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

Contracts

Digital contracts

Digital contract governance
In this section:

Contracts

Digital contracts

Digital contract governance
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
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
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

<table>
<thead>
<tr>
<th><strong>Contracts</strong></th>
<th><strong>Analog</strong></th>
<th><strong>Digital</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Traditional</strong></td>
<td><strong>Smart</strong></td>
</tr>
<tr>
<td><strong>Syntax</strong></td>
<td>Late Renaissance English with Latin neologisms</td>
<td>Yet another OO language</td>
</tr>
<tr>
<td><strong>Semantics</strong></td>
<td>What your lawyer says</td>
<td>What your programmer says</td>
</tr>
<tr>
<td><strong>Author</strong></td>
<td>Your lawyer</td>
<td>Your programmer</td>
</tr>
<tr>
<td><strong>Performance tracking</strong></td>
<td>Humint</td>
<td>Distributed ledger</td>
</tr>
<tr>
<td><strong>Guarantees</strong></td>
<td>What your lawyer says</td>
<td>What your programmer says</td>
</tr>
<tr>
<td><strong>Violations</strong></td>
<td>When your lawyer says</td>
<td>When your programmer says</td>
</tr>
</tbody>
</table>
# Contracts, smart and otherwise

<table>
<thead>
<tr>
<th>Contracts</th>
<th>Analog</th>
<th>Digital</th>
<th>Deixis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax</strong></td>
<td>Traditional</td>
<td>Smart</td>
<td>Domain specific language for specifying behavior only</td>
</tr>
<tr>
<td>Late Renaissance English with Latin neologisms</td>
<td>Yet another OO language</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Semantics</strong></td>
<td></td>
<td></td>
<td>Mathematically defined categorical, logical and operational semantics</td>
</tr>
<tr>
<td>What your lawyer says</td>
<td>What your programmer says</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Author</strong></td>
<td>Your lawyer</td>
<td>Your programmer</td>
<td>Your DBA</td>
</tr>
<tr>
<td><strong>Performance tracking</strong></td>
<td>Humint</td>
<td>Distributed ledger</td>
<td>Distributed ledger</td>
</tr>
<tr>
<td><strong>Guarantees</strong></td>
<td>What your lawyer says</td>
<td>What your programmer says</td>
<td>What your specification says</td>
</tr>
<tr>
<td><strong>Violations</strong></td>
<td>When your lawyer says</td>
<td>When your programmer says</td>
<td>When your specification says</td>
</tr>
</tbody>
</table>
## Contracts, smart and otherwise

<table>
<thead>
<tr>
<th>Contracts</th>
<th><strong>Analog</strong></th>
<th><strong>Digital</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Traditional</strong></td>
<td><strong>Smart</strong></td>
<td><strong>Deixis</strong></td>
</tr>
<tr>
<td><strong>Syntax</strong></td>
<td>Late Renaissance English with Latin neologisms</td>
<td>Yet another OO language</td>
</tr>
<tr>
<td><strong>Semantics</strong></td>
<td>What your lawyer says</td>
<td>What your programmer says</td>
</tr>
<tr>
<td><strong>Author</strong></td>
<td>Your lawyer</td>
<td>Your programmer</td>
</tr>
<tr>
<td><strong>Performance tracking</strong></td>
<td>Humint</td>
<td>Distributed ledger</td>
</tr>
<tr>
<td><strong>Guarantees</strong></td>
<td>What your lawyer says</td>
<td>What your programmer says</td>
</tr>
<tr>
<td><strong>Violations</strong></td>
<td>When your lawyer says</td>
<td>When your programmer says</td>
</tr>
</tbody>
</table>
## Contracts, smart and otherwise

<table>
<thead>
<tr>
<th>Contracts</th>
<th>Analog</th>
<th>Digital</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Traditional</td>
<td>Smart</td>
</tr>
<tr>
<td>Syntax</td>
<td>Late Renaissance English with Latin neologisms</td>
<td>Yet another OO language</td>
</tr>
<tr>
<td>Semantics</td>
<td>What your lawyer says</td>
<td>What your programmer says</td>
</tr>
<tr>
<td>Author</td>
<td>Your lawyer</td>
<td>Your programmer</td>
</tr>
<tr>
<td>Performance tracking</td>
<td>Humint</td>
<td>Distributed ledger</td>
</tr>
<tr>
<td>Guarantees</td>
<td>What your lawyer says</td>
<td>What your programmer says</td>
</tr>
<tr>
<td>Violations</td>
<td>When your lawyer says</td>
<td>When your programmer says</td>
</tr>
</tbody>
</table>
# Contracts, smart and otherwise

<table>
<thead>
<tr>
<th>Contracts</th>
<th>Analog</th>
<th>Digital</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax</strong></td>
<td>Traditional</td>
<td>Smart</td>
</tr>
<tr>
<td>Late Renaissance English with Latin neologisms</td>
<td>Yet another OO language</td>
<td>Domain specific language for specifying behavior only</td>
</tr>
<tr>
<td><strong>Semantics</strong></td>
<td>What your lawyer says</td>
<td>What your programmer says</td>
</tr>
<tr>
<td><strong>Author</strong></td>
<td>Your lawyer</td>
<td>Your programmer</td>
</tr>
<tr>
<td><strong>Performance tracking</strong></td>
<td>Humint</td>
<td>Distributed ledger</td>
</tr>
<tr>
<td><strong>Guarantees</strong></td>
<td>What your lawyer says</td>
<td>What your programmer says</td>
</tr>
<tr>
<td><strong>Violations</strong></td>
<td>When your lawyer says</td>
<td>When your programmer says</td>
</tr>
</tbody>
</table>
# Contracts, smart and otherwise

