Programming with Invariants through Targeted Synthesis

John Sarracino
Programming is Hard
Programming is Hard

Therac 25 — 1982
Programming is Hard

Ariane 5 —1996
Programming is Hard

We are sorry but maerskline.com is temporarily unavailable

We confirm that some Maersk IT systems are down. We are assessing the situation. The safety of your business and our people is our top priority. We will update when we have more information.

We apologize for any inconvenience this causes you.

Maersk Line team
Security

It's March 2018, and your Windows PC can be pwned by a web article (well, none of OURS)

Windows — 2018
Programming is Hard

Security

It's March 2018, and your Windows PC can be pwned by a web article (well, none of OURS)

Windows — always
Computers are a Sadness and I am the cure
Programs are a Sadness and I am the cure
Software Invariants

ubi_assert(e->num != e1->num);

if (keyProto.getKeyValue().size() != Ed25519Verify.PUBLIC_KEY_LEN) {
    throw new GeneralSecurityException("invalid Ed25519 public key: incorrect key length");
}

ensure
    cat_set : has_category(cat)
end
Software Invariants

```
ubi_assert(e->num != e1->num);

if (keyProto.getKeyValue().size() != Ed25519Verify.PUBLIC_KEY_LEN) {
    throw new GeneralSecurityException("invalid Ed25519 public key: incorrect key length");
}

ensure
    cat_set : has_category(cat)
end
```
Software Invariants

\texttt{ubi\_assert(e->num} !\texttt{=} e1->num); 

\texttt{if} (keyProto.getKeyValue().size() \texttt{!=} Ed25519Verify.PUBLIC\_KEY\_LEN) { 
  \texttt{throw new}
  \texttt{GeneralSecurityException("invalid Ed25519}
  \texttt{public} \texttt{key: incorrect key length");} 
}

\texttt{ensure}
  \texttt{cat\_set : has\_category(cat)}
\texttt{end}
Software Invariants

C

```c
ubi_assert(e->num != e1->num);
```

Java

```java
if (keyProto.getKeyValue().size() != Ed25519Verify.PUBLIC_KEY_LEN) {
    throw new GeneralSecurityException("invalid Ed25519 public key: incorrect key length");
}
```

Eiffel

```eiffel
ensure
    cat_set : has_category(cat)
end
```
Software Invariants

ubi_assert(e->num != e1->num);

if (keyProto.getKeyValue().size() != Ed25519Verify.PUBLIC_KEY_LEN)
    throw new GeneralSecurityException("invalid Ed25519 public key: incorrect key length");

ensure
cat_set : has_category(cat)
end

Language agnostic, Manually maintained.
Manual Invariant Programming

Invariants

Algorithm
Manual Invariant Programming

Invariants

Algorithm

Code
Why are Manual Invariants difficult?

Code changes =>
Programmer checks, maintains invariant

Invariant changes =>
Programmer checks, maintains code
The code change problem: Maintaining Invariants

```
for i in people.data.users:
    response = client.api.statuses.user_timeline.get(screen_name=i.screen_name)
    print 'Got', len(response.data), 'tweets from', i.screen_name
    if len(response.data) != 0:
        ldate = response.data[0]['created_at']
        ldate2 = datetime.strptime(ldate, '%a %b %d %H:%M:%S +0000 %Y')
        today = datetime.now()
        howlong = (today-ldate2).days
        if howlong < daywindow:
            print i.screen_name, 'has tweeted in the past', daywindow,
            totaltweets += len(response.data)
            for j in response.data:
                if j.entities.urls:
                    ...
else:
    print i.screen_name, 'has not tweeted in the past', daywindow
```

Invariant: “All urls are in the urlset”
The code change problem: Maintaining Invariants

```
for i in people.data.users:
    response = client.api.statuses.user_timeline.get(screen_name=i.screen_name)
    print 'Got', len(response.data), 'tweets from', i.screen_name
    if len(response.data) != 0:
        ldate = response.data[0]['created_at']
        ldate2 = datetime.strptime(ldate, '%a %b %d %H:%M:%S +0000 %Y')
        today = datetime.now()
        howlong = (today-ldate2).days
        if howlong < daywindow:
            print i.screen_name, 'has tweeted in the past', daywindow,
            totaltweets += len(response.data)
            for j in response.data:
                if j.entities.urls:
                    ... # code
                else:
                    print i.screen_name, 'has not tweeted in the past', daywindow
```

Invariant: “All urls are in the urlset”
The code change problem: Maintaining Invariants

```
for i in people.data.users:
    response = client.api.statuses.user_timeline.get(screen_name=i.screen_name)
    print 'Got', len(response.data), 'tweets from', i.screen_name
    if len(response.data) != 0:
        ldate = response.data[0]['created_at']
        ldate2 = datetime.strptime(ldate, '%a %b %d %H:%M:%S +0000 %Y')
        today = datetime.now()
        howlong = (today-ltdate2).days
        if howlong < daywindow:
            print i.screen_name, 'has tweeted in the past', daywindow,
            totaltweets += len(response.data)
        for j in response.data:
            if j.entities.urls:
                for k in j.entities.urls:
                    newurl = k['expanded_url']
            else:
                print i.screen_name, 'has not tweeted in the past', daywindow
```

Invariant: “All urls are in the urlset”
The code change problem: Maintaining Invariants

