Holistic Specifications for Robust Code

Sophia Drossopoulou
Imperial College London

based on prior work with James Noble (VU Wellington), Toby Murray (Uni Melbourne), Mark Miller (Agorics), and Susan Eisenbach, Shupeng Loh and Emil Klasan (Imperial)
Today

• Traditional Specifications do not adequately address Robustness

• Holistic Specifications — Summary and by Example

• Holistic Specification Semantics
Today

- Traditional Specifications do not adequately address Robustness
- Holistic Specifications — Summary and Examples
- Holistic Specification Semantics
Traditional Specification Languages do not adequately address robustness considerations
Traditional Specification Languages do not adequately address robustness considerations

Traditional Specs
Traditional Specification Languages do not adequately address robustness considerations

Traditional Specs

• designed for *closed* world

• pre- and post condition per function; *sufficient* conditions for some action/effect

• *explicit* about each individual function, and *implicit* about emergent behaviour
Traditional Specification Languages do not adequately address robustness considerations

Traditional Specs

- designed for *closed* world

- pre- and post condition per function; *sufficient* conditions for some action/effect

- *explicit* about each individual function, and *implicit* about emergent behaviour

Robustness considerations
Traditional Specifications Languages do not adequately address robustness considerations

Traditional Specs

- designed for closed world
- pre- and post condition per function; sufficient conditions for some action/effect
- explicit about each individual function, and implicit about emergent behaviour

Robustness considerations

- concerned with open world
- necessary conditions for some action/effect
- explicit about emergent behaviour
Today

• Traditional Specifications do not adequately address Robustness

• Holistic Specifications — Summary Examples

• Holistic Specification Semantics
Holistic Assertions — summary
Holistic Assertions — summary

e ::= this | x | e.fld | ...
Holistic Assertions — summary

e ::= this | x | e.fld | ...

A ::= e>e | e=e | ...

Holistic Assertions — summary

\[ e ::= \text{this} \mid x \mid e.\text{fld} \mid \ldots \]

\[ A ::= e > e \mid e = e \mid \ldots \]

\[ \mid A \rightarrow A \mid A \land A \mid \exists x. A \mid \ldots \]
Holistic Assertions — summary

e ::= this | x | e.fld | ...

A ::= e>e | e=e | ...
   | A → A | A ∧ A | ∃x. A | ...
   | Access(e,e')
Holistic Assertions — summary

e ::= this | x | e.fld | ...
A ::= e>e | e=e | ...
   | A → A | A ∧ A | ∃x. A | ...
   | Access(e,e')
   | Changes(e)
Holistic Assertions — summary

\[ e ::= \text{this} | x | e.fld | \ldots \]

\[ A ::= e>e | e=e | \ldots \]

| \( A \rightarrow A \) | \( A \land A \) | \( \exists x. A \) | \( \ldots \) |
| \( \text{Access}(e,e') \)
| \( \text{Changes}(e) \)
| \( \text{Will}(A) \) | \( \text{Was}(A) \) |
Holistic Assertions — summary

\[ e ::= \text{this} | x | e.fld | \ldots \]

\[ A ::= e>e | e=e | \ldots | A \rightarrow A | A \land A | \exists x. A | \ldots | \textbf{Access}(e,e') | \textbf{Changes}(e) | \textbf{Will}(A) | \textbf{Was}(A) | A \text{ in } S \]
Holistic Assertions — summary

\[ e ::= \text{this} \mid x \mid e.fld \mid \ldots \]

\[ A ::= e>e \mid e=e \mid \ldots \]

\[ \mid A \to A \mid A \land A \mid \exists x. A \mid \ldots \]

\[ \mid \text{Access}(e,e') \]

\[ \mid \text{Changes}(e) \]

\[ \mid \text{Will}(A) \mid \text{Was}(A) \]

\[ \mid A \text{ in } S \]

\[ \mid x.\text{Calls}(y,m,z_1,..z_n) \]
Holistic Assertions — summary

e ::= this | x | e.fld | ...

A ::= e>e | e=e | ...
   | A → A | A ∧ A | ∃x. A | ...
   | Access(e,e')
   | Changes(e)
   | Will(A) | Was(A)

| A in S
| x. Calls(y,m,z1,..zn)
| x obeys A
Holistic Assertions — summary

\[ e ::= \text{this} \mid x \mid e.fld \mid \ldots \]

\[ A ::= e > e \mid e = e \mid \ldots \]

\[ \mid A \rightarrow A \mid A \land A \mid \exists x. A \mid \ldots \]

\mid \textbf{Access}(e,e') \mid \textbf{Changes}(e) \mid \textbf{Will}(A) \mid \textbf{Was}(A) \mid A \textbf{ in } S \mid x.\textbf{Calls}(y,m,z_1,\ldots,z_n) \mid x \textbf{ obeys } A \mid \textbf{trust} \]

permission

authority

time

space

control

trust
Holistic Assertions — examples

- ERC20
- DAO
- DOM attenuation
- Bank & Account
- Escrow
Example 1: ERC20

a popular standard for initial coin offerings. (https://theethereum.wiki/w/index.php/ERC20_Token_Standard); allows clients to buy and transfer tokens, and to designate other clients to transfer on their behalf.

In particular, a client may call
- transfer: transfer some of her tokens to another clients,
- approve: authorise another client to transfer some of her tokens on her behalf.
- transferFrom: cause another client’s tokens to be transferred

Moreover, ERC20 keeps for each client
- balance the number of tokens she owns
classical specs - Hoare triples
classical specs - Hoare triples
classical specs - Hoare triples

\[ A \{ x \texttt{ calls } y.f(args) \} \]
classical specs - Hoare triples

\[ A \{ x \text{ calls } y.f(\text{args}) \} A' \]
If $A$ holds, and $x$ calls $y.f(args)$, then, $A'$ holds after the call.
classical specs - Hoare triples

If $A$ holds, and $x$ calls $y.f(\text{args})$, then, $A'$ holds after the call
classical specs - Hoare triples

If $A$ holds, and $x$ calls $y.f(args)$, then, $A'$ holds after the call.
ERC20 classical spec - transfer
ERC20 classical spec - transfer

For any ERC20 contract $e$, and different clients $c_1$, $c_2$. 
ERC20 classical spec - transfer

For any ERC20 contract $e$, and different clients $c_1$, $c_2$.

$c_1$’s balance is larger than $m$. 

For any ERC20 contract e, and different clients c1, c2.

c1’s balance is larger than m.

{ c1 calls e.transfer(c2,m) }
For any ERC20 contract \( e \), and different clients \( c_1, c_2 \).

\( c_1 \)'s balance is larger than \( m \).

\{ \( c_1 \) calls \( e . \) transfer(\( c_2, m \)) \}

\( c_1 \)'s balance decreases by \( m \), and \( c_2 \)'s balance increases by \( m \).
ERC20 classical spec - transfer

For any ERC20 contract $e$, and different clients $c_1$, $c_2$.

$c_1$’s balance is larger than $m$.

\[
\{ \text{c1 calls e.transfer(c2,m)} \}
\]

$c_1$’s balance decreases by $m$, and $c_2$’s balance increases by $m$.

\[
e: \text{ERC20} \land \ this = c_1 \neq c_2 \land e.\text{balance}(c_1) > m
\]
\[
\{ e.\text{transfer}(c_2,m) \}
\]
\[
e.\text{balance}(c_1) = e.\text{balance}(c_1)_{\text{pre}} - m \land e.\text{balance}(c_2) = e.\text{balance}(c_2)_{\text{pre}} + m
\]
ERC20 classical spec - transfer

For any ERC20 contract $e$, and different clients $c_1$, $c_2$.

$c_1$’s balance is larger than $m$.

\[
\{ \text{$c_1$ calls $e$.transfer($c_2,m$)} \}\]

$c_1$’s balance decreases by $m$, and $c_2$’s balance increases by $m$.

precondition

\[
e : \text{ERC20} \land \ this = c_1 \neq c_2 \land e.\text{balance}(c_1) > m
\]

\[
\{ \ e.\text{transfer}(c_2,m) \ \}
\]

\[
e.\text{balance}(c_1) = e.\text{balance}(c_1)_{\text{pre}} - m \land e.\text{balance}(c_2) = e.\text{balance}(c_2)_{\text{pre}} + m
\]
ERC20 classical spec - transfer

For any ERC20 contract $e$, and different clients $c_1$, $c_2$.

$c_1$'s balance is larger than $m$.

\[
\{ c_1 \text{ calls } e.\text{transfer}(c_2, m) \} \\
\text{\textit{c1}}'s \text{ balance decreases by } m, \text{ and } \textit{c2}'s \text{ balance increases by } m.
\]

\[
e:ERC20 \land \text{this } = c_1 \neq c_2 \land e.\text{balance}(c_1) > m \\
\{ \text{e.transfer}(c_2, m) \} \\
e.\text{balance}(c_1) = e.\text{balance}(c_1)_{\text{pre}} - m \land e.\text{balance}(c_2) = e.\text{balance}(c_2)_{\text{pre}} + m
\]
ERC20 classical spec - transfer

For any ERC20 contract $e$, and different clients $c_1$, $c_2$.

c1’s balance is larger than $m$.

\[
\{ \text{c1 calls } e.\text{transfer}(c2,m) \} \\
\text{c1’s balance decreases by } m, \text{ and c2’s balance increases by } m.
\]

e:\text{ERC20} \wedge \text{this } = c_1 \neq c_2 \wedge e.\text{balance}(c1) > m

\[
\{ e.\text{transfer}(c2,m) \}
\]

$e.\text{balance}(c1) = e.\text{balance}(c1)_{pre} - m \wedge e.\text{balance}(c2) = e.\text{balance}(c2)_{pre} + m$
ERC20 classical spec - transfer

For any ERC20 contract $e$, and different clients $c_1$, $c_2$.

$c_1$’s balance is larger than $m$.

\[
\{ \text{c1 calls } e\text{.transfer}(c_2,m) \}\]

$c_1$’s balance decreases by $m$, and $c_2$’s balance increases by $m$.

\[
e:ERC20 \land \text{this } = c_1 \neq c_2 \land e\text{.balance}(c_1) > m
\]

\[
\{ e\text{.transfer}(c_2,m) \}
\]

\[
e\text{.balance}(c_1) = e\text{.balance}(c_1)_{\text{pre}} - m \land e\text{.balance}(c_2) = e\text{.balance}(c_2)_{\text{pre}} + m
\]
For any ERC20 contract `e`, and different clients `c1`, `c2`.

`c1`'s balance is larger than `m`.