<table>
<thead>
<tr>
<th>Contracts</th>
<th>Analog</th>
<th>Digital</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax</strong></td>
<td>Traditional</td>
<td>Smart</td>
</tr>
<tr>
<td></td>
<td>Late Renaissance English with Latin neologisms</td>
<td>Yet another OO language</td>
</tr>
<tr>
<td><strong>Semantics</strong></td>
<td>What your lawyer says</td>
<td>What your programmer says</td>
</tr>
<tr>
<td><strong>Author</strong></td>
<td>Your lawyer</td>
<td>Your programmer</td>
</tr>
<tr>
<td><strong>Performance tracking</strong></td>
<td>Humint</td>
<td>Distributed ledger</td>
</tr>
<tr>
<td><strong>Guarantees</strong></td>
<td>What your lawyer says</td>
<td>What your programmer says</td>
</tr>
<tr>
<td><strong>Violations</strong></td>
<td>When your lawyer says</td>
<td>When your programmer says</td>
</tr>
</tbody>
</table>
## Contracts, smart and otherwise

<table>
<thead>
<tr>
<th>Contracts</th>
<th>Analog</th>
<th>Digital</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Traditional</td>
<td>Smart</td>
</tr>
<tr>
<td><strong>Syntax</strong></td>
<td>Late Renaissance English with Latin neologisms</td>
<td>Yet another OO language</td>
</tr>
<tr>
<td><strong>Semantics</strong></td>
<td>What your lawyer says</td>
<td>What your programmer says</td>
</tr>
<tr>
<td><strong>Author</strong></td>
<td>Your lawyer</td>
<td>Your programmer</td>
</tr>
<tr>
<td><strong>Performance tracking</strong></td>
<td>Humint</td>
<td>Distributed ledger</td>
</tr>
<tr>
<td>** Guarantees**</td>
<td>What your lawyer says</td>
<td>What your programmer says</td>
</tr>
<tr>
<td><strong>Violations</strong></td>
<td>When your lawyer says</td>
<td>When your programmer says</td>
</tr>
</tbody>
</table>
In this section:

Contracts

Digital contracts

Digital contract governance
Alice’s behavior as algebra

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

\[
\begin{align*}
\text{Alice} & \overset{\text{def}}{=} \text{dose}.\text{Alice1} \\
\text{Alice1} & \overset{\text{def}}{=} \text{ackdose}.\text{Alice2} \\
\text{Alice2} & \overset{\text{def}}{=} \text{bill}.\text{Alice3} \\
\text{Alice3} & \overset{\text{def}}{=} \text{pay}.\text{Alice}
\end{align*}
\]
Alice’s behavior as algebra

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

\[
\begin{align*}
\text{Alice} & \overset{\text{def}}{=} \text{dose}.\text{Alice}_1 \\
\text{Alice}_1 & \overset{\text{def}}{=} \text{ackdose}.\text{Alice}_2 \\
\text{Alice}_2 & \overset{\text{def}}{=} \text{bill}.\text{Alice}_3 \\
\text{Alice}_3 & \overset{\text{def}}{=} \text{pay}.\text{Alice}
\end{align*}
\]
Alice’s behavior as algebra

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

\[
\begin{align*}
\text{Alice} & \overset{\text{def}}{=} \text{dose.Alice1} \\
\text{Alice1} & \overset{\text{def}}{=} \text{ackdose.Alice2} \\
\text{Alice2} & \overset{\text{def}}{=} \text{bill.Alice3} \\
\text{Alice3} & \overset{\text{def}}{=} \text{pay.Alice}
\end{align*}
\]
We present Alice’s behavior as an algebraic system of equations of the same sort that gives rise to regular expressions with an associated interpretation as a labeled transitions system (LTS)

\[
\begin{align*}
\text{Alice} & \overset{\text{def}}{=} \text{dose} . \text{Alice1} \\
\text{Alice1} & \overset{\text{def}}{=} \text{ackdose} . \text{Alice2} \\
\text{Alice2} & \overset{\text{def}}{=} \text{bill} . \text{Alice3} \\
\text{Alice3} & \overset{\text{def}}{=} \text{pay} . \text{Alice}
\end{align*}
\]
Carol’s behavior as algebra

Carol \overset{\text{def}}{=} \text{bill} \cdot \text{Carol1} + \text{rcvdose} \cdot \text{Carol2}

\text{Carol1} \overset{\text{def}}{=} \text{rcvdose} \cdot \text{Carol3}

\text{Carol2} \overset{\text{def}}{=} \text{bill} \cdot \text{Carol3}

\text{Carol3} \overset{\text{def}}{=} \text{pay} \cdot \text{Carol}
Carol’s behavior as algebra

\begin{align*}
\text{Carol} & \overset{\text{def}}{=} \text{bill}.\text{Carol1} + \\
& \quad \text{rcvdose}.\text{Carol2} \\
\text{Carol1} & \overset{\text{def}}{=} \text{rcvdose}.\text{Carol3} \\
\text{Carol2} & \overset{\text{def}}{=} \text{bill}.\text{Carol3} \\
\text{Carol3} & \overset{\text{def}}{=} \overline{\text{pay}}.\text{Carol}
\end{align*}
Carol’s behavior as algebra

\[
\begin{align*}
\text{Carol} & \overset{\text{def}}{=} \text{bill}.\text{Carol1} + \\
& \quad \text{rcvdose}.\text{Carol2} \\
\text{Carol1} & \overset{\text{def}}{=} \text{rcvdose}.\text{Carol3} \\
\text{Carol2} & \overset{\text{def}}{=} \text{bill}.\text{Carol3} \\
\text{Carol3} & \overset{\text{def}}{=} \overline{\text{pay}}.\text{Carol}
\end{align*}
\]
Ted’s behavior as algebra