Tedious and error-prone, particularly with a large codebase

Invariant: “All urls are in the urlset”
The invariant change problem: Maintaining Code

```
for i in people.data.users:
    response = client.api.statuses.user_timeline.get(screen_name=i.screen_name)
    print 'Got', len(response.data), 'tweets from', i.screen_name
    if len(response.data) != 0:
        ltdate = response.data[0]['created_at']
        ltdate2 = datetime.strptime(ltdate, '%a %b %d %H:%M:%S +0000 %Y')
        today = datetime.now()
        howlong = (today-ltdate2).days
        if howlong < daywindow:
            print i.screen_name, 'has tweeted in the past', daywindow,
            tottweets += len(response.data)
            for j in response.data:
                if j.entities.urls:
                    for k in j.entities.urls:
                        newurl = k['expanded_url']
                        urlset.add((newurl, j.user.screen_name))
                else:
                    print i.screen_name, 'has not tweeted in the past', daywindow
```

Invariant: “All urls are in the urlset”
The invariant change problem: Maintaining Code

```
for i in people.data.users:
    response = client.apistatuses.user_timeline.get(screen_name=i.screen_name)
    print 'Got', len(response.data), 'tweets from', i.screen_name
    if len(response.data) != 0:
        ldate = response.data[0]['created_at']
        ldate2 = datetime.strptime(ldate, '%a %b %d %H:%M:%S %0000 %Y')
        today = datetime.now()
        howlong = (today-ldate2).days
        if howlong < daywindow:
            print i.screen_name, 'has tweeted in the past', daywindow,
            tottweets += len(response.data)
            for j in response.data:
                if j.entities.urls:
                    for k in j.entities.urls:
                        newurl = k['expanded_url']
                        urlset.add((newurl, j.user.screen_name))
else:
    print i.screen_name, 'has not tweeted in the past', daywindow
```

Invariant:

“All urls are in the urlset”

“All urls are in the urlset”

“The allocset contains the urlset ”
The invariant change problem: Maintaining Code

```
for i in people.data.users:
    response = client.api.statuses.user_timeline.get(screen_name=i.screen_name)
    print 'Got', len(response.data), 'tweets from', i.screen_name
    if len(response.data) != 0:
        ldate = response.data[0]['created_at']
        print i.screen_name, 'has not tweeted in the past 7 days!
```

Even more tedious and error-prone

```
Invariant:
"All urls are in the urlset"
"The allocset contains the urlset"
```

My thesis statement

Compiling and generating invariants is useful, novel, and broadly applicable.
My thesis proposal

Compiling and generating invariants is useful, novel, and broadly applicable.

I propose a general framework for maintaining invariants.
Proposed Work

- Data invariant calculus
- Extensions
- Invariants for Security
- GUI invariants
Outline

- Data invariant calculus
- Extensions
- Invariants for Security
- GUI invariants
Outline

- Data invariant calculus
- Extensions
- Invariants for Security
- GUI invariants
### Budgeting Example

#### Weekly, Monthly, Yearly

<table>
<thead>
<tr>
<th>Weekly</th>
<th>Monthly</th>
<th>Yearly</th>
</tr>
</thead>
<tbody>
<tr>
<td>-200</td>
<td>-800</td>
<td>-9600</td>
</tr>
<tr>
<td>-50</td>
<td>-200</td>
<td>-2400</td>
</tr>
<tr>
<td>750</td>
<td>3000</td>
<td>36000</td>
</tr>
<tr>
<td>-45</td>
<td>-180</td>
<td>-2160</td>
</tr>
</tbody>
</table>

#### Invariants

\[ \forall i.4 \times \text{weeks}[i] = \text{months}[i] \]
\[ \forall i.12 \times \text{months}[i] = \text{years}[i] \]
# Budgeting Example

## Invariants

\[ \forall i. 4 \times \text{weeks}[i] = \text{months}[i] \]
\[ \forall i. 12 \times \text{months}[i] = \text{years}[i] \]

<table>
<thead>
<tr>
<th>Weekly</th>
<th>Monthly</th>
<th>Yearly</th>
</tr>
</thead>
<tbody>
<tr>
<td>-200</td>
<td>-800</td>
<td>-9600</td>
</tr>
<tr>
<td>-50</td>
<td>-200</td>
<td>-2400</td>
</tr>
<tr>
<td>750</td>
<td>3000</td>
<td>36000</td>
</tr>
<tr>
<td>-45</td>
<td>-180</td>
<td>-2160</td>
</tr>
</tbody>
</table>

Color of Odd Rows

Color of Even Rows
# Budgeting Example

<table>
<thead>
<tr>
<th>Weekly</th>
<th>Monthly</th>
<th>Yearly</th>
</tr>
</thead>
<tbody>
<tr>
<td>-400</td>
<td>-1600</td>
<td>-19200</td>
</tr>
<tr>
<td>-50</td>
<td>-200</td>
<td>-2400</td>
</tr>
<tr>
<td>750</td>
<td>3000</td>
<td>36000</td>
</tr>
<tr>
<td>-45</td>
<td>-180</td>
<td>-2160</td>
</tr>
</tbody>
</table>

**Color of Odd Rows**

**Color of Even Rows**

**Invariants**

\[ \forall i. 4 \times weeks[i] = months[i] \]

\[ \forall i. 12 \times months[i] = years[i] \]
Budgeting Example

### Invariants

\[ \forall i. 4 \times \text{weeks}[i] = \text{months}[i] \]
\[ \forall i. 12 \times \text{months}[i] = \text{years}[i] \]

\[ \forall i. \text{weeks}[i] < 0 \implies \text{font}[i] = \text{red} \]
\[ \forall i. \text{weeks}[i] \geq 0 \implies \text{font}[i] = \text{black} \]