\{ `c1` calls `e.transfer(c2, m)` \}

`c1`'s balance decreases by `m`, and `c2`'s balance increases by `m`.
What if $c1$’s balance not large enough?
What if \( c_1 \)'s balance not large enough?

\[
e: \text{ERC20} \land \, \text{this} = c \land \, e, \text{balance}(c_1) < m
\]

\[
\{ \, e.\text{transfer}(c_2, m) \, \}
\]

\[
\forall \, c, \, e, \text{balance}(c) = e, \text{balance}(c)_{\text{pre}}
\]
ERC20 classic spec - authorised transfer
ERC20 classic spec - authorised transfer

For any ERC20 contract $e$, and different clients $c_1$, $c_2$, $c_3$. 
For any ERC20 contract $e$, and different clients $c_1$, $c_2$, $c_3$. $c_1$ is authorised to spend at least $m$ on $c_2$'s behalf and
For any ERC20 contract $e$, and different clients $c_1$, $c_2$, $c_3$. $c_1$ is authorised to spend at least $m$ on $c_2$'s behalf and $c_2$'s balance is at least $m$
ERC20 classic spec - authorised transfer

For any ERC20 contract e, and different clients c1, c2, c3. c1 is authorised to spend at least m on c2’s behalf and c2’s balance is at least m

\[
\{ \text{c1 calls } e\text{.transferFrom(c2,c3,m)} \}
\]
For any ERC20 contract $e$, and different clients $c_1$, $c_2$, $c_3$. $c_1$ is authorised to spend at least $m$ on $c_2$’s behalf and $c_2$’s balance is at least $m$

\[
\{ \text{c1 calls } e.\text{transferFrom(c2,c3,m)} \}
\]

$c_2$’s balance decreases by $m$, and $c_3$’s balance increases by $m$. 

For any ERC20 contract $e$, and different clients $c_1$, $c_2$, $c_3$.

$c_1$ is authorised to spend at least $m$ on $c_2$'s behalf and

$c_2$'s balance is at least $m$

\[
\{ \text{c1 calls e.transferFrom(c2,c3,m)} \}
\]

$c_2$'s balance decreases by $m$, and $c_3$'s balance increases by $m$.

\[
e:\text{ERC20} \land \text{this = c1}\neq c2\neq c3\neq c1 \land \\
e.\text{Authorized}(c1,c2,m') \land m'\geq m \\
e.\text{balance}(c1)\geq m \\
\{ e.\text{transferFrom(c2,c3,m)} \} \\
\]

$e.\text{balance}(c1) = e.\text{balance}(c1)_{\text{pre}} - m \land \\
\]

$e.\text{balance}(c2) = e.\text{balance}(c2)_{\text{pre}} + m \land \\
\]

$e.\text{Authorized}(c1,c2,m'-m)$
ERC20 classic spec - authorised transfer - 2
What if $c_1$ is not authorised, or $c_1$’s authorisation is insufficient, or $c_2$ has insufficient tokens?
What if \( c_1 \) is not authorised, or \( c_1' \)'s authorisation is insufficient, or \( c_2 \) has insufficient tokens?

\[
e: \text{ERC20} \land \text{this} = c_1 \neq c_2 \neq c_3 \neq c_1 \land (\neg e.\text{Authorized}(c_1,c_2,m) \lor e.\text{Authorized}(c_1,c_2,m') \land m' < m \lor e.\text{balance}(c_1) < m) \}
\{ e.\text{transferFrom}(c_1',m) \}
\forall c. e.\text{balance}(c) = e.\text{balance}(c)_{\text{pre}} \land 
\forall c,m. [ e.\text{Authorized}(c_1,c_2,m) \leftrightarrow e.\text{Authorized}(c_1,c_2,m)]
\]
ERC20 classic spec - authorising
ERC20 classic spec - authorising

e:ERC20 ∧ this = c1
    { e.approve(c2,m) }
    e.Authorized(c1,c2,m)
ERC20 classical spec

e:ERC20 ∧ this = c1≠c2 ∧ e.balance(c1) > m
    { e.transfer(c2,m) }
    e.balance(c1) = e.balance(c1)_{pre} - m ∧ e.balance(c2) = e.balance(c2)_{pre} + m
ERC20 classical spec

e:ERC20 ∧ this = c1≠c2 ∧ e.balance(c1) > m
{ e.transfer(c2, m) } e.balance(c1) = e.balance(c1)_{pre} - m ∧ e.balance(c2) = e.balance(c2)_{pre} + m

∀ c. e.balance(c) = e.balance(c)_{pre}
ERC20 classical spec

\[
e: \text{ERC20} \land \text{this} = c1 \neq c2 \land e.\text{balance}(c1) > m
\quad \{ \ e.\text{transfer}(c2,m) \ \}
\quad e.\text{balance}(c1) = e.\text{balance}(c1)_{\text{pre}} - m \land e.\text{balance}(c2) = e.\text{balance}(c2)_{\text{pre}} + m
\]

\[
e: \text{ERC20} \land \text{this} = c1 \land e.\text{balance}(c1) < m
\quad \{ \ e.\text{transfer}(c2,m) \ \}
\quad \forall c. \ e.\text{balance}(c) = e.\text{balance}(c)_{\text{pre}}
\]

\[
e: \text{ERC20} \land \text{this} = c1 \neq c2 \neq c3 \neq c1 \land
e.\text{Authorized}(c1,c2,m') \land m' \geq m \\
\land e.\text{balance}(c1) \geq m
\quad \{ \ e.\text{transferFrom}(c2,c3,m) \ \}
e.\text{balance}(c1) = e.\text{balance}(c1)_{\text{pre}} - m \land
e.\text{balance}(c2) = e.\text{balance}(c2)_{\text{pre}} + m \land e.\text{Authorized}(c1,c2,m'-m)
\]
ERC20   classical spec

\[
\begin{align*}
\text{e:ERC20} & \land \text{this} = c1 \neq c2 \land \text{e.balance}(c1) > m \\
& \{ \text{e.transfer}(c2, m) \} \\
\text{e.balance}(c1) = \text{e.balance}(c1)_{\text{pre}} - m & \land \text{e.balance}(c2) = \text{e.balance}(c2)_{\text{pre}} + m
\end{align*}
\]

\[
\begin{align*}
\text{e:ERC20} & \land \text{this} = c1 \land \text{e.balance}(c1) < m \\
& \{ \text{e.transfer}(c2, m) \} \\
\forall c. \text{e.balance}(c) = \text{e.balance}(c)_{\text{pre}}
\end{align*}
\]

\[
\begin{align*}
\text{e:ERC20} & \land \text{this} = c1 \neq c2 \neq c3 \neq c1 \land \\
& \text{e.Authorized}(c1, c2, m') \land m' \geq m \land \text{e.balance}(c1) \geq m \\
& \{ \text{e.transferFrom}(c2, c3, m) \} \\
\text{e.balance}(c1) = \text{e.balance}(c1)_{\text{pre}} - m & \land \\
\text{e.balance}(c2) = \text{e.balance}(c2)_{\text{pre}} + m & \land \text{e.Authorized}(c1, c2, m' - m)
\end{align*}
\]

\[
\begin{align*}
\text{e:ERC20} & \land \text{this} = c1 \neq c2 \neq c3 \neq c1 \land \\
& ( \neg \text{e.Authorized}(c1, c2, m) \lor \text{e.Authorized}(c1, c2, m') \land m' < m \\
& \lor \text{e.balance}(c1) < m ) \\
& \{ \text{e.transferFrom}(c2, c3, m) \} \\
\forall c. \text{e.balance}(c) = \text{e.balance}(c)_{\text{pre}} & \land \\
\forall c, m. [ \text{e.Authorized}(c1, c2, m) \leftrightarrow \text{e.Authorized}(c1, c2, m)]
\end{align*}
\]
ERC20 classical spec

\[\begin{align*}
e &: \text{ERC20} \land \text{this} = c_1 \neq c_2 \land e.\text{balance}(c_1) > m \\
& \quad \{ e.\text{transfer}(c_2, m) \} \\
e.\text{balance}(c_1) &= e.\text{balance}(c_1)_{\text{pre}} - m \land e.\text{balance}(c_2) = e.\text{balance}(c_2)_{\text{pre}} + m
\end{align*}\]

\[\begin{align*}
e &: \text{ERC20} \land \text{this} = c_1 \land e.\text{balance}(c_1) < m \\
& \quad \{ e.\text{transfer}(c_2, m) \} \\
\forall c. e.\text{balance}(c) &= e.\text{balance}(c)_{\text{pre}}
\end{align*}\]

\[\begin{align*}
e &: \text{ERC20} \land \text{this} = c_1 \neq c_2 \neq c_3 \neq c_1 \land \\
e.\text{Authorized}(c_1, c_2, m') \land m' \geq m \\
& \quad \land e.\text{balance}(c_1) \geq m \\
& \quad \{ e.\text{transferFrom}(c_2, c_3, m) \} \\
e.\text{balance}(c_1) &= e.\text{balance}(c_1)_{\text{pre}} - m \land \\
e.\text{balance}(c_2) &= e.\text{balance}(c_2)_{\text{pre}} + m \land e.\text{Authorized}(c_1, c_2, m' - m)
\end{align*}\]

\[\begin{align*}
e &: \text{ERC20} \land \text{this} = c_1 \neq c_2 \neq c_3 \neq c_1 \land \\
( \neg e.\text{Authorized}(c_1, c_2, m) \lor e.\text{Authorized}(c_1, c_2, m') \land m' < m \\
\lor e.\text{balance}(c_1) < m ) \\
& \quad \{ e.\text{transferFrom}(c_2, c_3, m) \} \\
\forall c. e.\text{balance}(c) &= e.\text{balance}(c)_{\text{pre}} \land \\
\forall c, m. [ e.\text{Authorized}(c_1, c_2, m) \leftrightarrow e.\text{Authorized}(c_1, c_2, m) ]
\end{align*}\]

\[\begin{align*}
e &: \text{ERC20} \land \text{this} = c_1 \{ e.\text{approve}(c_2, m) \} e.\text{Authorized}(c_1, c_2, m)
\end{align*}\]
ERC20 classical spec

\[\text{e:ERC20 } \land \text{ this } = c_1 \neq c_2 \land \text{ e.balance}(c_1) > m\]
\[
\begin{array}{l}
\{ \text{ e.transfer}(c_2, m) \} \\
\text{ e.balance}(c_1) = \text{ e.balance}(c_1)_{\text{pre}} - m \land \text{ e.balance}(c_2) = \text{ e.balance}(c_2)_{\text{pre}} + m
\end{array}
\]
ERC20 classical spec

e:ERC20  ∧  this = c1≠c2  ∧  e.balance(c1) > m
  {  e.transfer(c2,m)  }
  e.balance(c1) = e.balance(c1)_{pre} - m  ∧  e.balance(c2) = e.balance(c2)_{pre} + m

e:ERC20  ∧  this = c1≠c2≠c3≠c1  ∧  
  e.Authorized(c1,c2,m')  ∧  m'≥ m
  ∧  e.balance(c1)≥m
  {  e.transferFrom(c2,c3,m)  }
  e.balance(c1) = e.balance(c1)_{pre} - m  ∧
  e.balance(c2) = e.balance(c2)_{pre} + m  ∧  e.Authorized(c1,c2,m'-m)

\(∀ c.\)  e.balance(c) = e.balance(c)_{pre}

\(∀ c,m.\) [ e.Authorized(c1,c2,m) ↔ e.Authorized(c1,c2,m) ]

...  {  e.totalSupply()  }  ....

