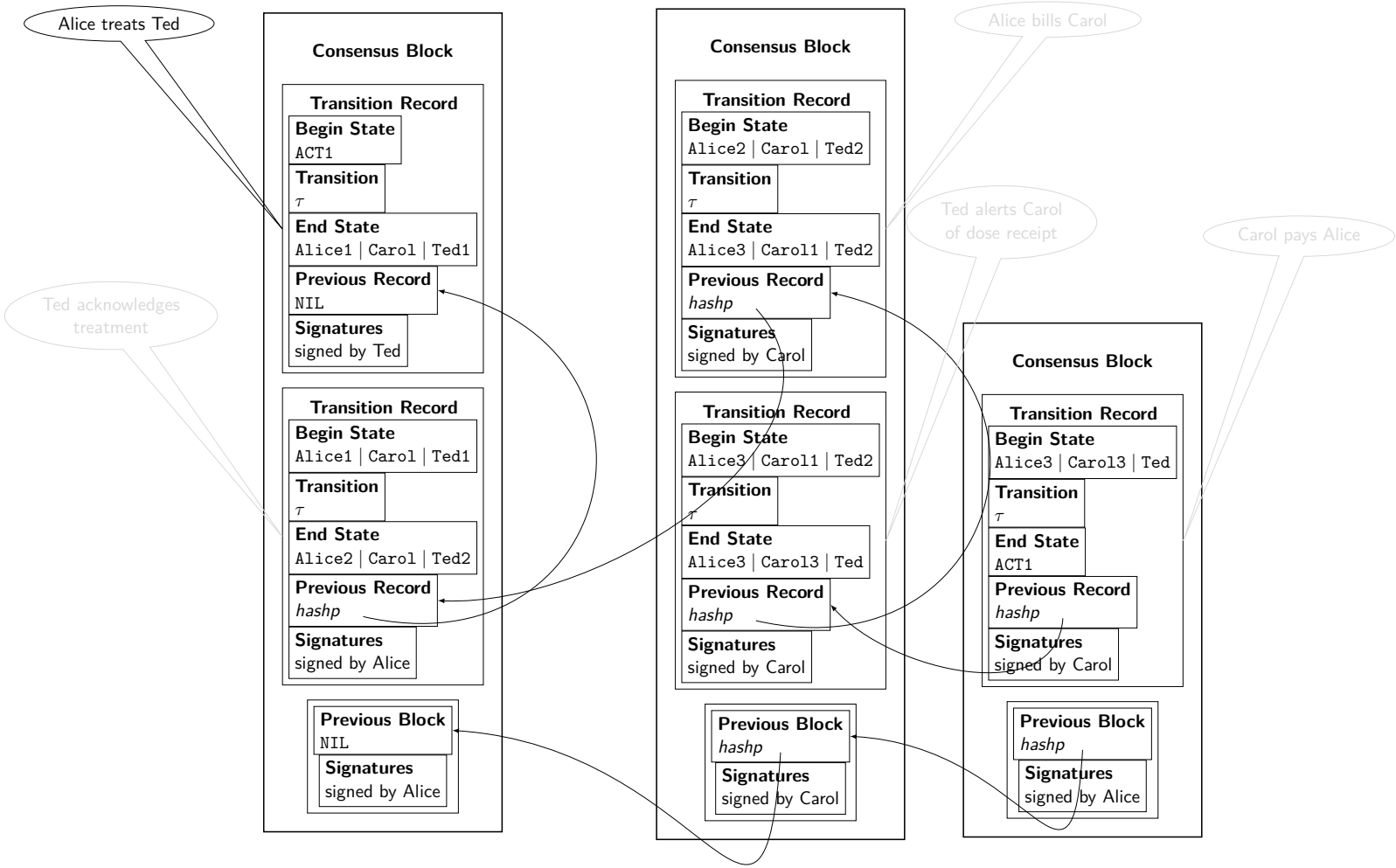
Transition records might be assembled into consensus blocks as follows