<table>
<thead>
<tr>
<th>Weekly</th>
<th>Monthly</th>
<th>Yearly</th>
</tr>
</thead>
<tbody>
<tr>
<td>-400</td>
<td>-1600</td>
<td>-19200</td>
</tr>
<tr>
<td>-50</td>
<td>-200</td>
<td>-2400</td>
</tr>
<tr>
<td>750</td>
<td>3000</td>
<td>36000</td>
</tr>
<tr>
<td>-45</td>
<td>-180</td>
<td>-2160</td>
</tr>
</tbody>
</table>

Color of Odd Rows

Color of Even Rows
# Budgeting Example

<table>
<thead>
<tr>
<th>Weekly</th>
<th>Monthly</th>
<th>Yearly</th>
</tr>
</thead>
<tbody>
<tr>
<td>-400</td>
<td>-1600</td>
<td>-19200</td>
</tr>
<tr>
<td>-50</td>
<td>-200</td>
<td>-2400</td>
</tr>
<tr>
<td>750</td>
<td>3000</td>
<td>36000</td>
</tr>
<tr>
<td>-45</td>
<td>-180</td>
<td>-2160</td>
</tr>
</tbody>
</table>

## Invariants

\[
\forall i. 4 \times weeks[i] = months[i] \\
\forall i. 12 \times months[i] = years[i]
\]

\[
\forall i. weeks[i] < 0 \implies font[i] = red \\
\forall i. weeks[i] \geq 0 \implies font[i] = black
\]
## Budgeting Example

<table>
<thead>
<tr>
<th>Weekly</th>
<th>Monthly</th>
<th>Yearly</th>
</tr>
</thead>
<tbody>
<tr>
<td>-400</td>
<td>-1600</td>
<td>-19200</td>
</tr>
<tr>
<td>25</td>
<td>100</td>
<td>1200</td>
</tr>
<tr>
<td>750</td>
<td>3000</td>
<td>36000</td>
</tr>
<tr>
<td>-45</td>
<td>-180</td>
<td>-2160</td>
</tr>
</tbody>
</table>

### Invariants

- \( \forall i. 4 \times \text{weeks}[i] = \text{months}[i] \)
- \( \forall i. 12 \times \text{months}[i] = \text{years}[i] \)

- \( \forall i. \text{weeks}[i] < 0 \implies \text{font}[i] = \text{red} \)
- \( \forall i. \text{weeks}[i] \geq 0 \implies \text{font}[i] = \text{black} \)

Color of Odd Rows

Color of Even Rows
## Budgeting Example

<table>
<thead>
<tr>
<th>Weekly</th>
<th>Monthly</th>
<th>Yearly</th>
</tr>
</thead>
<tbody>
<tr>
<td>-400</td>
<td>-1600</td>
<td>-19200</td>
</tr>
<tr>
<td>25</td>
<td>100</td>
<td>1200</td>
</tr>
<tr>
<td>750</td>
<td>3000</td>
<td>36000</td>
</tr>
<tr>
<td>-45</td>
<td>-180</td>
<td>-2160</td>
</tr>
</tbody>
</table>

### Color of Odd Rows

- Gray

### Color of Even Rows

- White

### Invariants

\[ \forall i. 4 \times \text{weeks}[i] = \text{months}[i] \]
\[ \forall i. 12 \times \text{months}[i] = \text{years}[i] \]
\[ \forall i. \text{weeks}[i] < 0 \implies \text{font}[i] = \text{red} \]
\[ \forall i. \text{weeks}[i] \geq 0 \implies \text{font}[i] = \text{black} \]
\[ \forall i. i \mod 2 = 0 \implies \text{bkg}[i] = \text{even} \]
\[ \forall i. i \mod 2 = 1 \implies \text{bkg}[i] = \text{odd} \]
Budgeting Example

<table>
<thead>
<tr>
<th>Weekly</th>
<th>Monthly</th>
<th>Yearly</th>
</tr>
</thead>
<tbody>
<tr>
<td>-400</td>
<td>-1600</td>
<td>-19200</td>
</tr>
<tr>
<td>25</td>
<td>100</td>
<td>1200</td>
</tr>
<tr>
<td>750</td>
<td>3000</td>
<td>36000</td>
</tr>
<tr>
<td>-45</td>
<td>-180</td>
<td>-2160</td>
</tr>
</tbody>
</table>

Color of Odd Rows

Color of Even Rows

Invariants

\[\forall i.4 \times \text{weeks}[i] = \text{months}[i]\]
\[\forall i.12 \times \text{months}[i] = \text{years}[i]\]

\[\forall i.\text{weeks}[i] < 0 \implies \text{font}[i] = \text{red}\]
\[\forall i.\text{weeks}[i] \geq 0 \implies \text{font}[i] = \text{black}\]

\[\forall i.i \mod 2 = 0 \implies \text{bkg}[i] = \text{even}\]
\[\forall i.i \mod 2 = 1 \implies \text{bkg}[i] = \text{odd}\]
## Budgeting Example