Ted def dose.Ted1
Ted1 def ackdose.Ted2 + rcvdose.Ted3
Ted2 def rcvdose.Ted
Ted3 def ackdose.Ted

Diagram:

Ted
\arrow{down}
dose
ackdose
rcvdose

Ted1
 Ted2
 Ted3
Ted’s behavior as algebra

\[
\begin{align*}
\text{Ted} & \stackrel{\text{def}}{=} \text{dose} \cdot \text{Ted}_1 \\
\text{Ted}_1 & \stackrel{\text{def}}{=} \text{ackdose} \cdot \text{Ted}_2 + \text{rcvdose} \cdot \text{Ted}_3 \\
\text{Ted}_2 & \stackrel{\text{def}}{=} \text{rcvdose} \cdot \text{Ted} \\
\text{Ted}_3 & \stackrel{\text{def}}{=} \text{ackdose} \cdot \text{Ted}
\end{align*}
\]
Ted’s behavior as algebra

\[
\begin{align*}
    \text{Ted} & \overset{\text{def}}{=} \text{dose}.\text{Ted1} \\
    \text{Ted1} & \overset{\text{def}}{=} \text{ackdose}.\text{Ted2} + \\
    & \quad \text{rcvdose}.\text{Ted3} \\
    \text{Ted2} & \overset{\text{def}}{=} \text{rcvdose}.\text{Ted} \\
    \text{Ted3} & \overset{\text{def}}{=} \text{ackdose}.\text{Ted}
\end{align*}
\]
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

\[
\begin{align*}
  x \xrightarrow{a} x' & \quad y \xrightarrow{a} y' \\
  x \mid y & \xrightarrow{\tau} x' \mid y'
\end{align*}
\]

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

\[
\begin{align*}
  \text{Alice} \xrightarrow{\text{dose}} & \quad \text{Alice}_1 \\
  \text{Ted} \xrightarrow{\text{dose}} & \quad \text{Ted}_1 \\
  \text{Alice} & \mid \text{Ted} \xrightarrow{\tau} \text{Alice}_1 \mid \text{Ted}_1
\end{align*}
\]

and via

\[
\begin{align*}
  \text{Ted}_1 \xrightarrow{\text{ackdose}} & \quad \text{Ted}_2 \\
  \text{Alice}_1 \xrightarrow{\text{ackdose}} & \quad \text{Alice}_2 \\
  \text{Ted}_1 \mid \text{Alice}_1 & \xrightarrow{\tau} \text{Ted}_2 \mid \text{Alice}_2
\end{align*}
\]
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

\[
\begin{align*}
\begin{array}{c}
x \xrightarrow{a} x' \\
y \xrightarrow{a} y'
\end{array}
\end{align*}
\]

\[
\begin{align*}
x | y \xrightarrow{\tau} x' | y'
\end{align*}
\]

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

\[
\begin{align*}
\text{Alice} \xrightarrow{\text{dose}} \text{Alice}_1 & \quad \text{Ted} \xrightarrow{\text{dose}} \text{Ted}_1 \\
\text{Alice} | \text{Ted} \xrightarrow{\tau} \text{Alice}_1 | \text{Ted}_1
\end{align*}
\]

and via

\[
\begin{align*}
\text{Ted}_1 \xrightarrow{\text{ackdose}} \text{Ted}_2 & \quad \text{Alice}_1 \xrightarrow{\text{ackdose}} \text{Alice}_2 \\
\text{Ted}_1 | \text{Alice}_1 \xrightarrow{\tau} \text{Ted}_2 | \text{Alice}_2
\end{align*}
\]
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

\[
\begin{align*}
    x \xrightarrow{\bar{a}} x' & \quad \quad y \xrightarrow{a} y' \\
    x \parallel y \xrightarrow{\tau} x' \parallel y'
\end{align*}
\]

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

\[
\text{Alice dose} \quad \text{Alice1} \quad \text{Ted dose} \quad \text{Ted1}
\]

\[
\text{Alice} \xrightarrow{\tau} \text{Alice1} \parallel \text{Ted1}
\]

and via

\[
\text{Ted1 dose} \quad \text{Ted2} \quad \text{Alice1 dose} \quad \text{Alice2}
\]

\[
\text{Ted1} \parallel \text{Alice1} \xrightarrow{\tau} \text{Ted2} \parallel \text{Alice2}
\]
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

\[
\begin{align*}
x \xrightarrow{\bar{a}} x' & \quad y \xrightarrow{a} y' \\
\hline
x \mid y \xrightarrow{\tau} x' \mid y'
\end{align*}
\]

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

\[
\begin{align*}
\text{Alice } & \xrightarrow{\text{dose}} \text{Alice1} & \text{Ted } & \xrightarrow{\text{dose}} \text{Ted1} \\
\text{Alice} \mid \text{Ted } & \xrightarrow{\tau} \text{Alice1} \mid \text{Ted1}
\end{align*}
\]

and via

\[
\begin{align*}
\text{Ted1 } & \xrightarrow{\text{ackdose}} \text{Ted2} & \text{Alice1 } & \xrightarrow{\text{ackdose}} \text{Alice2} \\
\text{Ted1} \mid \text{Alice1 } & \xrightarrow{\tau} \text{Ted2} \mid \text{Alice2}
\end{align*}
\]
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

\[
\begin{align*}
  x \xrightarrow{a} x' & \quad y \xrightarrow{a} y' \\
  x \mid y \xrightarrow{\tau} x' \mid y'
\end{align*}
\]