...  {  e.balanceOf(c)  }  ....
ERC20 classical spec

e:ERC20 ∧ this = c1≠c2 ∧ e.balance(c1) > m
    { e.transfer(c2,m) }

∀ c. e.balance(c) = e.balance(c)_{pre} ∧
∀ c,m. [ e.Authorized(c1,c2,m) ↔ e.Authorized(c1,c2,m) ]

... { e.transferFrom(c2,c3,m) }

∀ c, e.balance(c) = e.balance(c)_{pre} ∧
∀ c,m. [ e.Authorized(c1,c2,m) ↔ e.Authorized(c1,c2,m) ]
ERC20 classical spec

e:ERC20 \land \text{this} = c1 \neq c2 \land e.\text{balance}(c1) > m
\{ e.\text{transfer}(c2,m) \}
\quad e.\text{balance}(c1) = e.\text{balance}(c1)_{\text{pre}} - m \land e.\text{balance}(c2) = e.\text{balance}(c2)_{\text{pre}} + m

sufficient conditions for change of balance

e:ERC20 \land \text{this} = c1 \land e.\text{balance}(c1) < m
\{ e.\text{transfer}(c2,m) \}

\forall c. e.\text{balance}(c) = e.\text{balance}(c)_{\text{pre}}
ERC20 classical spec

\[\text{e:ERC20} \land \text{this} = c1 \neq c2 \land \text{e.balance}(c1) > m\]
\[
\{ \text{e.transfer}(c2,m) \}
\]
\[\text{e.balance}(c1) = \text{e.balance}(c1)_{\text{pre}} - m \land \text{e.balance}(c2) = \text{e.balance}(c2)_{\text{pre}} + m\]

\[\text{e:ERC20} \land \text{this} = c1 \land \text{e.balance}(c1) < m\]
\[
\{ \text{e.transfer}(c2,m) \}
\]

\[\forall \text{c. } \text{e.balance}(c) = \text{e.balance}(c)_{\text{pre}}\]

\[\text{e:ERC20} \land \text{this} = c1 \neq c2 \neq c3 \neq c1 \land \]
\[\text{e.Authorized}(c1,c2,m') \land m' \geq m\]
\[\land \text{e.balance}(c1) \geq m\]
\[
\{ \text{e.transferFrom}(c2,c3,m) \}
\]
\[\text{e.balance}(c1) = \text{e.balance}(c1)_{\text{pre}} - m \land \]
\[\text{e.balance}(c2) = \text{e.balance}(c2)_{\text{pre}} + m \land \text{e.Authorized}(c1,c2,m'-m)\]

\[\text{e:ERC20} \land \text{this} = c1 \neq c2 \neq c3 \neq c1 \land \]
\[\neg \text{e.Authorized}(c1,c2,m) \lor \text{e.Authorized}(c1,c2,m') \land m' < m\]
\[\lor \text{e.balance}(c1) < m\]
\[
\{ \text{e.transferFrom}(c2,c3,m) \}
\]
\[\forall \text{c. } \text{e.balance}(c) = \text{e.balance}(c)_{\text{pre}}\]
\[\forall \text{c,m. } [ \text{e.Authorized}(c1,c2,m) \leftrightarrow \text{e.Authorized}(c1,c2,m)]\]

\[\ldots \{ \text{e.totalSupply}() \} \ldots\]
\[\ldots \{ \text{e.allowanceOf}(c2) \} \ldots\]
ERC20 classical spec

Is that robust?
ERC20 classical spec

Is that robust?

a function that takes 0.5% from each account?
ERC20  classical spec

Is that robust?

a function that takes 0.5% from each account?

can authority increase?

... { e.totalSupply() } ...
... { e.balanceOf(c) } ...
... { e.allowanceOf(c2) } ...
Is that robust?

a “super-cleint,” authorised on all?

a function that takes 0.5% from each account?

can authority increase?
 ERC20 classical spec

Is that robust?

a “super-client,” authorised on all?

a function that takes 0.5% from each account?

can authority increase?
Is that robust?

I am worried about who/what can reduce my balance

can authority increase?

a “super-cleint,” authorised on all?

a function that takes 0.5% from each account?
holistic specs - invariants
holistic specs - invariants
holistic specs - invariants
holistic specs - invariants

• state
holistic specs - invariants

- state
- time
- space
- control
- permission
- authority
- (when/where do they hold)
holistic spec - reduce balance
holistic spec - reduce balance

∀ e:ERC20. ∀ c1: Client. ∀ m: Nat.

[ e.balance(c1) = Was(e.balance(c1)) - m]
∀ e:ERC20. ∀ c1: Client. ∀ m: Nat.
\[ e.balance(c1) = \text{Was}(e.balance(c1)) - m \]
holistic spec - reduce balance

\( \forall e : \text{ERC20}. \ \forall c1 : \text{Client}. \ \forall m : \text{Nat}. \)
\[
\left[ \ \text{e.balance}(c1) = \text{Was}(\text{e.balance}(c1)) - m \right]
\]
\[
\rightarrow
\]
\[
( \exists c2, c3 : \text{Client}. \text{Was} ( c1. \text{Calls}(e, \text{transfer}, c2, m)) )
\]
∀ e:ERC20. ∀ c1: Client. ∀ m: Nat.
[ e.balance(c1) = Was(e.balance(c1)) - m

→
( ∃ c2, c3: Client.
  Was( c1.Calls(e, transfer, c2, m) )
  ∧
  Was( e.Authorized(c1, c2, m) ∧
  c2.Calls(e, transferFrom, c1, c3, m) ) ]
holistic spec - reduce balance

∀ e:ERC20. ∀ c1: Client. ∀ m: Nat.
[ e.balance(c1) = Was(e.balance(c1)) - m

→
( ∃c2,c3: Client.
  Was( c1.Calls(e,transfer,c2,m) )
  ∨
  Was( e.Authorized(c1,c2,m) ∧
  c2.Calls(e,transferFrom,c1,c3,m) ) ) ]

This says: A client’s balance decreases only if that client, or somebody authorised by that client, made a payment.
holistic spec - reduce balance

∀ e:ERC20. ∀ c1: Client. ∀ m: Nat.

[ e.balance(c1) = Was(e.balance(c1)) - m

→

( ∃c2,c3: Client.

Was( c1.Calls(e, transfer, c2, m) ) )

∧

Was( e.Authorized(c1, c2, m) ∧

  c2.Calls(e, transferFrom, c1, c3, m) ) ) ]

This says: A client’s balance decreases only if that client, or somebody authorised by that client, made a payment.
This says: A client’s balance decreases *only* if that client, or somebody authorised by that client, made a payment.
holistic spec - authority
holistic spec - authority

e. \text{Authorized}(c1, c2, m) \iff

c2 is authorised by \textbf{c1} for \textbf{m} \ \text{iff}
e. $\text{Authorized}(c_1,c_2,m) \iff \text{Was}(c_1.\text{Calls}(e,\text{approve},c_2,m))$

c2 is authorised by c1 for m iff

in previous step c1 informed e that it authorised c2 for m
holistic spec - authority

e. Authorized(c1, c2, m) ≜

Was( c1.Calls(e, approve, c2, m))
∨
Was( e. Authorized(c1, c2, m+m') ∧ c2.Calls(e, transferFrom, c1, _, m'))

c2 is authorised by c1 for m iff

in previous step c1 informed e that it authorised c2 for m
or
in previous step c2 was authorised for m+m' and spent m' for c1
holistic spec - authority

e. Authorized(c1, c2, m) ≡

  Was( c1.Calls(e, approve, c2, m) )  
    ∨  
  Was( e. Authorized(c1, c2, m+m’) ∧ c2.Calls(e, transferFrom, c1, _, m’) )  
    ∨  
  Was( e. Authorized(c1, c2, m) ∧ ¬ c2.Calls(e, transferFrom, c1, _, _) )


c2 is authorised by c1 for m iff

  in previous step c1 informed e that it authorised c2 for m
  or
  in previous step c2 was authorised for m+m’ and spent m’ for c1
  or
  in previous step c2 was authorised for m and did not spend c1
holistic spec - authority

e.\text{Authorized}(c1, c2, m) \equiv

\begin{align*}
&\text{Was}(c1.\text{Calls}(e, \text{approve}, c2, m)) \\
&\quad \lor \\
&\text{Was}(e.\text{Authorized}(c1, c2, m+m') \land c2.\text{Calls}(e, \text{transferFrom}, c1, _, m')) \\
&\quad \lor \\
&\text{Was}(e.\text{Authorized}(c1, c2, m) \land \neg c2.\text{Calls}(e, \text{transferFrom}, c1, _, _, _))
\end{align*}

c2 is authorised by c1 for m \iff

in previous step c1 informed e that it authorised c2 for m
or
in previous step c2 was authorised for m+m' and spent m' for c1
or
in previous step c2 was authorised for m and did not spend c1
holistic spec - authority

e. Authorized(c1, c2, m) ≜

Was(c1.Calls(e, approve, c2, m))

∨

Was(e Authorized(c1, c2, m+m') ∧ c2.Calls(e, transferFrom, c1, _, m'))

∨

Was(e Authorized(c1, c2, m) ∧ ¬ c2.Calls(e, transferFrom, c1, _, _, _))

c2 is authorised by c1 for m iff

in previous step c1 informed e that it authorised c2 for m

or

in previous step c2 was authorised for m+m’ and spent m’ for c1

or

in previous step c2 was authorised for m and did not spend c1
classical vs holistic
classical vs holistic

\[ e:ERC20 \land e.\text{balance}(cl) > m \land e.\text{balance}(cl') = m' \land cl \neq cl' \]
\{ e.\text{transfer}(cl',m) \land \text{Caller}=cl \}

\[ e.\text{balance}(cl) = e.\text{balance}(cl)_{\text{pre}} - m \land e.\text{balance}(cl')_{\text{pre}} = m' + m \]

\[ e:ERC20 \land e.\text{balance}(cl) > m \land e.\text{balance}(cl') = m' \land cl \neq cl' \land \text{Authorized}(e, cl, cl'') \]
\{ e.\text{transferFrom}(cl',m) \land \text{Caller}=cl'' \}

\[ e.\text{balance}(cl) = e.\text{balance}(cl)_{\text{pre}} - m \land e.\text{balance}(cl')_{\text{pre}} = m' + m \]

\[ e:ERC20 \land e.\text{balance}(cl) > m \land e.\text{balance}(cl') = m' \]
\{ e.\text{allow}(cl') \land \text{Caller}=cl \} \land \text{Authorized}(e, cl, cl'') \]