### Weekly vs. Monthly vs. Yearly

<table>
<thead>
<tr>
<th>Weekly</th>
<th>Monthly</th>
<th>Yearly</th>
</tr>
</thead>
<tbody>
<tr>
<td>-400</td>
<td>-1600</td>
<td>-19200</td>
</tr>
<tr>
<td>25</td>
<td>100</td>
<td>1200</td>
</tr>
<tr>
<td>750</td>
<td>3000</td>
<td>36000</td>
</tr>
<tr>
<td>-45</td>
<td>-180</td>
<td>-2160</td>
</tr>
</tbody>
</table>

### Invariants

\[
\forall i. 4 \times \text{weeks}[i] = \text{months}[i]
\]

\[
\forall i. 12 \times \text{months}[i] = \text{years}[i]
\]

\[
\forall i. \text{weeks}[i] < 0 \implies \text{font}[i] = \text{red}
\]

\[
\forall i. \text{weeks}[i] \geq 0 \implies \text{font}[i] = \text{black}
\]

\[
\forall i. i \mod 2 = 0 \implies \text{bkg}[i] = \text{even}
\]

\[
\forall i. i \mod 2 = 1 \implies \text{bkg}[i] = \text{odd}
\]
Budgeting Code

```javascript
for (let index = 0; index != week.length; ++index) {
    week[index].oninput = () => {
        let w = parseFloat(week[index].textContent);
        month[index].textContent = w * 4;
        year[index].textContent = w * 12 * 4;
        refreshColors(index);
    }
}

month[index].oninput = () => {
    let m = parseFloat(month[index].textContent);
    week[index].textContent = m / 4;
    year[index].textContent = m * 12;
    refreshColors(index);
}

year[index].oninput = () => {
    let y = parseFloat(year[index].textContent);
    week[index].textContent = y / (4 * 12);
    month[index].textContent = y / 12;
    refreshColors(index);
}
```

```javascript
document.getElementById('evenPicker').oninput = () => {
    evenColor = document.getElementById('evenPicker').value;
    for (let index = 0; index != week.length; ++index) {
        if (index % 2 == 0) {
            week[index].style.backgroundColor = evenColor;
            month[index].style.backgroundColor = evenColor;
            year[index].style.backgroundColor = evenColor;
        } else {
            week[index].style.backgroundColor = oddColor;
            month[index].style.backgroundColor = oddColor;
            year[index].style.backgroundColor = oddColor;
        }
    }
}
```

```javascript
document.getElementById('oddPicker').oninput = () => {
    oddColor = document.getElementById('oddPicker').value;
    for (let index = 0; index != week.length; ++index) {
        if (index % 2 == 0) {
            week[index].style.backgroundColor = evenColor;
            month[index].style.backgroundColor = evenColor;
            year[index].style.backgroundColor = evenColor;
        } else {
            week[index].style.backgroundColor = oddColor;
            month[index].style.backgroundColor = oddColor;
            year[index].style.backgroundColor = oddColor;
        }
    }
}
```
Budgeting Code

Tedious even for this simple example
for (let index = 0; index != week.length; ++index) {
    week[index].oninput = () => {
        let w = parseFloat(week[index].textContent);
        month[index].textContent = w * 4;
        year[index].textContent = w * 12 * 4;
        refreshColors(index);
    }
}

month[index].oninput = () => {
    let m = parseFloat(month[index].textContent);
    week[index].textContent = m / 4;
    year[index].textContent = m * 12;
    refreshColors(index);
}

year[index].oninput = () => {
    let y = parseFloat(year[index].textContent);
    week[index].textContent = y / (4 * 12);
    month[index].textContent = y / 12;
    refreshColors(index);
}

document.getElementById('evenPicker').oninput = () => {
    evenColor = document.getElementById('evenPicker').value;
    for (let index = 0; index != week.length; ++index) {
        if (index % 2 === 0) {
            week[index].style.backgroundColor = evenColor;
            month[index].style.backgroundColor = evenColor;
            year[index].style.backgroundColor = evenColor;
        } else {
            week[index].style.backgroundColor = oddColor;
            month[index].style.backgroundColor = oddColor;
            year[index].style.backgroundColor = oddColor;
        }
    }
}

document.getElementById('oddPicker').oninput = () => {
    oddColor = document.getElementById('oddPicker').value;
    for (let index = 0; index != week.length; ++index) {
        if (index % 2 === 0) {
            week[index].style.backgroundColor = evenColor;
            month[index].style.backgroundColor = evenColor;
            year[index].style.backgroundColor = evenColor;
        } else {
            week[index].style.backgroundColor = oddColor;
            month[index].style.backgroundColor = oddColor;
            year[index].style.backgroundColor = oddColor;
        }
    }
}
Budgeting Code

Invariants

Algorithm

```javascript
for (let index = 0; index !== week.length; ++index) {
  week[index].oninput = () => {
    let w = parseFloat(week[index].textContent);
    month[index].textContent = w * 4;
    year[index].textContent = w * 12 * 4;
    refreshColors(index);
  }
}

month[index].oninput = () => {
  let m = parseFloat(month[index].textContent);
  week[index].textContent = m / 4;
  year[index].textContent = m * 12;
  refreshColors(index);
}

year[index].oninput = () => {
  let y = parseFloat(year[index].textContent);
  week[index].textContent = y / (4 * 12);
  month[index].textContent = y / 12;
  refreshColors(index);
}
```

```javascript
document.getElementById('evenPicker').oninput = () => {
  evenColor = document.getElementById('evenPicker').value;
  for (let index = 0; index !== week.length; ++index) {
    if (index % 2 === 0) {
      week[index].style.backgroundColor = evenColor;
      month[index].style.backgroundColor = evenColor;
      year[index].style.backgroundColor = evenColor;
    } else {
      week[index].style.backgroundColor = oddColor;
      month[index].style.backgroundColor = oddColor;
      year[index].style.backgroundColor = oddColor;
    }
  }
}
```

```javascript
document.getElementById('oddPicker').oninput = () => {
  oddColor = document.getElementById('oddPicker').value;
  for (let index = 0; index !== week.length; ++index) {
    if (index % 2 === 0) {
      week[index].style.backgroundColor = evenColor;
      month[index].style.backgroundColor = evenColor;
      year[index].style.backgroundColor = evenColor;
    } else {
      week[index].style.backgroundColor = oddColor;
      month[index].style.backgroundColor = oddColor;
      year[index].style.backgroundColor = oddColor;
    }
  }
}
```
Budgeting Code