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

\[
\begin{align*}
  &\text{Alice} \xrightarrow{\text{dose}} Alice_1 & &\text{Ted} \xrightarrow{\text{dose}} Ted_1 \\
  &Alice \mid Ted \xrightarrow{\tau} Alice_1 \mid Ted_1
\end{align*}
\]

and via

\[
\begin{align*}
  &Ted_1 \xrightarrow{\text{ackdose}} Ted_2 & &Alice_1 \xrightarrow{\text{ackdose}} Alice_2 \\
  &Ted_1 \mid Alice_1 \xrightarrow{\tau} Ted_2 \mid Alice_2
\end{align*}
\]
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

\[
\begin{align*}
  x & \xrightarrow{\bar{a}} x' \\
  y & \xrightarrow{a} y' \\
  x \mid y & \xrightarrow{\tau} x' \mid y'
\end{align*}
\]

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

\[
\begin{align*}
  \text{Alice} & \xrightarrow{\text{dose}} \text{Alice}_1 \\
  \text{Ted} & \xrightarrow{\text{dose}} \text{Ted}_1 \\
  \text{Alice} \mid \text{Ted} & \xrightarrow{\tau} \text{Alice}_1 \mid \text{Ted}_1
\end{align*}
\]

and via

\[
\begin{align*}
  \text{Ted}_1 & \xrightarrow{\text{ack dose}} \text{Ted}_2 \\
  \text{Alice}_1 & \xrightarrow{\text{ack dose}} \text{Alice}_2 \\
  \text{Ted}_1 \mid \text{Alice}_1 & \xrightarrow{\tau} \text{Ted}_2 \mid \text{Alice}_2
\end{align*}
\]
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

\[
\begin{align*}
  x \xrightarrow{\bar{a}} x' & \quad y \xrightarrow{a} y' \\
  x | y \xrightarrow{\tau} x' | y'
\end{align*}
\]

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

Alice \xrightarrow{\text{dose}} Alice1 \quad Ted \xrightarrow{\text{dose}} Ted1

Alice | Ted \xrightarrow{\tau} Alice1 | Ted1

and via

Ted1 \xrightarrow{\text{ackdose}} Ted2 \quad Alice1 \xrightarrow{\text{ackdose}} Alice2

Ted1 | Alice1 \xrightarrow{\tau} Ted2 | 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

\[
\begin{align*}
  x \xrightarrow{a} x' & \quad y \xrightarrow{a} y' \\
  x \mid y \xrightarrow{\tau} x' \mid y'
\end{align*}
\]

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

\[
\begin{align*}
  & \quad \text{Alice} \xrightarrow{\text{dose}} \text{Alice}_1 \quad \text{Ted} \xrightarrow{\text{dose}} \text{Ted}_1 \\
\rightarrow & \quad \text{Alice} \mid \text{Ted} \xrightarrow{\tau} \text{Alice}_1 \mid \text{Ted}_1
\end{align*}
\]

and via

\[
\begin{align*}
  & \quad \text{Ted}_1 \xrightarrow{\text{ackdose}} \text{Ted}_2 \quad \text{Alice}_1 \xrightarrow{\text{ackdose}} \text{Alice}_2 \\
\rightarrow & \quad \text{Ted}_1 \mid \text{Alice}_1 \xrightarrow{\tau} \text{Ted}_2 \mid \text{Alice}_2
\end{align*}
\]
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:

\[
\text{ACT1} \overset{\text{def}}{=} \text{Alice} \mid \text{Carol} \mid \text{Ted}
\]

\[
\begin{align*}
\text{Alice} & \overset{\text{def}}{=} \text{dose.Alice1} \\
\text{Alice1} & \overset{\text{def}}{=} \text{ackdose.Alice2} \\
\text{Alice2} & \overset{\text{def}}{=} \text{bill.Alice3} \\
\text{Alice3} & \overset{\text{def}}{=} \text{pay.Alice} \\
\text{Carol} & \overset{\text{def}}{=} \text{bill.Carol1} + \text{rcvdose.Carol2} \\
\text{Carol1} & \overset{\text{def}}{=} \text{rcvdose.Carol3} \\
\text{Carol2} & \overset{\text{def}}{=} \text{bill.Carol3} \\
\text{Carol3} & \overset{\text{def}}{=} \text{pay.Carol} \\
\text{Ted} & \overset{\text{def}}{=} \text{dose.Ted1} \\
\text{Ted1} & \overset{\text{def}}{=} \text{ackdose.Ted2} + \text{rcvdose.Ted3} \\
\text{Ted2} & \overset{\text{def}}{=} \text{rcvdose.Ted} \\
\text{Ted3} & \overset{\text{def}}{=} \text{ackdose.Ted}
\end{align*}
\]
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

\[
\text{ACT1} \overset{\text{def}}{=} \text{Alice} \mid \text{Carol} \mid \text{Ted}
\]