Distributed ledgers as carriers of contract executions

blocks

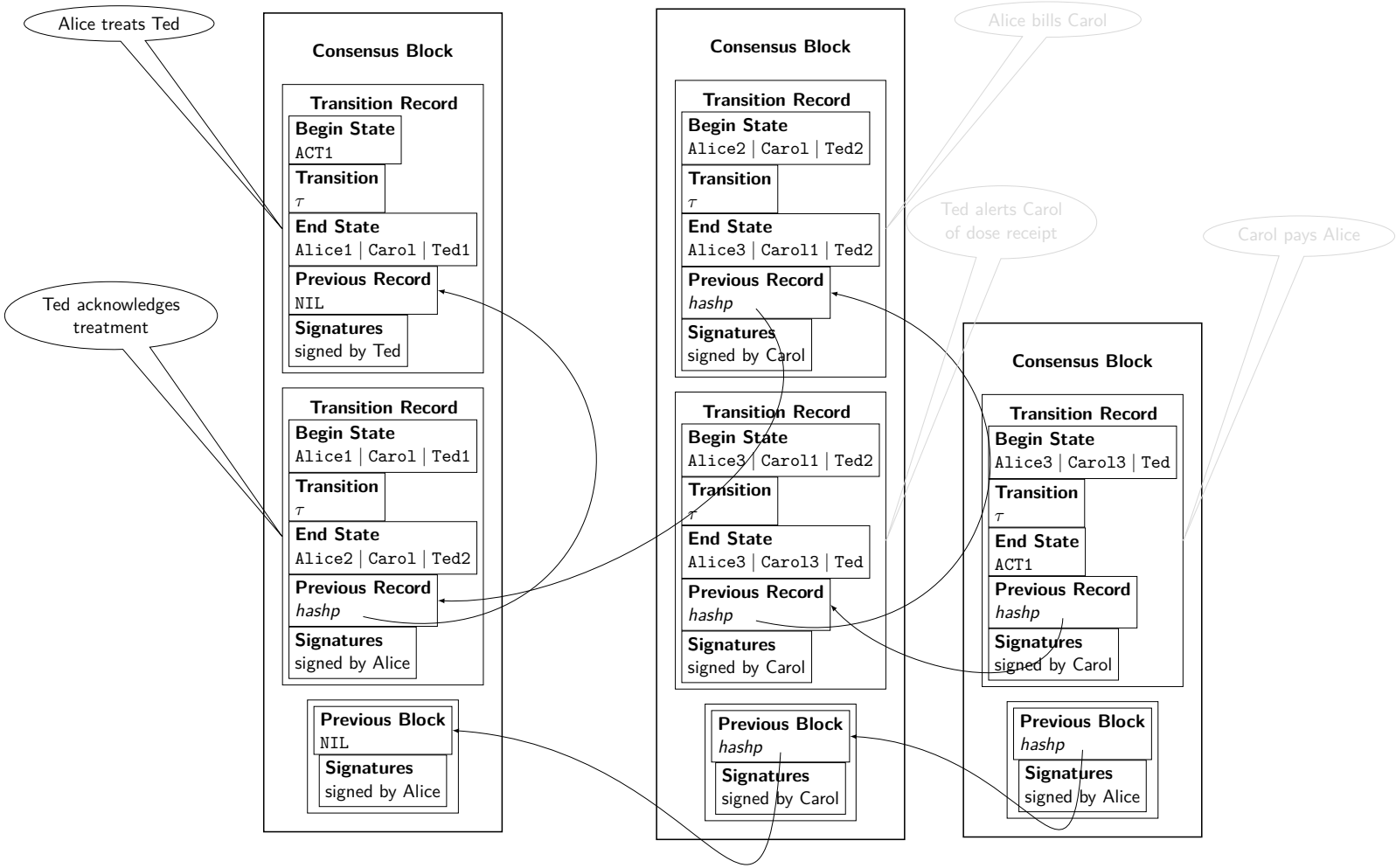
Transition records might be assembled into consensus blocks as follows



Distributed ledgers as carriers of contract executions

blocks

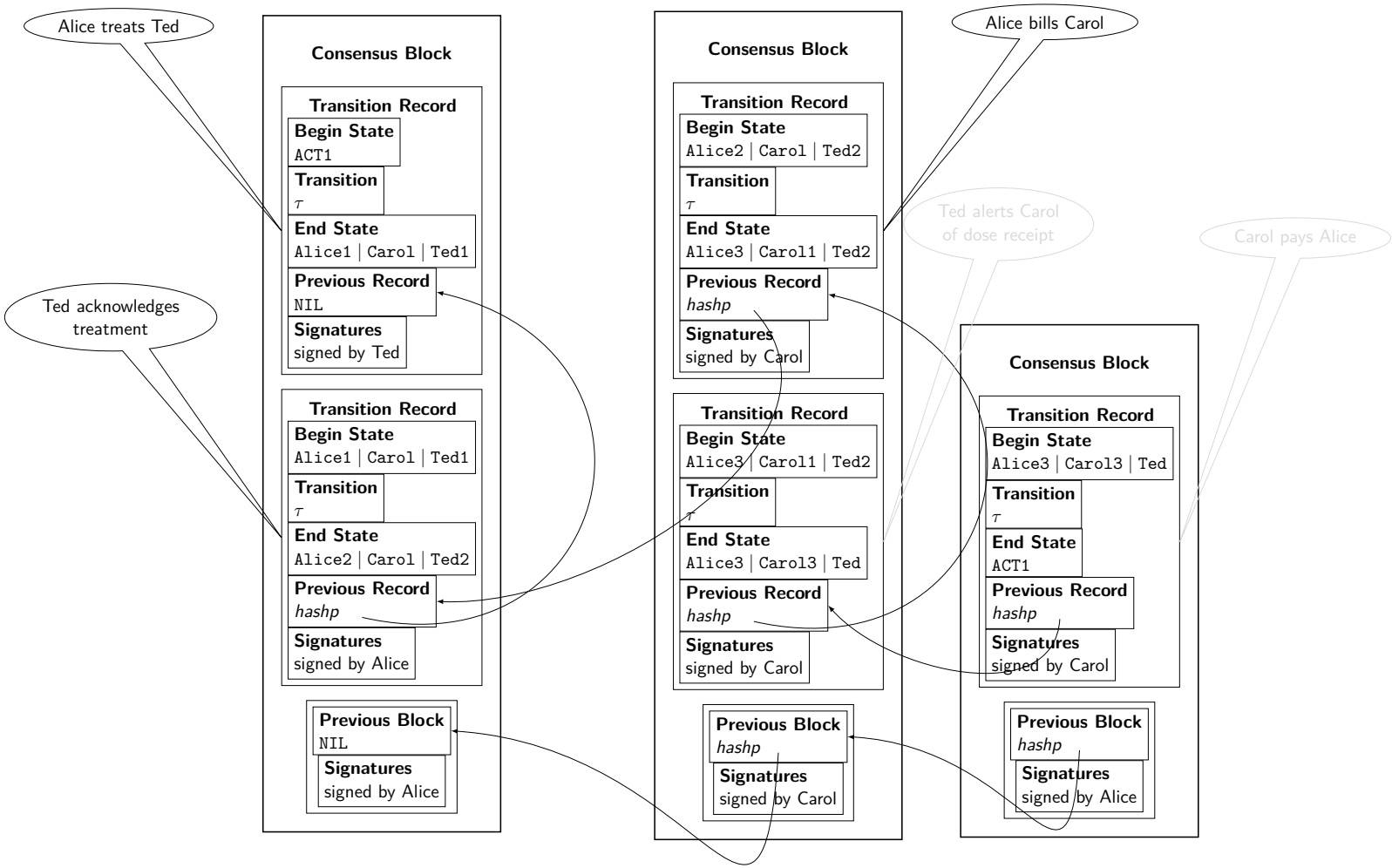
Transition records might be assembled into consensus blocks as follows



