What Do Relational Properties Have to Say About Legal Experts Systems?

Arthur Correnson - ProLaLa @ POPL 2023

CISPA
HELMHOLTZ CENTER FOR INFORMATION SECURITY

Context And Motivations

- Software systems are increasingly used in legal procedures
    - computing taxes
    - attributing social allowances
    - establishing contracts
    - E-voting

Context And Motivations

- Software systems are increasingly used in legal procedures
  - computing taxes
  - attributing social allowances
  - establishing contracts
  - E-voting

- What about the correctness of such software systems?
  - Do they reflect the law as stated in books?
  - Are they *fair* and *secure*?

On the Correctness of Software Systems

On the Correctness of Software Systems

- Functional correctness is a common question in program verification

$$\forall input, \texttt{prog}(input) = expected\_result(input)$$

On the Correctness of Software Systems

- Functional correctness is a common question in program verification

$$\forall input, \mathtt{prog}(input) = \boxed{expected\_result}(input)$$

On the Correctness of Legal Systems

- Functional correctness is a common question in program verification

$$\forall input, \texttt{prog}(input) = \boxed{expected\_result}(input)$$

- Difficult to apply in the context of Legal Experts systems
  - No clear formal specification of the correctness

On the Correctness of Legal Systems

- Functional correctness is a common question in program verification:

$$\forall input, \mathtt{prog}(input) = expected\_result(input)$$

- Difficult to apply in the context of Legal Experts systems
  - No clear formal specification of the correctness

How to assess the reliability of legal systems without a clear specification ?

Finding Other Correctness Criterion

Finding Other Correctness Criterion

Initial Problem: No clear 1-input vs 1-output specification

Finding Other Correctness Criterion

Initial Problem: No clear 1-input vs 1-output specification

↓

Solution: compare several outcomes of the program on different inputs instead

Finding Other Correctness Criterion

Initial Problem: No clear 1-input vs 1-output specification

↓

Solution: compare several outcomes of the program on different inputs instead

↓

Relational Specifications

Relational Properties

The example of taxes computation

Relational Properties

The example of taxes computation

> *For two individuals that differ only in age, the federal tax return of the older individual must be greater than or equal to that of the younger one.*
>
> — *From Metamorphic Testing and Debugging of Tax Preparation Software*
>
> Saeid Tizpaz-Niari, Morgan Wagner, Shiva Darian, Krystia Reed, Ashutosh Trivedi

Relational Properties

The example of taxes computation

For two individuals that differ only in age, the federal tax return of the older individual must be greater than or equal to that of the younger one.

— *From Metamorphic Testing and Debugging of Tax Preparation Software*

Saeid Tizpaz-Niari, Morgan Wagner, Shiva Darian, Krystia Reed, Ashutosh Trivedi

If the incomes of a household increase […], then its benefits will decrease […] and its income tax will increase […]

— *From Turning Catala Into a Proof Platform for the Law*

Alain Delaët, Denis Merigoux, Aymeric Fromherz

Relational Properties

In general

Different names depending on the field of application

- Relational Program Properties
- Metamorphic properties
- Hyper Properties

Common characteristics

- Compare several executions of the same system
- **Significantly** more expansive to verify than classical program properties

Relational Properties

In general

Different names depending on the field of application

- Relational Program Properties
- Metamorphic properties
- Hyper Properties

Common characteristics

- Compare several executions of the same system
- **Significantly** more expansive to verify than classical program properties

How to verify such properties ?

Verifying Relational Properties

using statistical testing

Informal Property

For two individuals that differ only in age, the federal tax return of the older individual must be greater than or equal to that of the younger one.

— *From Metamorphic Testing and Debugging of Tax Preparation Software*

Saeid Tizpaz-Niari, Morgan Wagner, Shiva Darian, Krystia Reed, Ashutosh Trivedi

Verifying Relational Properties

using statistical testing

Informal Property

For two individuals that differ only in age, the federal tax return of the older individual must be greater than or equal to that of the younger one.

— From Metamorphic Testing and Debugging of Tax Preparation Software

Saeid Tizpaz-Niari, Morgan Wagner, Shiva Darian, Krystia Reed, Ashutosh Trivedi

Formal Statement

$$\forall x \forall y, x =_{\texttt{age}} y \wedge x.\texttt{age} < 65 \wedge y.\texttt{age} \geq 65 \Rightarrow \texttt{F}(x) \leq \texttt{F}(y)$$

Verifying Relational Properties

using statistical testing

Informal Property

For two individuals that differ only in age, the federal tax return of the older individual must be greater than or equal to that of the younger one.