\[
\begin{align*}
\text{Alice} & \overset{\text{def}}{=} \text{dose.Alice1} \\
\text{Alice1} & \overset{\text{def}}{=} \text{ackdose.Alice2} \\
\text{Alice2} & \overset{\text{def}}{=} \text{bill.Alice3} \\
\text{Alice3} & \overset{\text{def}}{=} \text{pay.Alice} \\
\text{Carol} & \overset{\text{def}}{=} \text{bill.Carol1} + \text{rcvdose.Carol2} \\
\text{Carol1} & \overset{\text{def}}{=} \text{rcvdose.Carol3} \\
\text{Carol2} & \overset{\text{def}}{=} \text{bill.Carol3} \\
\text{Carol3} & \overset{\text{def}}{=} \text{pay.Carol} \\
\text{Ted} & \overset{\text{def}}{=} \text{dose.Ted1} \\
\text{Ted1} & \overset{\text{def}}{=} \text{ackdose.Ted2 + rcvdose.Ted3} \\
\text{Ted2} & \overset{\text{def}}{=} \text{rcvdose.Ted} \\
\text{Ted3} & \overset{\text{def}}{=} \text{ackdose.Ted}
\end{align*}
\]
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

\[
\text{ACT1} \overset{\text{def}}{=} \text{Alice} \mid \text{Carol} \mid \text{Ted}
\]

\[
\begin{align*}
\text{Alice} & \overset{\text{def}}{=} \text{dose.Alice1} \\
\text{Alice1} & \overset{\text{def}}{=} \text{ackdose.Alice2} \\
\text{Alice2} & \overset{\text{def}}{=} \text{bill.Alice3} \\
\text{Alice3} & \overset{\text{def}}{=} \text{pay.Alice} \\
\text{Carol} & \overset{\text{def}}{=} \text{bill.Carol1} + \\
& \quad \text{rcvdose.Carol2} \\
\text{Carol1} & \overset{\text{def}}{=} \text{rcvdose.Carol3} \\
\text{Carol2} & \overset{\text{def}}{=} \text{bill.Carol3} \\
\text{Carol3} & \overset{\text{def}}{=} \text{pay.Carol} \\
\text{Ted} & \overset{\text{def}}{=} \text{dose.Ted1} \\
\text{Ted1} & \overset{\text{def}}{=} \text{ackdose.Ted2} + \\
& \quad \text{rcvdose.Ted3} \\
\text{Ted2} & \overset{\text{def}}{=} \text{rcvdose.Ted} \\
\text{Ted3} & \overset{\text{def}}{=} \text{ackdose.Ted}
\end{align*}
\]
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

\[
\text{ACT1} \overset{\text{def}}{=} \text{Alice} | \text{Carol} | \text{Ted}
\]

\[
\begin{align*}
\text{Alice} & \overset{\text{def}}{=} \text{dose.Alice1} \\
\text{Alice1} & \overset{\text{def}}{=} \text{ackdose.Alice2} \\
\text{Alice2} & \overset{\text{def}}{=} \text{bill.Alice3} \\
\text{Alice3} & \overset{\text{def}}{=} \text{pay.Alice} \\
\text{Carol} & \overset{\text{def}}{=} \text{bill.Carol1} + \\
& \quad \text{rcvdose.Carol2} \\
\text{Carol1} & \overset{\text{def}}{=} \text{rcvdose.Carol3} \\
\text{Carol2} & \overset{\text{def}}{=} \text{bill.Carol3} \\
\text{Carol3} & \overset{\text{def}}{=} \text{pay.Carol} \\
\text{Ted} & \overset{\text{def}}{=} \text{dose.Ted1} \\
\text{Ted1} & \overset{\text{def}}{=} \text{ackdose.Ted2} + \\
& \quad \text{rcvdose.Ted3} \\
\text{Ted2} & \overset{\text{def}}{=} \text{rcvdose.Ted} \\
\text{Ted3} & \overset{\text{def}}{=} \text{ackdose.Ted}
\end{align*}
\]
\[ \text{ACT1} \overset{\text{def}}{=} \text{Alice} \mid \text{Carol} \mid \text{Ted} \]
\( \text{ACT1} \overset{\text{def}}{=} \text{Alice} | \text{Carol} | \text{Ted} \)
\[ \text{ACT1} \overset{\text{def}}{=} \text{Alice} \mid \text{Carol} \mid \text{Ted} \]
In this section:

Contracts

Digital contracts

Digital contract governance
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

  \[
  \text{ACT1} \xrightarrow{\tau} \\
  \text{Alice}_1 | \text{Carol} | \text{Ted}_1 \xrightarrow{\tau} \\
  \text{Alice}_2 | \text{Carol} | \text{Ted}_2 \xrightarrow{\tau} \\
  \text{Alice}_3 | \text{Carol}_1 | \text{Ted}_2 \xrightarrow{\tau} \\
  \text{ACT1}
  \]

  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 $\text{ACT}_1$, such as

$$\text{ACT}_1 \xrightarrow{\tau}$$

$$\text{Alice}_1 \mid \text{Carol} \mid \text{Ted}_1 \xrightarrow{\tau}$$

$$\text{Alice}_2 \mid \text{Carol} \mid \text{Ted}_2 \xrightarrow{\tau}$$

$$\text{Alice}_3 \mid \text{Carol}_1 \mid \text{Ted}_2 \xrightarrow{\tau}$$

is called a *execution* of $\text{ACT}_1$
- 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 \( \text{ACT1} \), such as

\[
\text{ACT1} \xrightarrow{\tau} \\
\text{Alice1} | \text{Carol} | \text{Ted1} \xrightarrow{\tau} \\
\text{Alice2} | \text{Carol} | \text{Ted2} \xrightarrow{\tau} \\
\text{Alice3} | \text{Carol1} | \text{Ted2} \xrightarrow{\tau} \\
\text{ACT1}
\]

is called a \textit{execution} of \( \text{ACT1} \).