Distributed ledgers as carriers of contract executions

blocks

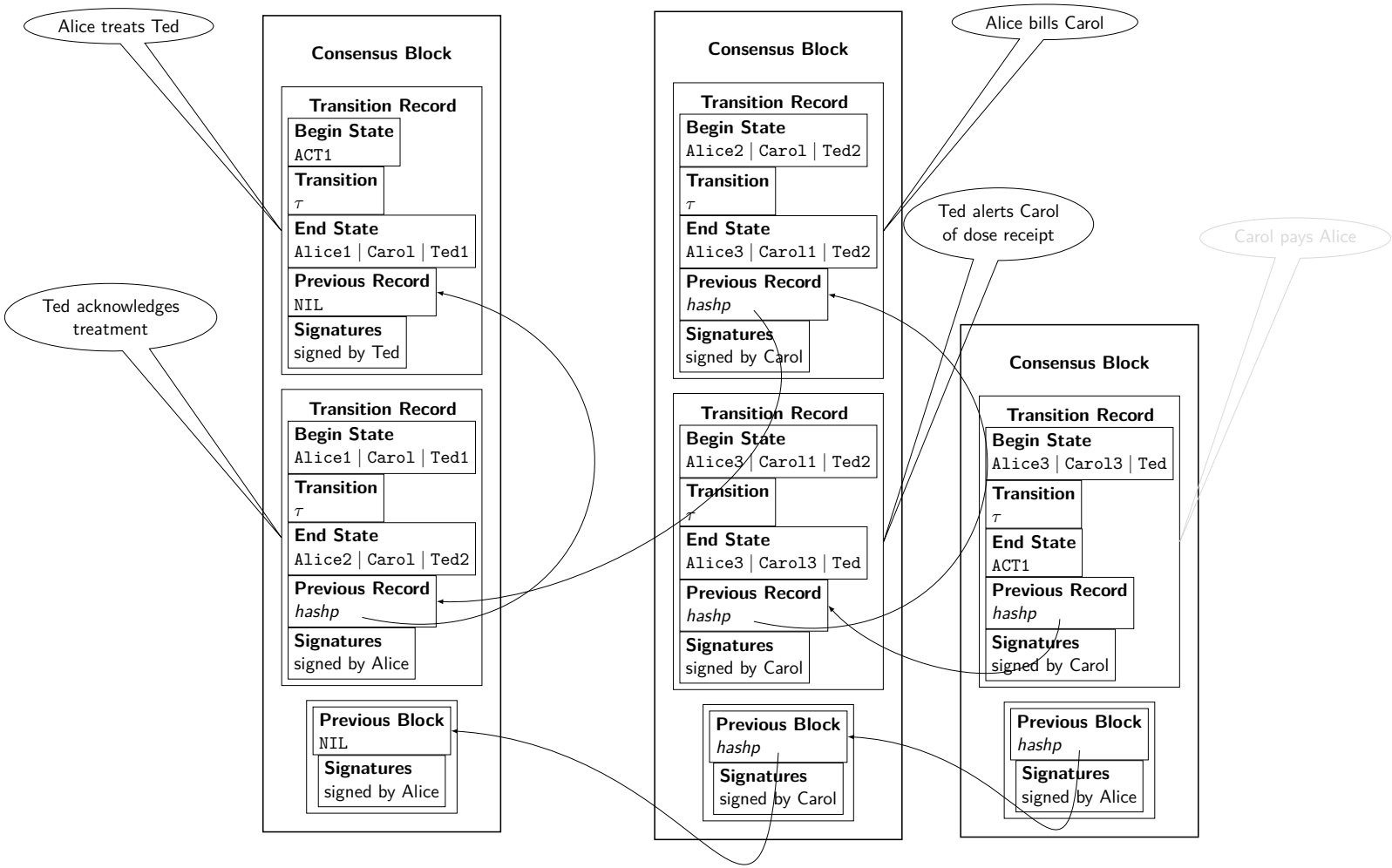
Transition records might be assembled into consensus blocks as follows