Classical

- per function; *sufficient* conditions for some action/effect
- *explicit* about individual function, and *implicit* about emergent behaviour
classical vs holistic

\[
e: \text{ERC20} \land e.\text{balance}(cl) > m \land e.\text{balance}(cl') = m' \land cl \neq cl'
\]
\[
\{ \text{e.transfer}(cl, m) \land \text{Caller}=cl \}
\]
\[
e.\text{balance}(cl) = e.\text{balance}(cl)_{\text{pre}} - m \land e.\text{balance}(cl')_{\text{pre}} = m' + m
\]

\[
e: \text{ERC20} \land e.\text{balance}(cl) > m \land e.\text{balance}(cl') = m' \land cl \neq cl'
\]
\[
\land \text{Authorized}(e, cl, cl'')
\]
\[
\{ \text{e.transferFrom}(cl', m) \land \text{Caller}=cl'' \}
\]
\[
e.\text{balance}(cl) = e.\text{balance}(cl)_{\text{pre}} - m \land e.\text{balance}(cl')_{\text{pre}} = m' + m
\]

\[
e: \text{ERC20} \land e.\text{balance}(cl) > m \land e.\text{balance}(cl') = m'
\]
\[
\{ \text{e.allow}(cl') \land \text{Caller}=cl \}
\]
\[
\text{Authorized}(e, cl, cl'')
\]

\[\ldots \ldots\]
\[\text{another 7 specs} \quad \ldots \quad \ldots\]

Classical

- per function; \textit{sufficient} conditions for some action/effect
- \textit{explicit} about individual function, and \textit{implicit} about emergent behaviour

20
classical vs holistic

**Classical**

- per function; *sufficient* conditions for some action/effect
- *explicit* about individual function, and *implicit* about emergent behaviour

**Holistic**

- *necessary* conditions for some action/effect
- *explicit* about emergent behaviour

∀ e:ERC20. ∀ cl: Client.

\[
\[ e.\text{balance}(cl) = \text{Was}(e.\text{balance}(cl)) - m \] \\
\rightarrow
\]

\( (\exists cl', cl'': \text{Client}. \text{Was} ( cl'.\text{Calls}(e.\text{transfer}(cl',m))) \)

\&

\text{Was} (\text{Authorized}(e, cl, cl'') \& cl''.\text{Calls}(e.\text{transferFrom}(cl, cl', m)))

Authorized(cl, cl') \equiv \exists m: \text{Nat. Was}*( cl'.\text{Calls}(e.\text{approve}(cl', m)))
classical vs holistic

Classical

- per function; :sufficient conditions for some action/effect
- explicit about individual function, and implicit about emergent behaviour

Holistic

- necessary conditions for some action/effect
- explicit about emergent behaviour

∀ e:ERC20. ∀ cl: Client.
[ e.balance(cl) = Was(e.balance(cl)) - m ]
→
( ∃ cl’, cl’': Client.
  Was ( cl.Calls(e.transfer(cl’,m)) )
  ∨
  Was ( Authorized(e, cl, cl’’ ) ∧ cl’’.Calls( e.transferFrom(cl, cl’, m) ) ) ]

Authorized(cl, cl’) ≜ ∃ m: Nat. Was*( cl.Calls(e.approve(cl’,m)) )
Example 2: DAO simplified

DAO, a “hub that disperses funds”; (https://www.ethereum.org/dao).

... clients may contribute and retrieve funds:

- `payIn(m)` pays into DAO m on behalf of client
- `repay()` withdraws all moneys from DAO

**Vulnerability:** Through a buggy version of `repay()`, a client could re-enter the call and deplete all funds of the DAO.
classical spec

Assuming DAO keeps a directory of contributions, and require:
R1: directory is compatible with the amount of ether kept in the DAO, and
R2: that withdraw reduces the ether by that amount.
classical spec

Assuming DAO keeps a directory of contributions, and require:

R1: directory is compatible with the amount of ether kept in the DAO, and

R2: that withdraw reduces the ether by that amount.

R1: $\forall d:\text{DAO}. \quad d.\text{ether} = \sum_{cl \in \text{dom}(d.\text{directory})} d.\text{directory}(cl)$
Assuming DAO keeps a directory of contributions, and require:

R1: directory is compatible with the amount of ether kept in the DAO, and
R2: that withdraw reduces the ether by that amount.

R1: $\forall d:DAO. \; d.\text{ether} = \sum_{cl \in \text{dom}(d.\text{directory})} d.\text{directory}(cl)$

R2: $d:DAO \land n:\text{Nat} \land d.\text{directory}(cl)=n>0 \land \text{this}=cl$

\{ d.\text{repay}() \}

$d.\text{directory}(cl)=0 \land d.\text{Calls}(cl,send,n)$
classical spec

Assuming DAO keeps a directory of contributions, and require:

R1: directory is compatible with the amount of ether kept in the DAO, and
R2: that withdraw reduces the ether by that amount.

R1: ∀d:DAO. d.ether = ∑ cl ∈ dom(d.directory) d.directory(cl)

R2: d:DAO ∧ n:Nat ∧ d.directory(cl)=n>0 ∧ this=cl
    { d.repay() }
    d.directory(cl)=0 ∧ d.Calls(cl,send,n)

    d:DAO ∧ n:Nat ∧ d.directory(cl)=0 ∧ this=cl
    { d.repay() }

"nothing changes"
Assuming DAO keeps a directory of contributions, and require:
R1: directory is compatible with the amount of ether kept in the DAO, and
R2: that withdraw reduces the ether by that amount.

R1: $\forall d:DAO. \ d.\text{ether} = \sum_{cl \in \text{dom}(d.\text{directory})} d.\text{directory}(cl)$

R2: $d:DAO \land n:\text{Nat} \land d.\text{directory}(cl)=n>0 \land \text{this}=cl$

{ $d.\text{repay}()$ }

$d.\text{directory}(cl)=0 \land d.\text{Calls}(cl,\text{send},n)$

This spec avoids the vulnerability,

“nothing changes”

This spec avoids the vulnerability,
classical spec

Assuming DAO keeps a directory of contributions, and require:

R1: directory is compatible with the amount of ether kept in the DAO, and
R2: that withdraw reduces the ether by that amount.

R1: \( \forall d: \text{DAO}. \quad d.\text{ether} = \sum_{cl \in \text{dom}(d.\text{directory})} d.\text{directory}(cl) \)

R2: \( d: \text{DAO} \land n: \text{Nat} \land d.\text{directory}(cl)=n>0 \land \text{this}=cl \)

\{ \quad d.\text{repay()} \quad \}

\( d.\text{directory}(cl)=0 \land d.\text{Calls}(cl, send, n) \)

This spec avoids the vulnerability, 😊

provided the attack goes through the function repay. 😢
classical spec

Assuming DAO keeps a directory of contributions, and require:
R1: directory is compatible with the amount of ether kept in the DAO, and
R2: that withdraw reduces the ether by that amount.

R1: \( \forall d: \text{DAO}. \ d.\text{ether} = \sum_{cl \in \text{dom}(d.\text{directory})} d.\text{directory}(cl) \)

R2: \( d: \text{DAO} \land n: \text{Nat} \land d.\text{directory}(cl)=n>0 \land \text{this}=cl \)
    \{ d.\text{repay}() \}
    d.\text{directory}(cl)=0 \land d.\text{Calls}(cl,send,n)

    d: \text{DAO} \land n: \text{Nat} \land d.\text{directory}(cl)=0 \land \text{this}=cl
    \{ d.\text{repay}() \}

    “nothing changes”

This spec avoids the vulnerability, 😄

*provided* the attack goes through the function `repay`. 😢

To avoid the vulnerability in general, we need to inspect the specification of *all* the functions in the DAO.
classical spec

Assuming DAO keeps a directory of contributions, and require:

R1: directory is compatible with the amount of ether kept in the DAO, and
R2: that withdraw reduces the ether by that amount.

R1: \( \forall d : \text{DAO}. \quad d.\text{ether} = \sum_{c| \in \text{dom}(d.\text{directory})} d.\text{directory}(cl) \)

R2: \( d : \text{DAO} \wedge n : \text{Nat} \wedge d.\text{directory}(cl) = n > 0 \wedge \text{this} = cl \)
    \{ \quad d.\text{repay}() \quad \}
    d.\text{directory}(cl) = 0 \wedge d.\text{Calls}(cl, \text{send}, n)

    d : \text{DAO} \wedge n : \text{Nat} \wedge d.\text{directory}(cl) = 0 \wedge \text{this} = cl
    \{ \quad d.\text{repay}() \quad \}

"nothing changes"

This spec avoids the vulnerability, 😃

**provided** the attack goes through the function \text{repay}. 😢

To avoid the vulnerability in general, we need to inspect the specification of all the functions in the DAO. DAO - interface has nineteen functions. 😢
holistic

[ cl.Calls(d.repay()) ∧ d.Balance(cl) = n
→
  d.ether ≧ n ∧ Will(d.Calls(cl.send(n))) ]
This specification avoids the vulnerability, regardless of which function introduces it:
The DAO will always be able to repay all its customers.
This specification avoids the vulnerability, regardless of which function introduces it:
The DAO will always be able to repay all its customers.

\[ \forall \text{cl:External. } \forall \text{d:DAO. } \forall n: \text{Nat.} \]
\[ [ \text{cl.Calls(d.repay()) } \land \text{d.Balance(cl) = n} \]
\[ \rightarrow \]
\[ \text{d.ether} \geq n \land \text{Will(d.Calls(cl.send(n)))} ] \]
[ cl Calls(d repay() ) ∧ d Balance(cl) = n
→
  d.ether ≧ n ∧ Will(d Calls(cl send(n ))) ]

This specification avoids the vulnerability, regardless of which function introduces it:
The DAO will always be able to repay all its customers.

d Balance(cl) ≡ 0 if cl Calls(d initialize() )
This specification avoids the vulnerability, regardless of which function introduces it:
The DAO will always be able to repay all its customers.

\[
\forall \ cl: \text{External. } \forall \ d: \text{DAO. } \forall \ n: \text{Nat.} \\
[ \ \text{cl.} \text{Calls}(\ d. \text{repay}() \ ) \land \ d.\text{Balance}(\text{cl}) = n \ \\
\rightarrow \\
\quad \ d.\text{ether} \geq n \land \text{Will}(\ d.\text{Calls}(\text{cl-send}(n))) ]
\]

\d.\text{Balance}(\text{cl}) = 0 \quad \text{if } \text{cl.} \text{Calls}(\ d.\text{initialize}(),\ )
\m+m' \quad \text{if } \text{Was}(\ d.\text{Balance}(\text{cl}),\m) \land \text{cl.} \text{Calls}(\ d.\text{payIn}(\m'))
This specification avoids the vulnerability, regardless of which function introduces it:
The DAO will always be able to repay all its customers.

d.Balance(cl) =
0 if cl.Calls(d,initialize() )
m+m' if Was(d.Balance(cl),m) \land cl.Calls(d.payIn(m') )
0 if Was(cl.Calls(d.repayIn() )

[ cl.Calls(d.repay() ) \land d.Balance(cl) = n
→
d.ether ≧ n \land \text{Will}(d.Calls(cl.send(n ))) ]
This specification avoids the vulnerability, regardless of which function introduces it:
The DAO will always be able to repay all its customers.