Invariants

Vast majority of code manually maintains invariant

Algorithm
Manual Invariant Programming

Invariants

Algorithm

Code
Invariant Synthesis

Invariants

Algorithm

Code
Language Design
Goals

Programmer thinks about invariants once

Compiler automatically handles invariants
Compiling Invariants

*Program Synthesis* compiles invariants*

**Inputs:**

1) program with holes e.g. \( x := \text{??}(y, x) \)
2) precondition e.g. \( y = 15 \)
3) requirement e.g. \( x = y \)
Compiling Invariants

Program Synthesis compiles invariants*

Inputs:

1) program with holes e.g. \(x := \_\_\_ (y, x)\)

2) precondition e.g. \(y = 15\)

3) requirement e.g. \(x = y\)

“expression using \(x\) and \(y\)” \(x := \_\_\_ (y, x)\)
Language Design Goals

Programmer thinks about invariants once

Compiler automatically handles invariants
Naive Algorithm

Synthesize at the end of every function.

Use invariants as preconditions, requirements.
Naive Algorithm

```javascript
function updateWeeks(value, i) {
    weeks[i] = value;
}
```

Pre

\[\forall i.4 \times weeks[i] = months[i]\]
\[\forall i.12 \times months[i] = years[i]\]

 Req

\[\forall i.4 \times weeks[i] = months[i]\]
\[\forall i.12 \times months[i] = years[i]\]
Naive Algorithm

```javascript
function updateWeeks(value, i) {
    weeks[i] = value;
}
```

Pre

\[
\forall i. 4 \times weeks[i] = months[i] \\
\forall i. 12 \times months[i] = years[i]
\]

Req

\[
\forall i. 4 \times weeks[i] = months[i] \\
\forall i. 12 \times months[i] = years[i]
\]
Naive Algorithm

function updateWeeks(value, i) {
    weeks[i] = value;
}

Does not scale to realistic programs

Pre

\forall i.4 \times \text{weeks}[i] = \text{months}[i]
\forall i.12 \times \text{months}[i] = \text{years}[i]

Req

\forall i.4 \times \text{weeks}[i] = \text{months}[i]
\forall i.12 \times \text{months}[i] = \text{years}[i]
Problem: Loops don’t scale

function adjustForInflation(scale) {
    for (let i = 0; i < weeks.length; ++i) {
        weeks[i] = weeks[i] * scale;
    }
}

---

Pre

∀i.4 * weeks[i] = months[i]
∀i.12 * months[i] = years[i]

Req

∀i.4 * weeks[i] = months[i]
∀i.12 * months[i] = years[i]

for (let i = 0; i < weeks.length; ++i) {
    weeks[i] = weeks[i] * scale;
}

for (let i = 0; i < weeks.length; ++i) {
    months[i] = ??(scale, i, weeks[i]);
    years[i] = ??(scale, i, weeks[i]);
}
Problem:
Loops don’t scale

function adjustForInflation(scale) {
  for (let i = 0; i < weeks.length; ++i) {
    weeks[i] = weeks[i] * scale;
  }
}

for (let i = 0; i < weeks.length; ++i) {
  weeks[i] = weeks[i] * scale;
}

Literal equality for this simple example
Key insight: move hole inside loop

```javascript
function adjustForInflation(scale) {
  for (let i = 0; i < weeks.length; ++i) {
    weeks[i] = weeks[i] * scale;
  }
}
```

Pre

\[
\forall i.4 \times \text{weeks}[i] = \text{months}[i]
\]

\[
\forall i.12 \times \text{months}[i] = \text{years}[i]
\]

Req

```javascript
for (let i = 0; i < weeks.length; ++i) {
  weeks[i] = weeks[i] * scale;
}
```
Key insight: move hole inside loop

```javascript
function adjustForInflation(scale) {
  for (let i = 0; i < weeks.length; ++i) {
    weeks[i] = weeks[i] * scale;
  }
}
```

Pre

\[ \forall i.4 \ast weeks[i] = months[i] \]
\[ \forall i.12 \ast months[i] = years[i] \]

Req

\[ \forall i.4 \ast weeks[i] = months[i] \]
\[ \forall i.12 \ast months[i] = years[i] \]
Simplification: No quantification!!

```javascript
function adjustForInflation(scale) {
  for (let i = 0; i < weeks.length; ++i) {
    weeks[i] = weeks[i] * scale;
  }
}
```

Pre

∀i.4 * weeks[i] = months[i]
∀i.12 * months[i] = years[i]