Distributed ledgers as carriers of contract executions

blocks

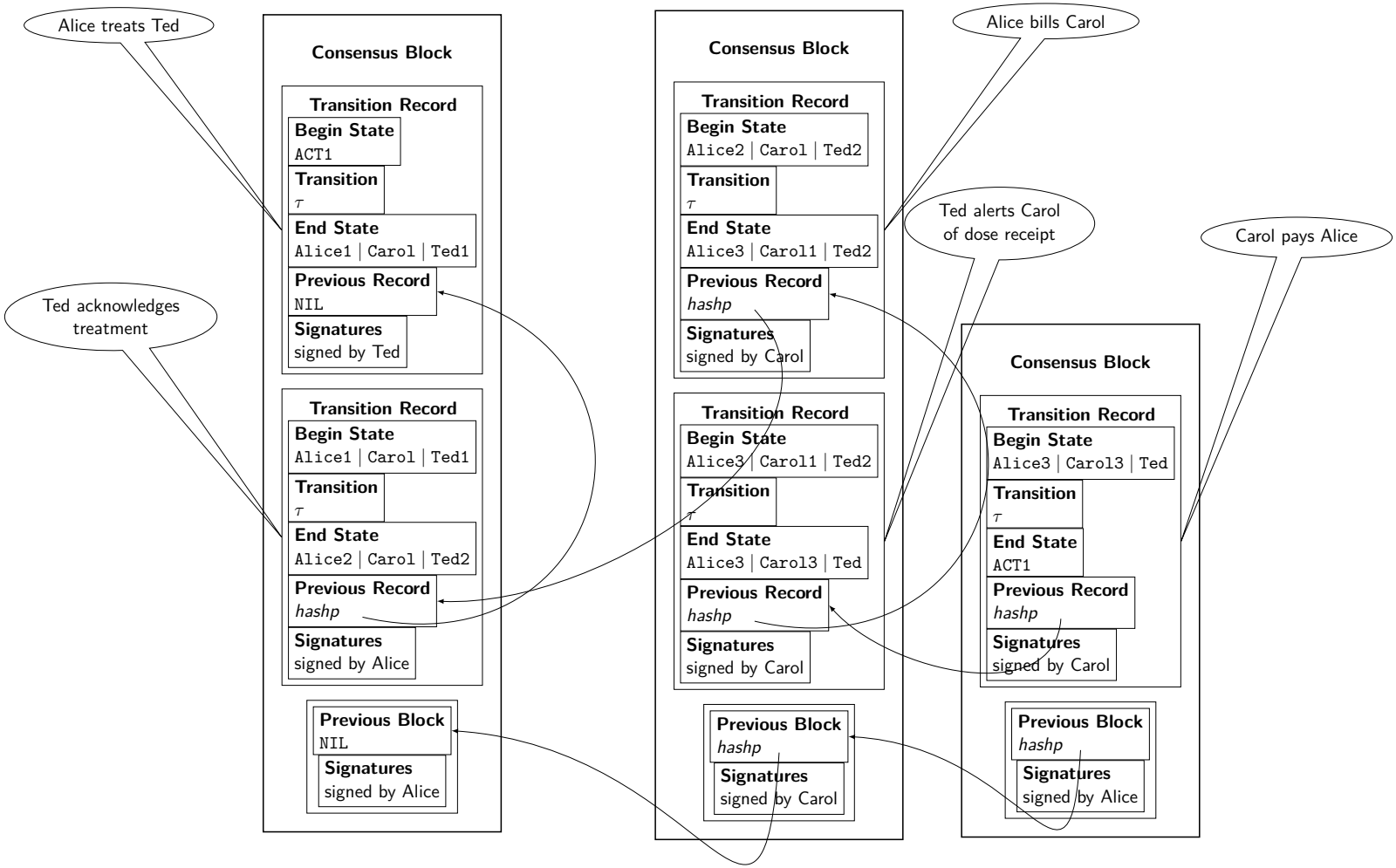
Transition records might be assembled into consensus blocks as follows



Distributed ledgers as carriers of contract executions

blocks

Transition records might be assembled into consensus blocks as follows



The logic of contracts

to be included in a formal system of derivation

Investigators of distributed, concurrent systems commonly speak in terms of the following desiderata

Safety Nothing bad ever happens

It is not the case that Carol pays Alice without Ted's having been previously treated

$\vdash \texttt{ACT1} : \neg(\mathbb{P} - \overline{\texttt{dose}})\langle\overline{\texttt{pay}}\rangle\texttt{true}$