$$\forall \text{cl:External. } \forall \text{d:DAO. } \forall \text{n:Nat.}$$
$$\left[ \text{cl.Calls(d.repay()) } \land \text{d.Balance(cl) = n} \rightarrow \right.$$  
$$\begin{align*}
d.\text{ether} &\geq n \land \text{Will(d.Calls(cl.send(n)))} 
\end{align*}$$

$$\text{d.Balance(cl)} = 0 \quad \text{if \ cl.Calls(d,initialize() )}$$
$$\text{m+m'} \quad \text{if \ Was(d.Balance(cl),m) } \land \text{cl.Calls(d.payIn(m') )}$$
$$0 \quad \text{if \ Was(cl.Calls(d.repayIn() )}$$
$$\text{Was(d.Balance(cl))} \quad \text{otherwise}$$
classical vs holistic

\[ \forall d: \text{DAO}. \quad d.\text{ether} = \sum_{cl \in \text{dom}(d.\text{directory})} d.\text{directory}(cl) \]

\[ d: \text{DAO} \land n: \text{Nat} \land d.\text{directory}(cl) = n > 0 \land \text{this} = cl \]
\{ d.\text{repay}() \}

\[ d.\text{directory}(cl) = 0 \land d.\text{Calls}(cl.\text{send}(n)) \]

\[ d: \text{DAO} \land n: \text{Nat} \land d.\text{directory}(cl) = 0 \land \text{this} = cl \]
\{ d.\text{repay}() \}

“nothing changes”
∀d:DAO. \( d.\text{ether} = \sum_{cl \in \text{dom}(d.\text{directory})} d.\text{directory}(cl) \)

\[
\begin{align*}
d:DAO & \land n:\text{Nat} \land d.\text{directory}(cl)=n>0 \land \text{this}=\text{cl} \\
& \{ d.\text{repay()} \} \\
d.\text{directory}(cl)=0 \land d.\text{Calls}(cl.\text{send}(n)) \\
d:DAO & \land n:\text{Nat} \land d.\text{directory}(cl)=0 \land \text{this}=\text{cl} \\
& \{ d.\text{repay()} \} \\
\end{align*}
\]
“nothing changes”

**Classical**

- per function; *sufficient* conditions for some action/effect
- *explicit* about individual function, and *implicit* about emergent behaviour
∀d:DAO. d.ether = \sum_{cl \in \text{dom}(d.\text{directory})} d.\text{directory}(cl)

\begin{align*}
d:DAO \land n:\text{Nat} \land d.\text{directory}(cl)=n>0 \land \text{this}=cl \\
{\{ \text{d.repay()} \}} \\
d.\text{directory}(cl)=0 \land d.\text{Calls}(cl.\text{send}(n))
\end{align*}

\begin{align*}
d:DAO \land n:\text{Nat} \land d.\text{directory}(cl)=0 \land \text{this}=cl \\
{\{ \text{d.repay()} \}}
\end{align*}

“nothing changes”

... specs for another 19 functions ... 

**Classical**

- per function; *sufficient* conditions for some action/effect
- *explicit* about individual function, and *implicit* about emergent behaviour
classical vs holistic

Classical

• per function; *sufficient* conditions for some action/effect

• *explicit* about individual function, and *implicit* about emergent behaviour

Holistic

• *necessary* conditions for some action/effect

• *explicit* about emergent behaviour

∀ d:DAO. d.ether = \( \sum_{cl \in \text{dom}(d.\text{directory})} d.\text{directory}(cl) \)

\[ d:\text{DAO} \land n:\text{Nat} \land d.\text{directory}(cl) = n>0 \land \text{this}=cl \]

\{ d.\text{repay}() \}

\[ d.\text{directory}(cl) = 0 \land d.\text{Calls}(cl.\text{send}(n)) \]

"nothing changes"

\[ \vdots \quad \text{specs for another 19 functions} \quad \vdots \]

∀ cl:External. ∀ d:DAO. ∀ n:Nat. \[ [ \text{cl.Calls}(d.\text{repay}()) \land d.\text{Balance}(cl) = n \] \[ \rightarrow \]

\[ \quad d.\text{ether} \equiv n \land \text{Will}(d.\text{Calls}(cl.\text{send}(n))) ] \]

\[ \quad d.\text{Balance}(cl) = m \]

\[ \quad m+m' \quad \text{if cl.Calls}(d,\text{initialize},m) \]

\[ \quad \text{if Was}(d.\text{Balance}(cl),m) \]

\[ \quad \land \text{Was}(\text{cl.Calls}(d.\text{payIn}(m'))) \]

\[ \quad 0 \quad \text{if Was}(\text{cl.Calls}(d.\text{repayIn}()) \]
Example 3: DOM attenuation
Example 3: DOM attenuation

Access to any Node gives access to complete tree
Example 3: DOM attenuation

Access to any Node gives access to complete tree
Example 3: DOM attenuation

Access to any Node gives access to complete tree
Example 3: DOM attenuation

Access to any Node gives access to complete tree

Wrappers have a height; Access to Wrapper w allows modification of Nodes under the w.height-th parent and nothing else
Example 3: DOM attenuation

Access to any Node gives access to complete tree

Wrappers have a height; Access to Wrapper \( w \) allows modification of Nodes under the \( w.height \)-th parent and nothing else
Example 3: DOM attenuation

Access to any Node gives access to **complete** tree

Wrappers have a height; Access to Wrapper w allows modification of Nodes under the w.height-th parent *and nothing else*
function mm(unknown) {
    n1 := Node(...); n2 := Node(n1,...); n3 := Node(n2,...); n4 := Node(n3,...);
    n2.p := "robust"; n3.p := "volatile";
function mm(unknwn) {
    n1:=Node(...); n2:=Node(n1,...); n3:=Node(n2,...); n4:=Node(n3,...);
    n2.p:="robust"; n3.p:="volatile";
    w=Wrapper(n4,1);
}
function mm(unknwn) {
    n1:=Node(...); n2:=Node(n1,...); n3:=Node(n2,...); n4:=Node(n3,...);
    n2.p:="robust"; n3.p:="volatile";
    w=Wrapper(n4,1);
    unknwn.untrusted(w);
    ....
function mm(unknown) {
    n1:=Node(...); n2:=Node(n1,...); n3:=Node(n2,...); n4:=Node(n3,...);
    n2.p:="robust"; n3.p:="volatile";
    w=Wrapper(n4,1);
    unknown.untrusted(w);
    ...
}

Here:  n3.p = ?????  
n2.p = ?????
function mm(unknown) {
    n1 := Node(...); n2 := Node(n1, ...); n3 := Node(n2, ...); n4 := Node(n3, ...);
    n2.p := "robust"; n3.p := "volatile";
    w = Wrapper(n4, 1);
    unknown.untrusted(w);
    ...
}

Here: n3.p = ????
    n2.p = "robust"
DOM attenuation

```javascript
function mm(unkwn) {
    n1 := Node(...); n2 := Node(n1,...); n3 := Node(n2,...); n4 := Node(n3,...);
    n2.p := "robust"; n3.p := "volatile";
    w := Wrapper(n4, 1);
    unknwn.untrusted(w);
    ...
}
```

Here: n3.p = ????
Here: n2.p = “robust”
function mm(unknwn)
    n1:=Node(...); n2:=
    n2.p:="robust"; n3.p:="volatile";
    w=Wrapper(n4,1);
    unknwn.untrusted(w);
    ...

Here: n3.p = ????
Here: n2.p = "robust"
function mm(unknwn) {
    n1 := Node(...); n2 := Node(n1,...); n3 := Node(n2,...); n4 := Node(n3,...);
    n2.p := "robust"; n3.p := "volatile";
    w = Wrapper(n4,1);
    unknwn.untrusted(w);
    ....

    Here: n3.p = ?????
    n2.p = "robust"
}
Access to Wrapper $w$ allows modification of $\texttt{Nodes}$ under the $w.\texttt{height}$-th parent and nothing else
Access to Wrapper $w$ allows modification of $\text{Nodes}$ under the $w\.\text{height}$-th parent and nothing else.
Access to Wrapper \( w \) allows modification of \( \text{Nodes} \) under the \( w.\text{height} \)-th parent and nothing else.
holistic

unknown1
holistic
If a node $\text{nd}$ is external to a set $S$ then any execution involving no more than $S$ does not modify $\text{nd}. p$

$\text{Exterma}(\text{nd},S)$ iff ....
holistic
∀S:Set. ∀nd:Node. 
[ Extenal(nd,S) → ¬ ( Will(Changes(nd.p)) in S ) ]
\( \forall S: \text{Set. } \forall \text{nd:Node.} \)

\[
\begin{align*}
\text{Extenal}(\text{nd}, S) & \Rightarrow \neg (\text{Will(Changes(\text{nd.p})) in } S) \\
\end{align*}
\]

Extenal(\text{nd}, S) \iff \ldots
∀S:Set. ∀nd:Node.
[ External(nd,S) → ¬ ( Will(Changes(nd.p)) in S ) ]
\( \forall S: \text{Set.} \ \forall \text{nd:Node.} \ \left[ \text{Extenal}(\text{nd},S) \rightarrow \neg (\text{Will(Changes}(\text{nd.p})) \text{ in } S) \right] \)

- \text{nd external to } S
- Execution involves no more than S

\text{Extenal}(\text{nd},S) \iff \ldots

\text{Extenal}(\text{nd},S)

\text{unknwn1}

\text{unknwn2}

w: \text{Wrapper height=1}

\text{Execution involves no more than S}
∀S:Set. ∀nd:Node.
[ External(nd,S) \rightarrow \neg (\text{Will(Changes(nd.p)) in S}) ]

Execution involves no more than S

nd external to S

Does not modify nd.p

∀S:Set. ∀nd:Node.
[ External(nd,S) \rightarrow \neg (\text{Will(Changes(nd.p)) in S}) ]

\text{Execution involves no more than S}

\text{nd external to S}

\text{Does not modify nd.p}
holistic
holistic
\( \forall S : \text{Set.} \forall nd : \text{Node.} \) 
\[ \text{Extenal}(nd, S) \rightarrow \neg (\text{Will} (\text{Changes}(nd.p)) \text{ in } S ) \]
∀S:Set. ∀nd:Node.
[ Extenal(nd,S) → ¬ ( Will(Changes(nd.p)) in S ) ]

Extenal(nd,S) iff ∀o∈S. ∀path
[ o.path ≠ nd ∨ o:Node ∨
  ∃ path’,fs. ( path=(path’.fs ∧ o.path’:Wrapper ∧
  Distance(o.path’,nd)>o.path’.height ) ]

Distance(nd,nd’) = min{ k | nd.parent^k = nd’.parent^j }
Extenal(nd,S) iff ∀o∈S. ∀path
   [ o.path ≠ nd ∨ 
     o:Node ∨ 
     ∃ path’,fs. ( path= path’.fs ∧ o.path’:Wrapper ∧ 
                     Distance(o.path’,nd)>o.path’.height ) ]