- 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

\[
\text{ACT1} \xrightarrow{\tau} \\
\text{Alice}_1 \mid \text{Carol} \mid \text{Ted}_1 \xrightarrow{\tau} \\
\text{Alice}_2 \mid \text{Carol} \mid \text{Ted}_2 \xrightarrow{\tau} \\
\text{Alice}_3 \mid \text{Carol}_1 \mid \text{Ted}_2 \xrightarrow{\tau} \\
\text{ACT1}
\]

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

Transition records might be assembled into consensus blocks as follows:

1. Alice treats Ted
2. Ted acknowledges treatment
3. Alice bills Carol
4. Carol pays Alice
5. Ted alerts Carol of dose receipt
Distributed ledgers as carriers of contract executions

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:

1. **Transition Record**
   - **Begin State**: Alice1 | Carol | Ted1
   - **Transition**: \( \tau \)
   - **End State**: Alice | Carol | Ted1
   - **Previous Record**: NIL
   - **Signatures**: signed by Ted

2. **Signature Block**
   - **Transition Record**
     - **Begin State**: Alice2 | Carol | Ted2
     - **Transition**: \( \tau \)
     - **End State**: Alice3 | Carol1 | Ted2
     - **Previous Record**: hashp
     - **Signatures**: signed by Alice
   - **Previous Block**: NIL
     - **Signatures**: signed by Alice

3. **Signature Block**
   - **Transition Record**
     - **Begin State**: Alice3 | Carol1 | Ted2
     - **Transition**: \( \tau \)
     - **End State**: Alice3 | Carol1 | Ted2
     - **Previous Record**: hashp
     - **Signatures**: signed by Carol
   - **Previous Block**: hashp
     - **Signatures**: signed by Carol

4. **Signature Block**
   - **Transition Record**
     - **Begin State**: Alice3 | Carol3 | Ted
     - **Transition**: \( \tau \)
     - **End State**: ACT1
     - **Previous Record**: hashp
     - **Signatures**: signed by Carol
   - **Previous Block**: hashp
     - **Signatures**: signed by Alice

Alice treats Ted
Ted acknowledges treatment
Alice bills Carol
Ted alerts Carol of dose receipt
Carol pays Alice
Transition records might be assembled into consensus blocks as follows:

- **Transition Record**
  - **Begin State**: Alice | Carol | Ted1
  - **Transition**: \( \tau \)
  - **End State**: Alice | Carol | Ted1
  - **Previous Record**: NIL
  - **Signatures**: signed by Ted

- **Transition Record**
  - **Begin State**: Alice | Carol | Ted1
  - **Transition**: \( \tau \)
  - **End State**: Alice | Carol | Ted2
  - **Previous Record**: hashp
  - **Signatures**: signed by Alice

- **Transition Record**
  - **Begin State**: Alice | Carol | Ted2
  - **Transition**: \( \tau \)
  - **End State**: Alice | Carol1 | Ted2
  - **Previous Record**: hashp
  - **Signatures**: signed by Carol

- **Transition Record**
  - **Begin State**: Alice | Carol1 | Ted2
  - **Transition**: \( \tau \)
  - **End State**: Alice3 | Carol3 | Ted
  - **Previous Record**: hashp
  - **Signatures**: signed by Carol

**Actions**:
- Alice treats Ted
- Ted acknowledges treatment
- Ted alerts Carol of dose receipt
- Carol pays Alice
- Alice bills Carol
Distributed ledgers as carriers of contract executions

Transition records might be assembled into consensus blocks as follows
Distributed ledgers as carriers of contract executions

Transition records might be assembled into consensus blocks as follows:

Alice treats Ted

Alice bills Carol

Ted acknowledges treatment

Ted alerts Carol of dose receipt

Carol pays Alice

Carol pays Alice
Distributed ledgers as carriers of contract executions blocks

Transition records might be assembled into consensus blocks as follows

Consensus Block

Transition Record
Begin State
ACT1
Transition
τ
End State
Alice1 | Carol | Ted1
Previous Record
NIL
Signatures
signed by Ted

Transition Record
Begin State
Alice1 | Carol | Ted1
Transition
τ
End State
Alice2 | Carol | Ted2
Previous Record
hashp
Signatures
signed by Alice

Previous Block
NIL
Signatures
signed by Alice

Alice treats Ted

Consensus Block

Transition Record
Begin State
Alice2 | Carol | Ted2
Transition
τ
End State
Alice3 | Carol1 | Ted2
Previous Record
hashp
Signatures
signed by Carol

Previous Block
hashp
Signatures
signed by Carol

Alice bills Carol

Consensus Block

Transition Record
Begin State
Alice3 | Carol1 | Ted2
Transition
τ
End State
Alice3 | Carol3 | Ted
Previous Record
hashp
Signatures
signed by Carol

Previous Block
hashp
Signatures
signed by Alice

Ted acknowledges treatment

Ted alerts Carol of dose receipt

Carol pays Alice

17/21
Distributed ledgers as carriers of contract executions

Transition records might be assembled into consensus blocks as follows:

- Alice treats Ted
- Ted acknowledges treatment
- Alice bills Carol
- Ted alerts Carol of dose receipt
- Carol pays Alice

Consensus Block

Transition Record

<table>
<thead>
<tr>
<th>Begin State</th>
<th>Transition</th>
<th>End State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice1</td>
<td>τ</td>
<td>Alice2</td>
</tr>
<tr>
<td>Alice2</td>
<td>τ</td>
<td>Alice3</td>
</tr>
<tr>
<td>Alice3</td>
<td>τ</td>
<td>ACT1</td>
</tr>
</tbody>
</table>