— From Metamorphic Testing and Debugging of Tax Preparation Software

Saeid Tizpaz-Niari, Morgan Wagner, Shiva Darian, Krystia Reed, Ashutosh Trivedi

Formal Statement

$$\forall x \forall y, x =_{\texttt{age}} y \wedge x.\texttt{age} < 65 \wedge y.\texttt{age} \geq 65 \Rightarrow \texttt{F}(x) \leq \texttt{F}(y)$$

Methodology

- Try to find inputs x and y that violates the relation
- The search space is explored using statistical methods

Verifying Relational Properties

using statistical testing

Informal Property

For two individuals that differ only in age, the federal tax return of the older individual must be greater than or equal to that of the younger one.

— *From Metamorphic Testing and Debugging of Tax Preparation Software*

Saeid Tizpaz-Niari, Morgan Wagner, Shiva Darian, Krystia Reed, Ashutosh Trivedi

Formal Statement

$$\forall x \forall y, x =_{\texttt{age}} y \land x.\texttt{age} < 65 \land y.\texttt{age} \geq 65 \Rightarrow \texttt{F}(x) \leq \texttt{F}(y)$$

How to prove the absence of *bad* inputs ?

Proving Relational Properties

by symbolic execution

$$\forall x \forall y, x =_{\texttt{age}} y \wedge x.\texttt{age} < 65 \wedge y.\texttt{age} \geq 65 \Rightarrow \texttt{F}(x) \leq \texttt{F}(y)$$

Proving Relational Properties

by symbolic execution

$$\forall x \forall y, x =_{\texttt{age}} y \land x.\texttt{age} < 65 \land y.\texttt{age} \geq 65 \Rightarrow \texttt{F}(x) \leq \texttt{F}(y)$$

```
def F(user):
  if user.incomes <= t:
    if user.age >= 65:
      A(user)
    else:
      B(user)
   else:
      C(user)
```

Proving Relational Properties

by symbolic execution

$$\forall x \forall y, x =_{\texttt{age}} y \wedge x.\texttt{age} < 65 \wedge y.\texttt{age} \geq 65 \Rightarrow \texttt{F}(x) \leq \texttt{F}(y)$$

```
def F(user):
  if user.incomes <= t:
    if user.age >= 65:
      A(user)
    else:
      B(user)
   else:
      C(user)
```

Possible Outcomes:

$\texttt{A}(x)$ if $x.\texttt{incomes} \leq t \wedge x.\texttt{age} \geq 65$

$\texttt{B}(x)$ if $x.\texttt{incomes} \leq t \wedge x.\texttt{age} < 65$

$\texttt{C}(x)$ if $x.\texttt{incomes} > t$

Proving Relational Properties

by symbolic execution

$$\forall x \forall y, \underbrace{x =_{\texttt{age}} y \land x.\texttt{age} < 65 \land y.\texttt{age} \geq 65}_{\varphi(x, y)} \Rightarrow \texttt{F}(x) \leq \texttt{F}(y)$$

Expected relation between inputs

```python
def F(user):
  if user.incomes <= t:
    if user.age >= 65:
      A(user)
    else:
      B(user)
   else:
      C(user)
```

Possible Outcomes:

$\texttt{A}(x)$ if $x.\texttt{incomes} \leq t \land x.\texttt{age} \geq 65$

$\texttt{B}(x)$ if $x.\texttt{incomes} \leq t \land x.\texttt{age} < 65$

$\texttt{C}(x)$ if $\underbrace{x.\texttt{incomes} > t \qquad\qquad\qquad}_{\pi(x)}$

Preconditions to reach each outcome

Proving Relational Properties

by symbolic execution

$$\forall x \forall y, x =_{\texttt{age}} y \land x.\texttt{age} < 65 \land y.\texttt{age} \geq 65 \Rightarrow \texttt{F}(x) \leq \texttt{F}(y)$$

Possible violations:

(1) $\exists x \exists y, \varphi(x, y) \land \pi_{\texttt{A}}(x) \land \pi_{\texttt{A}}(y) \land \texttt{A}(x) < \texttt{A}(y)$

$\lor$

(2) $\exists x \exists y, \varphi(x, y) \land \pi_{\texttt{A}}(x) \land \pi_{\texttt{B}}(y) \land \texttt{A}(x) < \texttt{B}(y)$

$\lor$

(3) $\exists x \exists y, \varphi(x, y) \land \pi_{\texttt{A}}(x) \land \pi_{\texttt{C}}(y) \land \texttt{A}(x) < \texttt{C}(y)$

(…) $\lor$

(9) $\exists x \exists y, \varphi(x, y) \land \pi_{\texttt{C}}(x) \land \pi_{\texttt{C}}(y) \land \texttt{C}(x) < \texttt{C}(y)$

Proving Relational Properties

by symbolic execution

$$\forall x \forall y, x =_{\mathtt{age}} y \wedge x.\mathtt{age} < 65 \wedge y.\mathtt{age} \geq 65 \Rightarrow \mathtt{F}(x) \leq \mathtt{F}(y)$$

Possible violations:

(1) $\exists x \exists y, \varphi(x,y) \wedge \pi_{\mathtt{A}}(x) \wedge \pi_{\mathtt{A}}(y) \wedge \mathtt{A}(x) < \mathtt{A}(y)$