Distance(nd,nd’) = min{ k | nd.parent^k = nd’.parent^j }
External\( (nd, S) \) iff \( \forall o \in S. \forall path \)
\[
\left[ \begin{array}{c}
o.\text{path} \neq nd \\
o: \text{Node} \\
\exists \ \text{path}', \text{fs} . \left( \text{path} = \text{path}'.\text{fs} \land o.\text{path}'.\text{Wrapper} \land \\
\text{Distance}(o.\text{path}', nd) > o.\text{path}'.\text{height} \right) \end{array} \right]
\]

\( \text{Distance}(nd, nd') = \min \{ k \mid nd.\text{parent}^k = nd'.\text{parent}^j \} \)
Extenal(\textcolor{red}{RedNode}, \textcolor{yellow}{YellowSet})

Extenal\((nd,S)\) iff \(\forall o \in S. \forall \text{path}\)
\[
\begin{array}{l}
o.\text{path} \neq nd \lor \\
o: \text{Node} \lor \\
\exists \text{path'}, \text{fs}. (\text{path}=\text{path'}.\text{fs} \land o.\text{path'}:\text{Wrapper} \land \text{Distance}(o.\text{path'}, nd) > o.\text{path'}.\text{height})
\end{array}
\]

\[
\text{Distance}(nd, nd') = \min\{ k \mid nd.\text{parent}^k = nd'.\text{parent}^j \}
\]
External\((nd,S)\) iff $\forall o \in S. \forall \text{path}
\[
[\quad o.\text{path} \neq nd \lor
o:\text{Node} \lor
\exists \text{path}',fs. (\text{path}=\text{path}'.fs \land o.\text{path}':\text{Wrapper} \land
\text{Distance}(o.\text{path}', nd) > o.\text{path}'.\text{height})\quad]\n\]

Distance\((nd, nd')\) = min\{ k \mid nd.\text{parent}^k = nd'.\text{parent}^j \\}
External$(\text{nd}, S)$ iff $\forall o \in S. \, \forall \text{path}
\left[ \begin{array}{l}
o.\text{path} \neq \text{nd} \lor 
o: \text{Node} \lor 
\exists \text{path}', \text{fs}.
\left( \begin{array}{l}
\text{path} = \text{path}'.\text{fs} \land o.\text{path}': \text{Wrapper} \land 
\text{Distance}(o.\text{path}', \text{nd}) \gt o.\text{path}'.\text{height}
\end{array} \right)
\end{array} \right]$  

Distance$(\text{nd}, \text{nd}') = \min\left\{ k \mid \text{nd}.\text{parent}^k = \text{nd}'.\text{parent}^j \right\}$
Extenal(RedNode, YellowSet)

\[\text{Extenal}(nd, S) \iff \forall o \in S. \forall \text{path} \]
\[
\left[ o.\text{path} \neq nd \lor
o: \text{Node} \lor
\exists \text{path}', fs. (\text{path} = \text{path}'.fs \land o.\text{path}' : \text{Wrapper} \land
\text{Distance}(o.\text{path}', nd) > o.\text{path}'.\text{height} ) \right]
\]

\[\text{Distance}(nd, nd') = \min \{ k \mid \text{nd.parent}^k = \text{nd}'.\text{parent}^j \} \]
\( \text{External}(\text{RedNode}, \text{YellowSet}) \)

\[
\text{External}(\text{nd}, S) \iff \forall o \in S. \forall \text{path} \\
[ \text{path} \neq \text{nd} \lor \text{nd} : \text{Node} \lor \\
\exists \text{path}', \text{fs}. (\text{path} = \text{path}'. \text{fs} \land \text{path}' : \text{Wrapper} \land \\
\text{Distance}(\text{path}', \text{nd}) > \text{path}'. \text{height} ) ]
\]

\[
\text{Distance}(\text{nd}, \text{nd}') = \min\{ k \mid \text{nd}. \text{parent}^k = \text{nd}'. \text{parent}^j \}
\]
Extenal(nd,S) iff $\forall o \in S. \forall path$

$[ o.path \neq nd \lor$

$o:Node \lor$

$\exists \ path',fs. ( path=\path'.fs \land o.path':Wrapper \land$

$Distance(o.path',nd) > o.path'.height ) ]$

$Distance(nd,nd') = \min\{ k \mid nd.parent^k = nd'.parent^j \}$
Extenal(nd,S) iff $\forall o \in S. \forall path$

$[ o.path \neq nd \lor$

$o:\text{Node} \lor$

$\exists path',fs. (path = path'.fs \land o.path':\text{Wrapper} \land$

$\text{Distance}(o.path',\text{nd}) > o.path'.height ) ]$

$\text{Distance}(\text{nd},\text{nd'}) = \min\{ k \mid \text{nd.parent}^k = \text{nd'}.parent^j \}$
Extenal(\textbf{RedNode}, \textbf{YellowSet})

\textbf{Extenal}(nd,S) \iff \forall o \in S. \forall \text{path}
\left[\begin{array}{l}
o.\text{path} \neq nd \lor 
o: \text{Node} \lor 
\exists \text{path}',fs. (\text{path} = \text{path}'.fs \land o.\text{path}':\text{Wrapper} \land 
\text{Distance}(o.\text{path}',nd) > o.\text{path}'.\text{height} \land \right]

\text{Distance}(nd,nd') = \min\{ k \mid nd.\text{parent}^k = nd'.\text{parent}^j \land \right\}
Extenal(\text{RedNode}, \text{YellowSet})

Extenal(nd,S) \iff \forall o \in S. \forall \text{path}[
\begin{align*}
o.\text{path} &\neq nd \lor \\
o: \text{Node} &\lor \\
\exists \text{path'}, \text{fs}. \left( \text{path} = \text{path'}.\text{fs} \land o.\text{path'}: \text{Wrapper} \land \\
\text{Distance}(o.\text{path'}, nd) > o.\text{path'}.\text{height} \right)
\end{align*}
]

\text{Distance}(nd, nd') = \min \{ k \mid nd.\text{parent}^k = nd'.\text{parent}^j \}
using holistic spec

```javascript
function mm(unknwn) {
    n1 := Node(...);
    n2 := Node(n1,...);
    n3 := Node(n2,...);
    n4 := Node(n3,...);
    n2.p := "robust";
    n3.p := "volatile";
}
```
using holistic spec

```javascript
function mm(unknwn) {
    n1 := Node(...); n2 := Node(n1,...); n3 := Node(n2,...); n4 := Node(n3,...);
    n2.p := “robust”; n3.p := “volatile”;
    w = Wrapper(n4,1);
}
```
function mm(unknwn) {
    n1:=Node(...); n2:=Node(n1,...); n3:=Node(n2,...); n4:=Node(n3,...);
    n2.p:="robust"; n3.p:="volatile";
    w=Wrapper(n4,1);
    unknwn.untrusted(w);
    ....
using holistic spec

```javascript
function mm(unknown) {
    n1 := Node(...);
    n2 := Node(n1,...);
    n3 := Node(n2,...);
    n4 := Node(n3,...);
    n2.p := "robust";
    n3.p := "volatile";
    w := Wrapper(n4,1);
    unknown.untrusted(w);
    ...
}
```

With holistic spec we can show that despite the call to unknown object, at this point: n2.p = "robust"
Using holistic spec

```javascript
function mm(unknwn) {
    n1 := Node(...); n2 := Node(n1,...); n3 := Node(n2,...); n4 := Node(n3,...);
    n2.p := "robust"; n3.p := "volatile";
    w = Wrapper(n4,1);
    unknwn.untrusted(w);
    ...
}
```

With holistic spec we can show that despite the call to unknown object, at this point:

```
n2.p = "robust"
```

unknown

w:Wrapper height=1
Bank and Account

- Banks and Accounts
- Accounts hold money
- Money can be transferred between Accounts
- A bank's currency = sum of balances of accounts held by bank

[Miller et al, Financial Crypto 2000]
**Bank/Account - 2**

- **Pol_1**: With two accounts of same bank one can transfer money between them.
- **Pol_2**: Only someone with the Bank of a given currency can violate conservation of that currency.
- **Pol_3**: The bank can only inflate its own currency.
- **Pol_4**: No one can affect the balance of an account they do not have.
- **Pol_5**: Balances are always non-negative.
- **Pol_6**: A reported successful deposit can be trusted as much as one trusts the account one is depositing to.

[Miller et al, Financial Crypto 2000]
Pol_4 — holistic

Pol_4: No-one can affect the balance of an account they do not have
Pol_4  — holistic

- **Pol_4**: No-one can affect the balance of an account they do not have

\[ a: \text{Account} \land \text{Will}(\text{Changes}(a.\text{balance})) \text{ in } S \]
Pol_4 — holistic

- **Pol_4**: No-one can affect the balance of an account they do not have

\[
\begin{align*}
a &: \text{Account} & \land & \text{Will ( Changes}(a.\text{balance}) & \text{ in S }) \\
\rightarrow \\
\exists o \in S. \text{Access}(o,a)
\end{align*}
\]
Pol_4 — holistic

- **Pol_4**: No-one can affect the balance of an account they do not have

\[
a: \text{Account} \land \text{Will}(\text{Changes}(a.\text{balance})) \text{ in } S\rightarrow \\
\exists o \in S. \text{Access}(o,a)
\]

*This says*: If some execution starts now and involves at most the objects from \(S\), and modifies \(a.\text{balance}\) at some future time, then at least one of the objects in \(S\) can access \(a\) directly now.
Pol_4 — holistic

Pol_4: No-one can affect the balance of an account they do not have

\[ a: \text{Account} \land \text{Will}(\text{Changes}(a.\text{balance})) \text{ in } S \rightarrow \exists o \in S. \text{Access}(o,a) \]

This says: If some execution starts now and involves at most the objects from \( S \), and modifies \( a.\text{balance} \) at some future time, then at least one of the objects in \( S \) can access \( a \) directly now.
Pol_4 — classical

Pol_4: No-one can affect the balance of an account they do not have
Pol_4 — classical

- Pol_4: No-one can affect the balance of an account they do not have

???
???
??
???
Today

- Traditional Specifications do not adequately address Robustness
- Holistic Specifications — Summary and by Example
- Holistic Specification Semantics
Giving meaning to holistic Assertions
Giving meaning to holistic Assertions

We define in a “conventional” way (omit from slides):

module \( M : \text{Ident} \longrightarrow \text{ClassDef} \cup \text{PredicateDef} \cup \text{FunctionDef} \)

configuration \( \sigma : \text{Heap} \times \text{Stack} \times \text{Code} \)

execution \( M, \sigma \rightarrow \sigma' \)
Giving meaning to holistic Assertions

We define in a “conventional” way (omit from slides):

module $M : \text{Ident} \rightarrow \text{ClassDef} \cup \text{PredicateDef} \cup \text{FunctionDef}$
configuration $\sigma : \text{Heap} \times \text{Stack} \times \text{Code}$
execution $M, \sigma \rightarrow \sigma'$