Previous Record

| hashp |

Signatures

signed by Alice

Consensus Block

Transition Record

<table>
<thead>
<tr>
<th>Begin State</th>
<th>Transition</th>
<th>End State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice2</td>
<td>τ</td>
<td>Alice3</td>
</tr>
<tr>
<td>Alice3</td>
<td>τ</td>
<td>Alice3</td>
</tr>
</tbody>
</table>

Previous Record

| hashp |

Signatures

signed by Alice

Consensus Block

Transition Record

<table>
<thead>
<tr>
<th>Begin State</th>
<th>Transition</th>
<th>End State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice3</td>
<td>τ</td>
<td>Alice3</td>
</tr>
<tr>
<td>Alice3</td>
<td>τ</td>
<td>ACT1</td>
</tr>
</tbody>
</table>

Previous Record

| hashp |

Signatures

signed by Alice
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

\[ \text{It is not the case that Carol pays Alice without Ted's having been previously treated} \]
\[ \vdash \text{ACT1: } \neg (P \rightarrow \text{dose}) \langle \text{pay} \rangle \text{true} \]

**Liveness**  Something good eventually happens

\[ \text{Whenever Alice treats Ted, Carol eventually pays Alice} \]
\[ \vdash \text{ACT1: } [\text{dose}] \forall \text{E} \langle \text{pay} \rangle \text{true} \]

**Fairness**  Something good eventually happens to everyone

\[ \text{No matter what else Alice may be doing, she eventually treats Ted} \]
\[ \vdash \text{Alice: } \forall P \langle \text{dose} \rangle \text{true} \]
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

\[ \neg (P \land \text{dose}) \langle \text{pay} \rangle \text{true} \]

**Liveness**  Something good eventually happens

\[ \langle \text{dose} \rangle A \langle \text{pay} \rangle \text{true} \]

**Fairness**  Something good eventually happens to everyone

\[ \neg (P \land \text{dose}) \langle \text{pay} \rangle \text{true} \]
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*
\[ \neg (P - \text{dose})(\text{pay}) \text{true} \]

**Liveness**  Something good eventually happens
*Whenever Alice treats Ted, Carol eventually pays Alice*
\[ \text{dose} \text{AE}(\text{pay}) \text{true} \]

**Fairness**  Something good eventually happens to everyone
*No matter what else Alice may be doing, she eventually treats Ted*
\[ \text{AP}(\text{dose}) \text{true} \]
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

\[\text{It is not the case that Carol pays Alice without Ted's having been previously treated} \]
\[\vdash \text{ACT1: } \neg (P \rightarrow \text{dose}) \langle \text{pay} \rangle \text{true} \]

**Liveness** Something good eventually happens

\[\text{Whenever Alice treats Ted, Carol eventually pays Alice} \]
\[\vdash \text{ACT1: } \text{dose} \ [A \exists \langle \text{pay} \rangle ] \text{true} \]

**Fairness** Something good eventually happens to everyone

\[\text{No matter what else Alice may be doing, she eventually treats Ted} \]
\[\vdash \text{Alice: } A P \langle \text{dose} \rangle \text{true} \]
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

\[ \text{It is not the case that Carol pays Alice without Ted’s having been previously treated} \]

\[ \vdash \text{ACT1}: \neg (P - \text{dose}) \langle \text{pay} \rangle \text{true} \]

**Liveness**  Something good eventually happens

\[ \text{Whenever Alice treats Ted, Carol eventually pays Alice} \]

\[ \vdash \text{ACT1}: [\text{dose}] \text{AE} \langle \text{pay} \rangle \text{true} \]

**Fairness**  Something good eventually happens to everyone

\[ \text{No matter what else Alice may be doing, she eventually treats Ted} \]

\[ \vdash \text{Alice}: \text{AP} \langle \text{dose} \rangle \text{true} \]
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*

\[ \vdash \text{ACT1}: \neg (P - \text{dose})(\text{pay}) \text{true} \]

**Liveness** Something good eventually happens

*Whenever Alice treats Ted, Carol eventually pays Alice*

\[ \vdash \text{ACT1}: [\text{dose}]A \text{E}(\text{pay}) \text{true} \]

**Fairness** Something good eventually happens to everyone

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

\[ \vdash \text{Alice}: A \text{P}(\text{dose}) \text{true} \]
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
  $\vdash \text{ACT1: } \text{A}'0AC179DB.\text{End\ State}' \in \lambda'$

- The criteria by which it is discerned that an execution is a path in an LTS is a judgment. 
  Extracted from the ledger, $x_0 \xrightarrow{\alpha_1} x_1 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_n} x_n$
  $\vdash \text{ACT1: } \langle\alpha_1\rangle\langle\alpha_2\rangle\ldots\langle\alpha_n\rangle\text{true}$

- That such a trace is encoded on a blockchain is a judgment.
  $\vdash \text{ACT1: } A \wedge_{1 \leq i \leq n} \text{Transition\ Record}(i).\text{Transition} = \alpha_i$

- 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

  \[\text{ACT1: } 0AC179DB.\text{End State} \in \mathcal{X}\]

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

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

  \[\text{ACT1: } \langle \alpha_1 \rangle \langle \alpha_2 \rangle \cdots \langle \alpha_n \rangle \text{true}\]

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

  \[\text{ACT1: } \bigwedge_{1 \leq j \leq n} \text{Transition Record}(j).\text{Transition} = \alpha_j\]

- 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

\[
\text{The value in the End State field of the transition record pointed to by 0AC179DB is a state of the LTS}
\]
\[
\text{\[\text{ACT1}: \forall 0AC179DB.\text{End State} \in \mathcal{X}\]}
\]