Req

∀i.4 * weeks[i] = months[i]
∀i.12 * months[i] = years[i]
function adjustForInflation(scale) {
    for (let i = 0; i < weeks.length; ++i) {
        weeks[i] = weeks[i] * scale;
        months[i] = ??(scale, i, weeks[i]);
        years[i] = ??(scale, i, weeks[i]);
    }
}
Simplification: No quantification!!

function adjustForInflation(scale) {
    for (let i = 0; i < weeks.length; ++i) {
        weeks[i] = weeks[i] * scale;
    }
}

4 * weeks[i] = months[i]
12 * months[i] = years[i]

Pre

weeks[i] = weeks[i] * scale;

months[i] = ??(scale, i, weeks[i]);

years[i] = ??(scale, i, weeks[i]);

Req

4 * weeks[i] = months[i]
12 * months[i] = years[i]
Simplification: No quantification!!

function adjustForInflation(scale) {
  for (let i = 0; i < weeks.length; ++i) {
    weeks[i] = weeks[i] * scale;
    months[i] = ??(scale, i, weeks[i]);
    years[i] = ??(scale, i, weeks[i]);
  }
}

Targeted Synthesis!
Simplification:
No quantification!!

function adjustForInflation(scale) {
    for (let i = 0; i < weeks.length; ++i) {
        weeks[i] = weeks[i] * scale;
    }
}

Pre

4 * weeks[i] = months[i]
12 * months[i] = years[i]

Req

weeks[i] = weeks[i] * scale;
months[i] = months[i] * scale;
years[i] = years[i] * scale;

4 * weeks[i] = months[i]
12 * months[i] = years[i]
Spyder

Separate invariant, imperative programs

Compiler combines with targeted synthesis
Spyder

Automatic Invariant maintenance

Compiler combines with targeted synthesis
for (w: int) with idx in (weeks) {
    if (idx == i) {
        w = val;
    }
}

Imperative Language
for (w: int) with idx in (weeks) {
    if (idx == i) {
        w = val;
    }
}
for (w: int) with idx in (weeks) {
    if (idx == i) {
        w = val;
    }
}

Imperative Language

In-out iterators
Imperative AST

Block ::= Stmt₁ ... Stmtₙ
Stmt ::= Assign v Expr
| Cond Expr Block Block
| For (v₁ ... vₙ) [v] (v₁ ... vₙ) Block
| VarDecl (v, Ty)
Expr ::= v | i | true | false
| BinOp bop Expr Expr
| Index Expr Expr
| UnOp uop Expr
bop ::= +, ×, %, ⇒, , ⇔, ...
uop ::= ¬, !
Invariant Language

```
foreach (week, month) in (weeks, months) {
    week = 4 * month
}
```
Invariant Language

```plaintext
foreach (week, month) in (weeks, months) {
    week = 4 * month
}
```

Elementwise Quantification
Invariant AST

\[
\text{Inv} ::= \text{ForEach} \ (v_1 \ldots v_n) \ [v] \ (v_1 \ldots v_n) \ \text{Inv} \\
| \text{BaseExpr} \ \text{Expr}
\]
Compilation Algorithm
Language Design

Goals

Programmer thinks about invariants once

Compiler automatically handles invariants
Compilation Algorithm

1) bring necessary variables into scope
2) specialize invariants to loops
3) ??? (find template)
4) profit (synthesize)
Compilation Algorithm

1) bring necessary variables into scope
2) specialize invariants to loops
3) ??? (find template)
4) profit (synthesize)
Format

Invariants

Code

Explanatory Caption

Problem/Result
Simple Example

x > 2 * y

y = y + 1;
Simple Example

$x > 2 \times y$

$y = y + 1$

$X \sim Y$
Simple Example

x > 2 * y

y = y + 1;

X ~ Y

Y edited
Simple Example

$x > 2 \times y$

$y = y + 1$

$X \sim Y$

$Y$ edited

$x = ??(x, y)$
Simple Example

\[ x > 2 \ast y \]

\[ y = y + 1; \]

X, Y in scope

\[ x = \text{??}(x, y); \]
Simple Example

\[ x > 2 \times y \]

\[ y = y + 1; \]

\[ x = ??(x, y); \]
Simple Example

\[ x > 2 \times y \]

\[ y = y + 1; \]

\[ x = x + 2; \]
Compilation Algorithm

1) bring necessary variables into scope
2) specialize invariants to loops
3) ??? (find template)
4) profit (synthesize)
Compilation Algorithm

1) bring necessary variables into scope
2) specialize invariants to loops
3) ??? (find template)
4) profit (synthesize)
Extending Loops

foreach (w, m) in (weeks, months){
    w = 4 * m
}

for (w: int) in (weeks) {
    w = w * 1.05;
}
Extending Loops

```plaintext
foreach (w, m) in (weeks, months) {
    w = 4 * m
}
```

```plaintext
for (w: int) in (weeks) {
    w = w * 1.05;
}
```

weeks ~ months
Extending Loops

foreach (w, m) in (weeks, months) {
  w = 4 * m
}

for (w: int) in (weeks) {
  w = w * 1.05;
}

weeks ~ months

need months
Extending Loops

\[
\text{foreach (w, m) in (weeks, months)\{} \\
\hspace{1em} w = 4 * m \\
\text{\}}
\]

\[
\text{for (w: int) in (weeks) \{} \\
\hspace{1em} w = w * 1.05; \\
\text{\}}
\]

weeks \sim months

need months
Extending Loops

```plaintext
foreach (w, m) in (weeks, months) {
    w = 4 * m
}
```

```plaintext
for (w: int, m: int) in (weeks, months) {
    w = w * 1.05;
}
```

weeks ~ months

need months
Compilation Algorithm

1) bring necessary variables into scope
2) specialize invariants to loops
3) ??? (find template)
4) profit (synthesize)
Compilation Algorithm