Define module concatenation $\ast$ so that
$M \ast M'$ undefined, iff $\text{dom}(M) \cap \text{dom}(M') \neq \emptyset$
otherwise
$(M \ast M')(id) = M(id)$ if $M'(id)$ undefined, else $M'(id)$
Giving meaning to holistic Assertions

We define in a “conventional” way (omit from slides):

module $M : \text{Ident} \rightarrow \text{ClassDef} \cup \text{PredicateDef} \cup \text{FunctionDef}$
configuration $\sigma : \text{Heap} \times \text{Stack} \times \text{Code}$
execution $M, \sigma \rightarrow \sigma'$

Define module concatenation $\ast$ so that
$\ast$ undefined, iff $\text{dom}(M) \cap \text{dom}(M') \neq \emptyset$
otherwise
$(\ast M')(id) = M(id)$ if $M'(id)$ undefined, else $M'(id)$

**Lemma**
- $M \ast M' = M' \ast M$
- $(M1 \ast M2) \ast M3 = M1 \ast (M2 \ast M3)$
- $M, \sigma \rightarrow \sigma' \land M \ast M'$ defined $\rightarrow M \ast M', \sigma \rightarrow \sigma'$
Giving meaning to holistic Assertions

We define in a “conventional” way (omit from slides):

module \( M : \text{Ident} \longrightarrow \text{ClassDef} \cup \text{PredicateDef} \cup \text{FunctionDef} \)
configuration \( \sigma : \text{Heap} \times \text{Stack} \times \text{Code} \)
execution \( M, \sigma \rightarrow \sigma' \)

Define module concatenation \( * \) so that
\( M * M' \) undefined, iff \( \text{dom}(M) \cap \text{dom}(M') \neq \emptyset \)
otherwise
\((M * M')(id) = M(id) \) if \( M'(id) \) undefined, else \( M'(id) \)

We will define
\( M, \sigma \models A \)
Initial(\( \sigma \)) and Arising(M)
\( M \models A \)
Giving meaning to holistic Assertions
Giving meaning to holistic Assertions

We define in a “conventional” way (omit from slides):

module \( M : \text{Ident} \rightarrow \text{ClassDef} \cup \text{PredicateDef} \cup \text{FunctionDef} \)
configuration \( \sigma : \text{Heap} \times \text{Stack} \times \text{Code} \)
execution \( M, \sigma \rightarrow \sigma' \)

Define module concatenation \( * \) so that
\[ M * M' \text{ undefined, iff } \text{dom}(M) \cap \text{dom}(M') \neq \emptyset \]
otherwise
\[ (M * M')(\text{id}) = M(\text{id}) \text{ if } M'(\text{id}) \text{ undefined, else } M'(\text{id}) \]
Giving meaning to holistic Assertions

We define in a “conventional” way (omit from slides):

module $M : \text{Ident} \rightarrow \text{ClassDef} \cup \text{PredicateDef} \cup \text{FunctionDef}$
configuration $\sigma : \text{Heap} \times \text{Stack} \times \text{Code}$
execution $M, \sigma \rightarrow \sigma'$

Define module concatenation $*$ so that
$M*M'$ undefined, iff $\text{dom}(M) \cap \text{dom}(M') \neq \emptyset$
otherwise $(M*M')(id) = M(id)$ if $M'(id)$ undefined, else $M'(id)$

We will define $M, \sigma \models A$
Initial($\sigma$) and Arising($M$)
$M \models A$
Holistic Assertions — summary
Holistic Assertions — summary

e ::= this | x | e.fld | ...
Holistic Assertions — summary

e ::= this | x | e.fld | ...

A ::= e>e | e=e | ...
Holistic Assertions — summary

e ::= this | x | e.fld | ...

A ::= e>e | e=e | ...
    | A → A | A ∧ A | ∃x. A | ...

Holistic Assertions — summary

e ::= this | x | e.fld | ...

A ::= e>e | e=e | ...
    | A → A | A ∧ A | ∃x. A | ...
    | Access(e,e')
Holistic Assertions — summary

e ::= this | x | e.fld | ...

A ::= e>e | e=e | ...
   | A → A | A ∧ A | ∃x. A | ...
   | **Access**(e,e')
   | **Changes**(e)
Holistic Assertions — summary

e ::= this | x | e.fld | ...

A ::= e>e | e=e | ...
    | A → A | A ∧ A | ∃x. A | ...
    | Access(e,e')
    | Changes(e)
    | Will(A) | Was(A)
Holistic Assertions — summary

\[
\begin{align*}
e &::= \text{this} \mid x \mid e.fld \mid \ldots \\
A &::= e>e \mid e=e \mid \ldots \\
&\quad \mid A \rightarrow A \mid A \land A \mid \exists x. A \mid \ldots \\
&\quad \mid \text{Access}(e,e') \\
&\quad \mid \text{Changes}(e) \\
&\quad \mid \text{Will}(A) \mid \text{Was}(A) \\
&\quad \mid A \in S
\end{align*}
\]
Holistic Assertions — summary

\[ e ::= \text{this} \mid x \mid e.\text{fld} \mid \ldots \]

\[ A ::= e>e \mid e=e \mid \ldots \]
\[ \mid A \rightarrow A \mid A \land A \mid \exists x. A \mid \ldots \]
\[ \mid \text{Access}(e,e') \]
\[ \mid \text{Changes}(e) \]
\[ \mid \text{Will}(A) \mid \text{Was}(A) \]
\[ \mid A \text{ in } S \]
\[ \mid x.\text{Call}(y,m,z_1,\ldots,z_n) \]
Holistic Assertions — summary

e ::= this | x | e.fld | ...

A ::= e>e | e=e | ...
| A → A | A ∧ A | ∃x. A | ...
| Access(e,e')
| Changes(e)
| Will(A) | Was(A)
| A in S
| x.Call(y,m,z1,..zn)
| x obeys A
Holistic Assertions — summary

\[ e ::= \text{this} \mid x \mid e.fld \mid \ldots \]

\[ A ::= e > e \mid e = e \mid \ldots \]

\[ \mid A \rightarrow A \mid A \land A \mid \exists x. A \mid \ldots \]

<table>
<thead>
<tr>
<th><strong>Access</strong>((e,e'))</th>
<th>permission</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Changes</strong>((e))</td>
<td>authority</td>
</tr>
<tr>
<td><strong>Will</strong>((A))</td>
<td>time</td>
</tr>
<tr>
<td><strong>Was</strong>((A))</td>
<td></td>
</tr>
<tr>
<td><strong>A in</strong>(S)</td>
<td>space</td>
</tr>
<tr>
<td><strong>x.Call</strong>((y,m,z1,..zn))</td>
<td>control</td>
</tr>
<tr>
<td><strong>x obeys</strong>(A)</td>
<td>trust</td>
</tr>
</tbody>
</table>
Semantics of Expressions

e ::= this | x | e.fld | func(e1,...en) | ...

Define $\mathcal{L}_e \downarrow_{M,\sigma}$ as expected
Semantics of Expressions

e ::= this | x | e.fld | func(e1,...en) | ...

Define $e \downarrow_{M,\sigma}$ as expected
Semantics of Expressions

\[ e ::= \text{this} \mid x \mid e.fld \mid \text{func}(e_1, \ldots, e_n) \mid \ldots \]

Define \( \semantics{e}{M, \sigma} \) as expected

Eg, \( \semantics{\text{unknwn1}.nd.par.par.par.p}{M, \sigma} = 888 \)
Semantics of holistic Assertions

“Conventional part”

\[ A ::= e > e \quad | \quad A \rightarrow A \quad | \quad \exists x. A \quad | \quad \ldots \]
Semantics of holistic Assertions

“Conventional part”

\[
A ::= e>e \mid A \rightarrow A \mid \exists x.A \mid ...
\]

We define \( M, \sigma \models A \)
“Conventional part”

\[ A ::= e > e \mid A \rightarrow A \mid \exists x. A \mid ... \]

We define \( M, \sigma \models A \)

\[ M, \sigma \models e > e' \iff \llbracket e \rrbracket_{M, \sigma} > \llbracket e' \rrbracket_{M, \sigma} \]

\[ M, \sigma \models A \rightarrow A' \iff M, \sigma \models A \text{ implies } M, \sigma \models A' \]

\[ M, \sigma \models \exists x. A \iff M, \sigma[z \mapsto i] \models A[x \mapsto z] \]

for some \( i \in \text{dom}(\sigma.\text{heap}) \), and \( z \) free in \( A \)
Semantics of holistic Assertions

“Unconventional part”

\[ A ::= \text{Access}(x, x') \mid \text{Changes}(e) \mid \text{Will}(A) \mid A \text{ in } S \mid x.\text{Calls}(y, m, z_1, \ldots, z_n) \]
Semantics of holistic Assertions

“Unconventional part”

\[ A ::= \text{Access}(x,x') \mid \text{Changes}(e) \mid \text{Will}(A) \mid A \text{ in } S \mid x.\text{Calls}(y,m,z_1,..z_n) \]

\[ M, \sigma \models \text{Access}(x,x') \iff \models x \downarrow M,\sigma = x' \downarrow M,\sigma \lor \models x.\text{fld} \downarrow M,\sigma = x' \downarrow M,\sigma \text{ for some field fld} \lor \models \text{this} \downarrow M,\sigma = x \downarrow M,\sigma \land y \downarrow M,\sigma = x' \downarrow M,\sigma \]

\( \land y \) is formal parameter of current function
Semantics of holistic Assertions

“Unconventional part”

A ::= \textbf{Access}(x,x') | \textbf{Changes}(e) | \textbf{Will}(A) | A \text{ in } S | x.\textbf{Calls}(y,m,z_1,..z_n)

\[ M,\sigma \models \textbf{Access}(x,x') \text{ iff } \land_{\text{\textbf{x} = \textbf{x}'}} M,\sigma = \land_{\text{\textbf{x} = \textbf{x}'}} M,\sigma \lor \land_{\text{\textbf{x}.fld}} M,\sigma = \land_{\text{\textbf{x}'}} M,\sigma \text{ for some field fld} \lor \land_{\text{this}} M,\sigma = \land_{\text{\textbf{x}}} M,\sigma \land \land_{\text{\textbf{y}}} M,\sigma = \land_{\text{\textbf{x}'}} M,\sigma \land \text{\textbf{y} is formal parameter of current function} \]

\[ M,\sigma \models \textbf{Changes}(e) \text{ iff } M,\sigma \rightarrow \sigma' \land \land_{\text{e}} M,\sigma \neq \land_{\text{e}} M,\sigma'. \]
Semantics of holistic Assertions