Liveness Something good eventually happens

Whenever Alice treats Ted, Carol eventually pays Alice

$\vdash \texttt{ACT1} : [\texttt{dose}]\mathbb{AE}\langle\overline{\texttt{pay}}\rangle\texttt{true}$

Fairness Something good eventually happens to everyone

No matter what else Alice may be doing, she eventually treats Ted

$\vdash \texttt{Alice} : \mathbb{AP}\langle\overline{\texttt{dose}}\rangle\texttt{true}$

The logic of contracts

to be included in a formal system of derivation

Investigators of distributed, concurrent systems commonly speak in terms of the following desiderata

Safety Nothing bad ever happens

It is not the case that Carol pays Alice without Ted's having been previously treated

$\vdash \texttt{ACT1} : \neg(\mathbb{P} - \overline{\texttt{dose}})\langle\overline{\texttt{pay}}\rangle\texttt{true}$

Liveness Something good eventually happens

Whenever Alice treats Ted, Carol eventually pays Alice

$\vdash \texttt{ACT1} : [\texttt{dose}]\mathbb{AE}\langle\overline{\texttt{pay}}\rangle\texttt{true}$

Fairness Something good eventually happens to everyone

No matter what else Alice may be doing, she eventually treats Ted

$\vdash \texttt{Alice} : \mathbb{AP}\langle\overline{\texttt{dose}}\rangle\texttt{true}$

The logic of contracts

to be included in a formal system of derivation

Investigators of distributed, concurrent systems commonly speak in terms of the following desiderata

Safety Nothing bad ever happens

It is not the case that Carol pays Alice without Ted's having been previously treated

$\vdash \texttt{ACT1} : \neg(\mathbb{P} - \overline{\texttt{dose}})\langle\overline{\texttt{pay}}\rangle\texttt{true}$

Liveness Something good eventually happens

Whenever Alice treats Ted, Carol eventually pays Alice

$\vdash \texttt{ACT1} : [\overline{\texttt{dose}}]\mathbb{AE}\langle\overline{\texttt{pay}}\rangle\texttt{true}$

Fairness Something good eventually happens to everyone

No matter what else Alice may be doing, she eventually treats Ted

$\vdash \texttt{Alice} : \mathbb{AP}\langle\overline{\texttt{dose}}\rangle\texttt{true}$

The logic of contracts

to be included in a formal system of derivation

Investigators of distributed, concurrent systems commonly speak in terms of the following desiderata

Safety Nothing bad ever happens

It is not the case that Carol pays Alice without Ted's having been previously treated

$\vdash \texttt{ACT1} : \neg(\mathbb{P} - \overline{\texttt{dose}})\langle\overline{\texttt{pay}}\rangle\texttt{true}$

Liveness Something good eventually happens

Whenever Alice treats Ted, Carol eventually pays Alice

$\vdash \texttt{ACT1} : [\overline{\texttt{dose}}]\mathbb{AE}\langle\overline{\texttt{pay}}\rangle\texttt{true}$

Fairness Something good eventually happens to everyone

No matter what else Alice may be doing, she eventually treats Ted

$\vdash \texttt{Alice} : \mathbb{AP}\langle\overline{\texttt{dose}}\rangle\texttt{true}$

The logic of contracts

to be included in a formal system of derivation

Investigators of distributed, concurrent systems commonly speak in terms of the following desiderata

Safety Nothing bad ever happens

It is not the case that Carol pays Alice without Ted's having been previously treated

$\vdash \mathtt{ACT1} : \neg(\mathbb{P} - \overline{\mathtt{dose}})\langle\overline{\mathtt{pay}}\rangle\mathtt{true}$

Liveness Something good eventually happens

Whenever Alice treats Ted, Carol eventually pays Alice

$\vdash \mathtt{ACT1} : [\overline{\mathtt{dose}}]\mathbb{AE}\langle\overline{\mathtt{pay}}\rangle\mathtt{true}$

Fairness Something good eventually happens to everyone

No matter what else Alice may be doing, she eventually treats Ted

$\vdash \mathtt{Alice} : \mathbb{AP}\langle\overline{\mathtt{dose}}\rangle\mathtt{true}$

The logic of contracts

to be included in a formal system of derivation

Investigators of distributed, concurrent systems commonly speak in terms of the following desiderata

Safety Nothing bad ever happens

It is not the case that Carol pays Alice without Ted's having been previously treated

$\vdash \mathtt{ACT1} : \neg(\mathbb{P} - \overline{\mathtt{dose}})\langle\overline{\mathtt{pay}}\rangle\mathtt{true}$

Liveness Something good eventually happens

Whenever Alice treats Ted, Carol eventually pays Alice

$\vdash \mathtt{ACT1} : [\overline{\mathtt{dose}}]\mathbb{AE}\langle\overline{\mathtt{pay}}\rangle\mathtt{true}$

Fairness Something good eventually happens to everyone

No matter what else Alice may be doing, she eventually treats Ted

$\vdash \mathtt{Alice} : \mathbb{AP}\langle\overline{\mathtt{dose}}\rangle\mathtt{true}$