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

\[
\text{Extracted from the ledger, } x_0 \xrightarrow{\alpha_1} x_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} x_n
\]
\[
\text{\[\text{ACT1}: \langle\alpha_1\rangle\langle\alpha_2\rangle\cdots\langle\alpha_n\rangle\text{true}\]}
\]

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

\[
\text{\[\text{ACT1}: \forall 1 \leq i \leq n \text{ Transition.Record}(i).\text{Transition} = \alpha_i\]}
\]

- 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*

  $\text{ACT1: } \forall 0AC179DB.\text{End State} \in \mathcal{X}$

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

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

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

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

  $\text{ACT1: } \forall 1 \leq i \leq n \text{Transition.Record}(i).\text{Transition} = \alpha_i$

- 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
  \[ \text{ACT1: } \forall '0AC179DB.\text{End\_State} \in \mathcal{X} \]

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

  Extracted from the ledger, \[ x_0 \xrightarrow{\alpha_1} x_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} x_n \]
  \[ \text{ACT1: } \langle \alpha_1 \rangle \langle \alpha_2 \rangle \cdots \langle \alpha_n \rangle \text{true} \]

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

  \[ \text{ACT1: } \exists \bigwedge_{1 \leq i \leq n} \text{Transition\_Record}(i).\text{Transition} = \alpha_i \]

- 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

  \[ \text{The value in the End State field of the transition record pointed to by 0AC179DB is a state of the LTS} \]

  \[ \vdash \text{ACT1: } \forall\text{0AC179DB.End State} \in \mathcal{X} \]

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

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

  \[ \vdash \text{ACT1: } \langle \alpha_1 \rangle \langle \alpha_2 \rangle \cdots \langle \alpha_n \rangle \text{true} \]

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

  \[ \vdash \text{ACT1: } \exists \bigwedge_{1 \leq i \leq n} \text{Transition.Record}(i).\text{Transition} = \alpha_i \]

- 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, \( \langle \mathcal{X}, \mathcal{R}_a \subseteq \mathcal{X} \times \mathcal{X} \mid a \in \mathcal{A} \rangle \)
- 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 \( x_0 = \text{ACT1} \) and for \( 1 \leq i \leq n \), \( x_i \in \mathcal{X} \), \( \alpha_i \in \mathcal{A} \)

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

derivability judgments such as \( \vdash \text{ACT1}: \neg (\text{P} \rightarrow \text{dose}) \langle \text{pay} \rangle \text{true} \) 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, \( \langle \mathcal{X}, \{ R_a \subseteq \mathcal{X} \times \mathcal{X} \mid a \in A \} \rangle \)
- 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 \( x_0 = \text{ACT1} \) and for \( 1 \leq i \leq n, x_i \in \mathcal{X}, \alpha_i \in A \)
admissibility \( x_0 \xrightarrow{\alpha_1} x_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_n} x_n \) is, in fact, an execution of \( \text{ACT1} \)
derivability judgments such as \( \vdash \text{ACT1}: \neg (\text{P} \rightarrow \text{dose}) \langle \text{pay} \rangle \text{true} \) 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, 
  \( \langle \mathcal{X}, \{R_a \subseteq \mathcal{X} \times \mathcal{X} \mid a \in \mathcal{A}\} \rangle \)

- 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 \( x_0 = \text{ACT1} \) and for \( 1 \leq i \leq n \), \( x_i \in \mathcal{X} \), \( \alpha_i \in \mathcal{A} \)

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

  derivability judgments such as \( \vdash \text{ACT1}: \lnot (P \rightarrow \text{dose}) \langle \text{pay} \rangle \text{true} \) 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, \( \langle X, \{ R_a \subseteq X \times X | a \in A \} \rangle \)
- 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 \( x_0 = \text{ACT1} \) and for \( 1 \leq i \leq n, x_i \in X, \alpha_i \in A \)

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

derivability judgments such as \( \vdash \text{ACT1}: \neg (P \rightarrow \text{dose}) \langle \text{pay} \rangle \text{true} \) 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, 
  \( \langle \mathcal{X}, \{ \mathcal{R}_a \subseteq \mathcal{X} \times \mathcal{X} | a \in A \} \rangle \)
- 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** \( x_0 = \text{ACT1} \) and for \( 1 \leq i \leq n \), \( x_i \in \mathcal{X}, \alpha_i \in A \)

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

**derivability** judgments such as \( \vdash \text{ACT1}: \neg (\text{P} \rightarrow \text{dose}) \langle \text{pay} \rangle \text{true} \) 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, 
  \( \langle \mathcal{X}, \{ R_a \subseteq \mathcal{X} \times \mathcal{X} | a \in A \} \rangle \)
- 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**  \( x_0 = \text{ACT1} \) and for \( 1 \leq i \leq n \), \( x_i \in \mathcal{X}, \alpha_i \in A \)

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

**derivability**  judgments such as \( \vdash \text{ACT1}: \neg(\overline{P \rightarrow \text{dose}}) \langle \text{pay} \rangle \text{true} \)

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, \( \langle \mathcal{X}, \{ R_a \subseteq \mathcal{X} \times \mathcal{X} | a \in \mathcal{A} \} \rangle \)
- 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** \( x_0 = \text{ACT1} \) and for \( 1 \leq i \leq n \), \( x_i \in \mathcal{X}, \alpha_i \in \mathcal{A} \)

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

**derivability** judgments such as \( \vdash \text{ACT1}: \neg(\neg \text{dose}) \langle \text{pay} \rangle \text{true} \) 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

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

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