“Unconventional part”

\[ A ::= \text{Access}(x,x') \mid \text{Changes}(e) \mid \text{Will}(A) \mid A \text{ in } S \mid x.\text{Calls}(y,m,z_1,..z_n) \]

\[ M, \sigma \models \text{Access}(x,x') \text{ iff } \begin{align*}
& \downarrow x \downarrow_{M,\sigma} = \downarrow x' \downarrow_{M,\sigma} \lor \\
& \downarrow x.fld \downarrow_{M,\sigma} = \downarrow x' \downarrow_{M,\sigma} \text{ for some field fld} \lor \\
& \downarrow \text{this} \downarrow_{M,\sigma} = \downarrow x \downarrow_{M,\sigma} \land \downarrow y \downarrow_{M,\sigma} = \downarrow x' \downarrow_{M,\sigma} \\
& \land y \text{ is formal parameter of current function}
\end{align*} \]

\[ M, \sigma \models \text{Changes}(e) \text{ iff } M, \sigma \rightarrow \sigma' \land \downarrow e \downarrow_{M,\sigma} \neq \downarrow e \downarrow_{M,\sigma'} \]

\[ M, \sigma \models \text{Will}(A) \text{ iff } \exists \sigma'.[ M, \sigma \rightarrow^* \sigma' \land M, \sigma' \models A ] \]
Semantics of holistic Assertions

“Unconventional part”

A ::= \textbf{Access}(x, x') \mid \textbf{Changes}(e) \mid \textbf{Will}(A) \mid A \textbf{ in } S \mid x.\textbf{Calls}(y, m, z_1, .. z_n)

$M, \sigma \models \textbf{Access}(x, x') \iff \ll x \rr_{M, \sigma} = \ll x' \rr_{M, \sigma} \lor$

$\ll x.\text{fld} \rr_{M, \sigma} = \ll x' \rr_{M, \sigma}$ for some field fld \lor

$\ll \text{this} \rr_{M, \sigma} = \ll x \rr_{M, \sigma} \land \ll y \rr_{M, \sigma} = \ll x' \rr_{M, \sigma}$

$\land y$ is formal parameter of current function

$M, \sigma \models \textbf{Changes}(e) \iff M, \sigma \leadsto \sigma' \land \ll e \rr_{M, \sigma} \neq \ll e \rr_{M, \sigma'}$

$M, \sigma \models \textbf{Will}(A) \iff \exists \sigma'.[ M, \sigma \leadsto^* \sigma' \land M, \sigma' \models A ]$

$M, \sigma \models A \textbf{ in } S \iff M, \sigma @ Os \models A \text{ where } Os = \ll S \rr_{M, \sigma}$
Semantics of holistic Assertions

“Unconventional part”

\[ A ::= \text{Access}(x, x') \mid \text{Changes}(e) \mid \text{Will}(A) \mid A \text{ in } S \mid x.\text{Calls}(y, m, z_1, \ldots z_n) \]

\[ M, \sigma \models \text{Access}(x, x') \iff \llbracket x \rrbracket_{M, \sigma} = \llbracket x' \rrbracket_{M, \sigma} \lor \llbracket x.\text{fld} \rrbracket_{M, \sigma} = \llbracket x' \rrbracket_{M, \sigma} \text{ for some field fld } \lor \llbracket \text{this} \rrbracket_{M, \sigma} = \llbracket x \rrbracket_{M, \sigma} \land \llbracket y \rrbracket_{M, \sigma} = \llbracket x' \rrbracket_{M, \sigma} \land y \text{ is formal parameter of current function} \]

\[ M, \sigma \models \text{Changes}(e) \iff M, \sigma \rightarrow \sigma' \land \llbracket e \rrbracket_{M, \sigma} \neq \llbracket e \rrbracket_{M, \sigma'} \]

\[ M, \sigma \models \text{Will}(A) \iff \exists \sigma'.[ M, \sigma \rightarrow^* \sigma' \land M, \sigma' \models A ] \]

\[ M, \sigma \models A \text{ in } S \iff M, \sigma@\text{Os} \models A \text{ where } \text{Os} = \llbracket S \rrbracket_{M, \sigma} \]

\[ M, \sigma \models x.\text{Calls}(y, m, z_1, \ldots z_n) \iff \llbracket \text{this} \rrbracket_{M, \sigma} = \llbracket x \rrbracket_{M, \sigma} \land \sigma.\text{code}=y'.m(z_1'..z_n') \land \ldots \]
Semantics of holistic Assertions
- the full truth -

\[ M, \sigma \vDash \textbf{Access}(e, e') \text{ iff } \ldots \text{ as before } \ldots \]

\[ M, \sigma \vDash \textbf{Changes}(e) \text{ iff } M, \sigma \rightarrow \sigma' \land \bot_{M, \sigma} \neq \bot_{[z \mapsto y]M, \sigma'_{[y \mapsto \sigma(z)]}} \]

\[ \text{where } \{z\} = \text{Free}(e) \land y \text{ fresh in } e, \sigma, \sigma' \]

\[ M, \sigma \vDash \textbf{Will}(A) \text{ iff } \exists \sigma', \sigma'', \phi. [ \sigma = \sigma'. \phi \land M, \phi \rightarrow^* \sigma' \land M, \sigma'_{[y \mapsto \sigma(z)]} \vdash A[z \mapsto y] ] \]

\[ \text{where } \{z\} = \text{Free}(A) \land y \text{ fresh in } A, \sigma, \sigma' \]

\[ M, \sigma \vDash A \textbf{In } S \text{ iff } M, \sigma@o_s \vdash A \text{ where } o_s = \bot_{S_{M, \sigma}} \]

\[ M, \sigma \vDash x. \textbf{Calls}(y, m, z_1, \ldots, z_n) \text{ iff } \ldots \text{ as before } \ldots \]
Semantics of holistic Assertions
- the full truth -

\[ M, \sigma \models \textbf{Access}(e, e') \text{ iff } \ldots \text{ as before } \ldots \]

\[ M, \sigma \models \textbf{Changes}(e) \text{ iff } M, \sigma \rightarrow \sigma' \land |e|_{M, \sigma} \neq |e[z \mapsto y]|_{M, \sigma'[y \mapsto \sigma(z)]} \]

where \( \{z\} = \text{Free}(e) \land y \text{ fresh in } e, \sigma, \sigma' \)

\[ M, \sigma \models \textbf{Will}(A) \text{ iff } \exists \sigma', \sigma'', \phi. [ \sigma = \sigma'. \phi \land M, \phi \rightarrow^* \sigma' \land M, \sigma'[y \mapsto \sigma(z)] \models A[z \mapsto y] ] \]

where \( \{z\} = \text{Free}(A) \land y \text{ fresh in } A, \sigma, \sigma' \)

\[ M, \sigma \models A \textbf{In } S \text{ iff } M, \sigma@O_s \models A \text{ where } O_s = \downarrow S \downarrow_{M, \sigma} \]

\[ M, \sigma \models x.\textbf{Calls}(y, m, z_1, \ldots z_n) \text{ iff } \ldots \text{ as before } \ldots \]
A runtime configuration is *initial* iff
1) The heap contains only one object, of class Object
2) The stack consists of just one frame, where `this` points to that object.

The code can be arbitrary

\[
\text{Initial}(\sigma) \ \text{iff} \ \sigma.\text{heap} = (1 \mapsto (\text{Object}, \ldots)) \land \sigma.\text{stack} = (\text{this} \mapsto 1).
\]

A runtime configuration \( \sigma \) *arises* from a module \( M \) if there is some initial configuration \( \sigma_0 \) whose execution \( M \) in reaches \( \sigma \) in a finite number of steps.

\[
\text{Arising}(M) = \{ \sigma \mid \exists \sigma_0. \ \text{Initial}(\sigma_0) \land M, \sigma_0 \rightarrow^* \sigma \}
\]
Arising expresses “defensiveness”

Assume a Tree-module, $M_{\text{tree}}$. 
Arising expresses “defensiveness”

Assume a Tree-module, $M_{\text{tree}}$. 
Arising expresses “defensiveness”

Assume a Tree-module, $M_{\text{tree}}$. 

---

48
Arising expresses “defensiveness”

Assume a Tree-module, $M_{tree}$.

Blue configuration arises from $M_{tree}^* M'$ for some module $M'$.
Arising expresses “defensiveness”

Assume a Tree-module, \( M_{\text{tree}} \).

Blue configuration arises from \( M_{\text{tree}} \times M' \) for some module \( M' \)

Brown configuration does not arise from \( M_{\text{tree}} \times M' \) for any module \( M' \)
Giving meaning to Assertions

\[ M \models A \ \text{iff} \ \forall M'. \forall \sigma \in \text{Arising}(M^*M'). \ M^*M', \ \sigma \models A \]

A module \( M \) satisfies an assertion \( A \) if all runtime configurations \( \sigma \) which arise from execution of code from \( M^*M' \) (for any module \( M' \)), satisfy \( A \).
Giving meaning to Assertions

\[ M \models A \iff \forall M', \forall \sigma \in \text{Arising}(M^*M'), M^*M', \sigma \models A \]

A module \( M \) satisfies an assertion \( A \) if all runtime configurations \( \sigma \) which arise from execution of code from \( M^*M' \) (for any module \( M' \)), satisfy \( A \).
Giving meaning to Assertions

\[ M \models A \text{ iff } \forall M'. \forall \sigma \in \text{Arising}(M*M'), M*M', \sigma \models A \]

A module \( M \) satisfies an assertion \( A \) if all runtime configurations \( \sigma \) which arise from execution of code from \( M*M' \) (for any module \( M' \)), satisfy \( A \).
Summary of our Proposal

\[ A ::= e>e \ | \ e=e \ | \ f(e1,..en) \ | \ ... \]
\[ \ | \ A \rightarrow A \ | \ A \land A \ | \ \exists x. A \ | \ ... \]
\[ \ | \ \text{Access}(x,y) \]
\[ \text{permission} \]
\[ \ | \ \text{Changes}(e) \]
\[ \text{authority} \]
\[ \ | \ \text{Will}(A) \ | \ \text{Was}(A) \]
\[ \text{time} \]
\[ \ | \ A \text{ in } S \]
\[ \text{space} \]
\[ \ | \ x.\text{Calls}(y,m,z1,..zn) \]
\[ \text{call} \]

\[ M, \sigma \models A \]
\[ \text{Arising}(M) \]
\[ M \models A \]
Classical Specification vs Holistic Specification

- fine-grained
- per function

- ADT as a whole
- emergent behaviour
Classical Specification vs Holistic Specification

- fine-grained
- per function

- ADT as a whole
- emergent behaviour
Classical Specification vs Holistic Specification

- fine-grained
- per function

- ADT as a whole
- emergent behaviour

Which is “stronger”? “Closed” ADT with classical spec implies holistic spec. (closed: no functions can be added, all functions have classical specs, ghost state has known representation)
Classical Specification vs Holistic Specification

- fine-grained
- per function

- ADT as a whole
- emergent behaviour

**Which is “stronger”?**

“Closed” ADT with classical spec implies holistic spec.
(closed: no functions can be added, all functions have classical specs, ghost state has known representation)

**Why do we need holistic specs?**

* “closed ADT” is sometimes too strong a requirement.
* Holistic aspect is cross-cutting (eg no payment without authorization)
* Allows reasoning in open world (eg DOM wrappers)
Thank you