Metadata on contract executions

as judgments

Everything is a judgment

- The criteria of well-formedness of a distributed ledger data structure are judgments
  The value in the **End State** *field of the transition record pointed to by* `0AC179DB` *is a state of the LTS*
  $\vdash \texttt{ACT1}: \mathbb{A}\texttt{`0AC179DB.End\_State'} \in \mathcal{X}$
- The criteria by which it is discerned that an execution is a path in an LTS is a judgment
  Extracted from the ledger, $x_0 \xrightarrow{\alpha_1} x_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} x_n$
  $\vdash \texttt{ACT1}: \langle \alpha_1 \rangle \langle \alpha_2 \rangle \cdots \langle \alpha_n \rangle \texttt{true}$
- That such a trace is encoded on a blockchain is a judgment
  $\vdash \texttt{ACT1}: \mathbb{A} \bigwedge_{1 \leqslant i \leqslant n} \texttt{Transition\_Record}(i).\texttt{Transition} = \alpha_i$
- Thus, a contract, operationalized as an LTS and traced in a distributed ledger, corresponds to a set of judgments

Metadata on contract executions

as judgments

Everything is a judgment

- The criteria of well-formedness of a distributed ledger data structure are judgments
  The value in the **End State** *field of the transition record pointed to by* `0AC179DB` *is a state of the LTS*
  $\vdash \texttt{ACT1} : \mathbb{A}\ulcorner\texttt{0AC179DB.End\_State}\urcorner \in \mathcal{X}$
- The criteria by which it is discerned that an execution is a path in an LTS is a judgment
  Extracted from the ledger, $x_0 \xrightarrow{\alpha_1} x_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} x_n$
  $\vdash \texttt{ACT1} : \langle \alpha_1 \rangle \langle \alpha_2 \rangle \cdots \langle \alpha_n \rangle \texttt{true}$
- That such a trace is encoded on a blockchain is a judgment
  $\vdash \texttt{ACT1} : \mathbb{A} \bigwedge_{1 \leqslant i \leqslant n} \texttt{Transition\_Record}(i).\texttt{Transition} = \alpha_i$
- Thus, a contract, operationalized as an LTS and traced in a distributed ledger, corresponds to a set of judgments

Metadata on contract executions

as judgments

Everything is a judgment

- The criteria of well-formedness of a distributed ledger data structure are judgments

  The value in the **End State** *field of the transition record pointed to by* `0AC179DB` *is a state of the LTS*

  $\vdash \texttt{ACT1} : \mathbb{A}\ulcorner\texttt{0AC179DB.End\_State}\urcorner \in \mathcal{X}$

- The criteria by which it is discerned that an execution is a path in an LTS is a judgment

  Extracted from the ledger, $x_0 \xrightarrow{\alpha_1} x_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} x_n$

  $\vdash \texttt{ACT1} : \langle\alpha_1\rangle\langle\alpha_2\rangle\cdots\langle\alpha_n\rangle\texttt{true}$

- That such a trace is encoded on a blockchain is a judgment

  $\vdash \texttt{ACT1} : \mathbb{A} \bigwedge_{1 \leqslant i \leqslant n} \texttt{Transition\_Record}(i).\texttt{Transition} = \alpha_i$

- Thus, a contract, operationalized as an LTS and traced in a distributed ledger, corresponds to a set of judgments

Metadata on contract executions

as judgments

Everything is a judgment

- The criteria of well-formedness of a distributed ledger data structure are judgments
  The value in the **End State** *field of the transition record pointed to by* `0AC179DB` *is a state of the LTS*
  $\vdash \texttt{ACT1} : \mathbb{A}\ulcorner\texttt{0AC179DB.End\_State}\urcorner \in \mathcal{X}$
- The criteria by which it is discerned that an execution is a path in an LTS is a judgment
  Extracted from the ledger, $x_0 \xrightarrow{\alpha_1} x_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} x_n$
  $\vdash \texttt{ACT1} : \langle\alpha_1\rangle\langle\alpha_2\rangle\cdots\langle\alpha_n\rangle\texttt{true}$
- That such a trace is encoded on a blockchain is a judgment
  $\vdash \texttt{ACT1} : \mathbb{A} \bigwedge_{1 \leqslant i \leqslant n} \texttt{Transition\_Record}(i).\texttt{Transition} = \alpha_i$
- Thus, a contract, operationalized as an LTS and traced in a distributed ledger, corresponds to a set of judgments

Metadata on contract executions

as judgments

Everything is a judgment

- The criteria of well-formedness of a distributed ledger data structure are judgments
  The value in the **End State** *field of the transition record pointed to by* `0AC179DB` *is a state of the LTS*
  $\vdash \texttt{ACT1} : \mathbb{A}\ulcorner\texttt{0AC179DB.End\_State}\urcorner \in \mathcal{X}$
- The criteria by which it is discerned that an execution is a path in an LTS is a judgment
  Extracted from the ledger, $x_0 \xrightarrow{\alpha_1} x_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} x_n$
  $\vdash \texttt{ACT1} : \langle\alpha_1\rangle\langle\alpha_2\rangle\cdots\langle\alpha_n\rangle\texttt{true}$
- That such a trace is encoded on a blockchain is a judgment
  $\vdash \texttt{ACT1} : \mathbb{A}\bigwedge_{1\leqslant i\leqslant n}\texttt{Transition\_Record}(i).\texttt{Transition} = \alpha_i$
- Thus, a contract, operationalized as an LTS and traced in a distributed ledger, corresponds to a set of judgments