1) bring necessary variables into scope
2) specialize invariants to loops
3) ??? (find template)
4) profit (synthesize)
Simplifying Invariants

```csharp
foreach (w, m) in (weeks, months){
    w = 4 * m
}
```

```csharp
for (w: int, m: int) in (weeks, months){
    w = w * 1.05;
}
```
Simplifying Invariants

Key insight: foreach body $\Leftrightarrow$ for invariant

```plaintext
foreach (w, m) in (weeks, months){
    w = 4 * m
}

for (w: int, m: int) in (weeks, months){
    w = w * 1.05;
}
```
Simplifying Invariants

Key insight: foreach body <=> for invariant
Simplifying Invariants

Key insight: foreach body $\leftrightarrow$ for invariant
Compilation Algorithm

1) bring necessary variables into scope
2) specialize invariants to loops
3) ??? (find template)
4) profit (synthesize)
Compilation Algorithm

1) bring necessary variables into scope
2) specialize invariants to loops
3) ??? (find template)
4) profit (synthesize)
**Budgeting Example**

### Invariants

\[ \forall i. 4 \times \text{weeks}[i] = \text{months}[i] \]
\[ \forall i. 12 \times \text{months}[i] = \text{years}[i] \]

\[ \forall i. \text{weeks}[i] < 0 \implies \text{font}[i] = \text{red} \]
\[ \forall i. \text{weeks}[i] \geq 0 \implies \text{font}[i] = \text{black} \]

\[ \forall i. i \mod 2 = 0 \implies \text{bkg}[i] = \text{even} \]
\[ \forall i. i \mod 2 = 1 \implies \text{bkg}[i] = \text{odd} \]

<table>
<thead>
<tr>
<th>Weekly</th>
<th>Monthly</th>
<th>Yearly</th>
</tr>
</thead>
<tbody>
<tr>
<td>-400</td>
<td>-1600</td>
<td>-19200</td>
</tr>
<tr>
<td>25</td>
<td>100</td>
<td>1200</td>
</tr>
<tr>
<td>750</td>
<td>3000</td>
<td>36000</td>
</tr>
<tr>
<td>-45</td>
<td>-180</td>
<td>-2160</td>
</tr>
</tbody>
</table>

**Color of Odd Rows**

**Color of Even Rows**
Budgeting Example

<table>
<thead>
<tr>
<th>Weekly</th>
<th>Monthly</th>
<th>Yearly</th>
</tr>
</thead>
<tbody>
<tr>
<td>-400</td>
<td>-1600</td>
<td>-19200</td>
</tr>
<tr>
<td>25</td>
<td>100</td>
<td>1200</td>
</tr>
<tr>
<td>750</td>
<td>3000</td>
<td>36000</td>
</tr>
<tr>
<td>-45</td>
<td>-180</td>
<td>-2160</td>
</tr>
</tbody>
</table>

Color of Odd Rows

Color of Even Rows

Invariants

\[ \forall i. 4 \times \text{weeks}[i] = \text{months}[i] \]
\[ \forall i. 12 \times \text{months}[i] = \text{years}[i] \]

\[ \forall i. \text{weeks}[i] < 0 \implies \text{font}[i] = \text{red} \]
\[ \forall i. \text{weeks}[i] \geq 0 \implies \text{font}[i] = \text{black} \]

\[ \forall i. i \mod 2 = 0 \implies \text{bkg}[i] = \text{even} \]
\[ \forall i. i \mod 2 = 1 \implies \text{bkg}[i] = \text{odd} \]
Budgeting Example

Need to synthesize loops, conditionals

<table>
<thead>
<tr>
<th>Weekly</th>
<th>Monthly</th>
<th>Yearly</th>
</tr>
</thead>
<tbody>
<tr>
<td>-400</td>
<td>-2500</td>
<td>-1500</td>
</tr>
<tr>
<td>250</td>
<td>1250</td>
<td>750</td>
</tr>
<tr>
<td>750</td>
<td>3750</td>
<td>2250</td>
</tr>
</tbody>
</table>

Color of Odd Rows

\( \forall i. i \text{ mod } 2 = 0 \implies bkg[i] = \text{even} \)

\( \forall i. i \text{ mod } 2 = 1 \implies bkg[i] = \text{odd} \)

Color of Even Rows
Budgeting Example

**Invariants**

- \( \forall i. i \mod 2 = 0 \implies bkg[i] = \text{evenColor} \)
- \( \forall i. i \mod 2 = 1 \implies bkg[i] = \text{oddColor} \)

**Color of Odd Rows**

**Color of Even Rows**

Need to synthesize **loops**, conditionals
foreach (color) in (bkgColors) {
    color = bkg
}
bkg = blue;
foreach (color) in (bkgColors) {
    color = bkg
}

bkg = blue;

bkg ~ bkgColors

bkg edited
foreach (color) in (bkgColors) {
    color = bkg
}

bkg = blue;

bkg ~ bkgColors

bkg edited

for (color: int) in (bkgColors){
    color = ??(bkg, blue);
}
Loops

```java
foreach (color) in (bkgColors) {
    color = bkg;
}
```

That's the skeleton...