$\vee$

(2) $\exists x \exists y, \varphi(x,y) \wedge \pi_{\mathtt{A}}(x) \wedge \pi_{\mathtt{B}}(y) \wedge \mathtt{A}(x) < \mathtt{B}(y)$

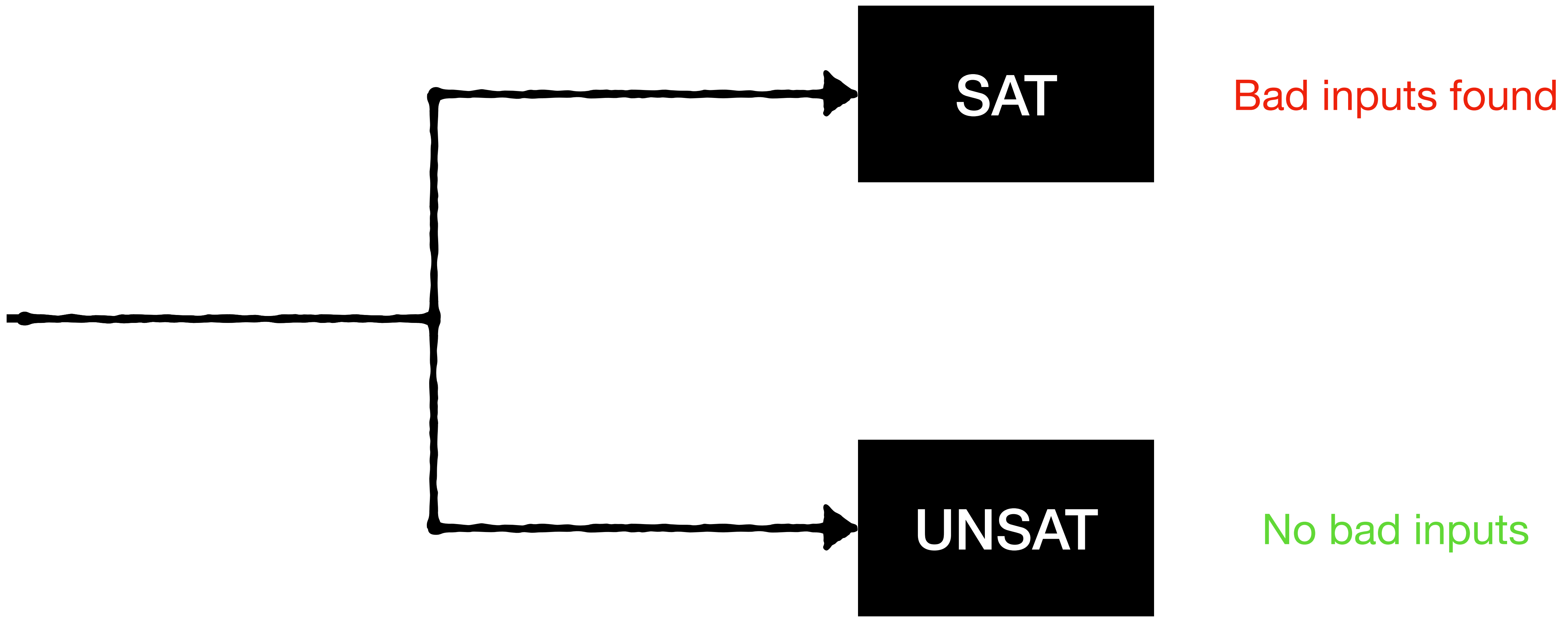
$\vee$

(3) $\exists x \exists y, \varphi(x,y) \wedge \pi_{\mathtt{A}}(x) \wedge \pi_{\mathtt{C}}(y) \wedge \mathtt{A}(x) < \mathtt{C}(y)$

(…) $\vee$

(9) $\exists x \exists y, \varphi(x,y) \wedge \pi_{\mathtt{C}}(x) \wedge \pi_{\mathtt{C}}(y) \wedge \mathtt{C}(x) < \mathtt{C}(y)$

Automated Provers



SAT

Bad inputs found

UNSAT

No bad inputs

What Next ?

Develop a specification language to write relational properties as encountered in the context of legal systems

- simple syntax, built-in abstractions specific to the application field

What Next ?

Develop a specification language to write relational properties as encountered in the context of legal systems

- simple syntax, built-in abstractions specific to the application field

Integrate an automated verifier for relational properties inside the domain specific programming language **CATALA**

- Hopefully much more efficient than targeting a general purpose language

What Next ?

Develop a specification language to write relational properties as encountered in the context of legal systems

- simple syntax, built-in abstractions specific to the application field

Integrate an automated verifier for relational properties inside the domain specific programming language **CATALA**

- Hopefully much more efficient than targeting a general purpose language

Investigate other forms of relational properties

- Temporal Hyper Properties to specify *fairness* or *non-interference* properties
  e.g. in the context of e-voting platforms