Metadata on contract executions

as judgments

Everything is a judgment

- The criteria of well-formedness of a distributed ledger data structure are judgments
  The value in the **End State** *field of the transition record pointed to by* `0AC179DB` *is a state of the LTS*
  $\vdash \texttt{ACT1} : \mathbb{A}\ulcorner\texttt{0AC179DB.End\_State}\urcorner \in \mathcal{X}$
- The criteria by which it is discerned that an execution is a path in an LTS is a judgment
  Extracted from the ledger, $x_0 \xrightarrow{\alpha_1} x_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} x_n$
  $\vdash \texttt{ACT1} : \langle\alpha_1\rangle\langle\alpha_2\rangle\cdots\langle\alpha_n\rangle\texttt{true}$
- That such a trace is encoded on a blockchain is a judgment
  $\vdash \texttt{ACT1} : \mathbb{A}\bigwedge_{1\leqslant i\leqslant n}\texttt{Transition\_Record}(i).\texttt{Transition} = \alpha_i$
- Thus, a contract, operationalized as an LTS and traced in a distributed ledger, corresponds to a set of judgments

Verifiability

- We have abstracted a contract as a labeled transition system, $\langle \mathcal{X}, \{\mathcal{R}_a \subseteq \mathcal{X} \times \mathcal{X} | a \in \mathcal{A}\}\rangle$
- We subsequently showed how to encode an execution from such an LTS on a distributed ledger data structure
- From a distributed ledger data structure we can decode a sequence, $x_0 \xrightarrow{\alpha_1} x_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} x_n$, which is putatively an execution

well-formedness $x_0 = \texttt{ACT1}$ and for $1 \leqslant i \leqslant n$, $x_i \in \mathcal{X}$, $\alpha_i \in \mathcal{A}$

admissibility $x_0 \xrightarrow{\alpha_1} x_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} x_n$ is, in fact, an execution of $\texttt{ACT1}$

derivability judgments such as $\vdash \texttt{ACT1} : \neg(\mathbb{P} - \overline{\texttt{dose}})\langle\overline{\texttt{pay}}\rangle\texttt{true}$ which may decorate either transition records or blocks must be known to be derivable

Verifiability

- We have abstracted a contract as a labeled transition system, $\langle \mathcal{X}, \{\mathcal{R}_a \subseteq \mathcal{X} \times \mathcal{X} | a \in \mathcal{A}\}\rangle$
- We subsequently showed how to encode an execution from such an LTS on a distributed ledger data structure
- From a distributed ledger data structure we can decode a sequence, $x_0 \xrightarrow{\alpha_1} x_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} x_n$, which is putatively an execution

well-formedness $x_0 = \texttt{ACT1}$ and for $1 \leqslant i \leqslant n$, $x_i \in \mathcal{X}$, $\alpha_i \in \mathcal{A}$

admissibility $x_0 \xrightarrow{\alpha_1} x_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} x_n$ is, in fact, an execution of $\texttt{ACT1}$

derivability judgments such as $\vdash \texttt{ACT1} : \neg(\mathbb{P} - \overline{\texttt{dose}})\langle\overline{\texttt{pay}}\rangle\texttt{true}$ which may decorate either transition records or blocks must be known to be derivable

Verifiability

- We have abstracted a contract as a labeled transition system, $\langle \mathcal{X}, \{\mathcal{R}_a \subseteq \mathcal{X} \times \mathcal{X} | a \in \mathcal{A}\}\rangle$
- We subsequently showed how to encode an execution from such an LTS on a distributed ledger data structure
- From a distributed ledger data structure we can decode a sequence, $x_0 \xrightarrow{\alpha_1} x_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} x_n$, which is putatively an execution

well-formedness $x_0 = \texttt{ACT1}$ and for $1 \leqslant i \leqslant n$, $x_i \in \mathcal{X}$, $\alpha_i \in \mathcal{A}$

admissibility $x_0 \xrightarrow{\alpha_1} x_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} x_n$ is, in fact, an execution of $\texttt{ACT1}$

derivability judgments such as $\vdash \texttt{ACT1} : \neg(\mathbb{P} - \overline{\texttt{dose}})\langle\overline{\texttt{pay}}\rangle\texttt{true}$ which may decorate either transition records or blocks must be known to be derivable

Verifiability

- We have abstracted a contract as a labeled transition system, $\langle \mathcal{X}, \{\mathcal{R}_a \subseteq \mathcal{X} \times \mathcal{X} | a \in \mathcal{A}\}\rangle$
- We subsequently showed how to encode an execution from such an LTS on a distributed ledger data structure
- From a distributed ledger data structure we can decode a sequence, $x_0 \xrightarrow{\alpha_1} x_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} x_n$, which is putatively an execution

well-formedness $x_0 = \texttt{ACT1}$ and for $1 \leqslant i \leqslant n$, $x_i \in \mathcal{X}$, $\alpha_i \in \mathcal{A}$

admissibility $x_0 \xrightarrow{\alpha_1} x_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} x_n$ is, in fact, an execution of $\texttt{ACT1}$

derivability judgments such as $\vdash \texttt{ACT1} : \neg(\mathbb{P} - \overline{\texttt{dose}})\langle\overline{\texttt{pay}}\rangle\texttt{true}$ which may decorate either transition records or blocks must be known to be derivable

Verifiability

- We have abstracted a contract as a labeled transition system, $\langle \mathcal{X}, \{\mathcal{R}_a \subseteq \mathcal{X} \times \mathcal{X} | a \in \mathcal{A}\}\rangle$
- We subsequently showed how to encode an execution from such an LTS on a distributed ledger data structure
- From a distributed ledger data structure we can decode a sequence, $x_0 \xrightarrow{\alpha_1} x_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} x_n$, which is putatively an execution

well-formedness $x_0 = \texttt{ACT1}$ and for $1 \leqslant i \leqslant n$, $x_i \in \mathcal{X}$, $\alpha_i \in \mathcal{A}$

admissibility $x_0 \xrightarrow{\alpha_1} x_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} x_n$ is, in fact, an execution of $\texttt{ACT1}$

derivability judgments such as $\vdash \texttt{ACT1} : \neg(\mathbb{P} - \overline{\texttt{dose}})\langle\overline{\texttt{pay}}\rangle\texttt{true}$ which may decorate either transition records or blocks must be known to be derivable

Verifiability

- We have abstracted a contract as a labeled transition system, $\langle \mathcal{X}, \{\mathcal{R}_a \subseteq \mathcal{X} \times \mathcal{X} | a \in \mathcal{A}\}\rangle$
- We subsequently showed how to encode an execution from such an LTS on a distributed ledger data structure
- From a distributed ledger data structure we can decode a sequence, $x_0 \xrightarrow{\alpha_1} x_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} x_n$, which is putatively an execution

well-formedness $x_0 = \texttt{ACT1}$ and for $1 \leqslant i \leqslant n$, $x_i \in \mathcal{X}$, $\alpha_i \in \mathcal{A}$

admissibility $x_0 \xrightarrow{\alpha_1} x_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} x_n$ is, in fact, an execution of $\texttt{ACT1}$

derivability judgments such as $\vdash \texttt{ACT1} : \neg(\mathbb{P} - \overline{\texttt{dose}})\langle\overline{\texttt{pay}}\rangle\texttt{true}$ which may decorate either transition records or blocks must be known to be derivable

Verifiability

- We have abstracted a contract as a labeled transition system, $\langle \mathcal{X}, \{\mathcal{R}_a \subseteq \mathcal{X} \times \mathcal{X} | a \in \mathcal{A}\}\rangle$
- We subsequently showed how to encode an execution from such an LTS on a distributed ledger data structure
- From a distributed ledger data structure we can decode a sequence, $x_0 \xrightarrow{\alpha_1} x_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} x_n$, which is putatively an execution

well-formedness $x_0 = \texttt{ACT1}$ and for $1 \leqslant i \leqslant n$, $x_i \in \mathcal{X}$, $\alpha_i \in \mathcal{A}$

admissibility $x_0 \xrightarrow{\alpha_1} x_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} x_n$ is, in fact, an execution of $\texttt{ACT1}$

derivability judgments such as $\vdash \texttt{ACT1} : \neg(\mathbb{P} - \overline{\texttt{dose}})\langle\overline{\texttt{pay}}\rangle\texttt{true}$ which may decorate either transition records or blocks must be known to be derivable

Accountability

unverifiability

ill-formedness There is some earliest signed transition record (and block) in which ill-formedness is detected

inadmissibility There is some earliest signed transition record which records a transition in a sequence which is not an execution of `ACT1`

nonderivability There is some earliest signed record or block containing a judgment for which no derivation is known

Accountability

unverifiability

ill-formedness There is some earliest signed transition record (and block) in which ill-formedness is detected

inadmissibility There is some earliest signed transition record which records a transition in a sequence which is not an execution of `ACT1`

nonderivability There is some earliest signed record or block containing a judgment for which no derivation is known

Accountability

unverifiability

ill-formedness There is some earliest signed transition record (and block) in which ill-formedness is detected

inadmissibility There is some earliest signed transition record which records a transition in a sequence which is not an execution of `ACT1`

nonderivability There is some earliest signed record or block containing a judgment for which no derivation is known

Accountability

unverifiability

ill-formedness There is some earliest signed transition record (and block) in which ill-formedness is detected

inadmissibility There is some earliest signed transition record which records a transition in a sequence which is not an execution of `ACT1`

nonderivability There is some earliest signed record or block containing a judgment for which no derivation is known

δεῖξις (1) mode of proof; (2) proof, specimen; (3) display, exhibition

deixis (1) designating words relating to the time and place of utterance; (2) proving directly