What about the invariant?
foreach (color) in (bkgColors) {
    color = bkg
}

bkg = blue;

for (color: int) in (bkgColors) {
    color = ???(bkg, blue);
}
foreach (color) in (bkgColors) {
    color = bkg
}

bkg = blue;

for (color: int) in (bkgColors) {
    color = ??(bkg, blue);
}

Use foreach spec for body
foreach (color) in (bkgColors) {
    color = bkg
}

bkg = blue;
foreach (color) in (bkgColors) {
    color = bkg
}

bkg = blue;

for (color: int) in (bkgColors)
    color = blue;

Budgeting Example

Invariants

- $i \cdot 4 \cdot \alpha \cdot \text{weeks} [i] = \text{months} [i]$
- $i \cdot 12 \cdot \alpha \cdot \text{months} [i] = \text{years} [i]$
- $i \cdot \text{weeks} [i] < 0 \rightarrow \text{font} [i] = \text{red}$
- $i \cdot \text{weeks} [i] > 0 \rightarrow \text{font} [i] = \text{black}$
- $i \cdot \text{mod} 2 = 0 \rightarrow \text{bkg} [i] = \text{evenColor}$
- $i \cdot \text{mod} 2 = 1 \rightarrow \text{bkg} [i] = \text{oddColor}$

Need to synthesize loops, conditionals
Budgeting Example

Need to synthesize loops, **conditionals**

<table>
<thead>
<tr>
<th>Weekly</th>
<th>Monthly</th>
<th>Yearly</th>
</tr>
</thead>
<tbody>
<tr>
<td>-40</td>
<td>-25</td>
<td>750</td>
</tr>
<tr>
<td>-45</td>
<td>-180</td>
<td>-2160</td>
</tr>
</tbody>
</table>

**Color of Odd Rows**

\[ \forall i. i \mod 2 = 0 \implies bkg[i] = \text{even} \]

\[ \forall i. i \mod 2 = 1 \implies bkg[i] = \text{odd} \]
Synthesizing Conditionals

\[(\text{idx} \mod 2 = 0) \implies \text{color} = \text{even}\]

\[\text{even} = \text{red};\]
Synthesizing Conditionals

\[(idx \% 2 = 0) \rightarrow \text{color} = \text{even}\]

even = red;

Implications as conditionals
Synthesizing Conditionals

(idx % 2 == 0) \implies color = \text{even}

even = \text{red};

Implications as conditionals

if (idx % 2 == 0) {
    color = ??(even, red, idx);
}
Synthesizing Conditionals

\( (\text{idx} \mod 2 = 0) \implies \text{color} = \text{even} \)

even = red;

\[
\begin{align*}
\text{if} \ (\text{idx} \mod 2 == 0) \ {\{} \\
& \quad \text{color} = ??(\text{even}, \text{red}, \text{idx}); \\
\{ \end{align*}
\]
Synthesizing Conditionals

$(\text{idx} \mod 2 = 0) \implies \text{color} = \text{even}$

\[
\text{even} = \text{red};
\]

```java
if (idx % 2 == 0) {
    color = even;
}
```
Case studies*

Game of Life (1d, 2d)

Budgeting calculator

Split expenses spreadsheet

*
Game of Life

Invariants

\[ \forall i. \text{model}[i] \implies \text{view}[i] = \text{black} \]
\[ \forall i. \neg \text{model}[i] \implies \text{view}[i] = \text{white} \]

running \iff start\_enabled

Code

- State transition
- Interactive display
- Start/Stop toggle

* By Kieff - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=101736
**Budgeting**

### Weekly, Monthly, Yearly

<table>
<thead>
<tr>
<th></th>
<th>Weekly</th>
<th>Monthly</th>
<th>Yearly</th>
</tr>
</thead>
<tbody>
<tr>
<td>-400</td>
<td>-1600</td>
<td>-19200</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>100</td>
<td>1200</td>
<td></td>
</tr>
<tr>
<td>750</td>
<td>3000</td>
<td>36000</td>
<td></td>
</tr>
<tr>
<td>-45</td>
<td>-180</td>
<td>-2160</td>
<td></td>
</tr>
</tbody>
</table>

**Color of Odd Rows**

**Color of Even Rows**

---

**Invariants**

\[
\forall i.4 \ast \text{weeks}[i] = \text{months}[i]
\]

\[
\forall i.12 \ast \text{months}[i] = \text{years}[i]
\]

\[
\forall i.\text{weeks}[i] < 0 \implies \text{font}[i] = \text{red}
\]

\[
\forall i.\text{weeks}[i] \geq 0 \implies \text{font}[i] = \text{black}
\]

\[
\forall i.\text{mod} 2 = 0 \implies \text{bkg}[i] = \text{even}
\]

\[
\forall i.\text{mod} 2 = 1 \implies \text{bkg}[i] = \text{odd}
\]

**Code**

- Interactive cells
- Inflation
- Backgrounds
### Split expenses

<table>
<thead>
<tr>
<th>Weekly</th>
<th>Monthly</th>
<th>Yearly</th>
</tr>
</thead>
<tbody>
<tr>
<td>-400</td>
<td>-1600</td>
<td>-19200</td>
</tr>
<tr>
<td>25</td>
<td>100</td>
<td>1200</td>
</tr>
<tr>
<td>750</td>
<td>3000</td>
<td>36000</td>
</tr>
<tr>
<td>-45</td>
<td>-180</td>
<td>-2160</td>
</tr>
</tbody>
</table>

#### Invariants

- \( \forall i. \text{mid}[i] = \text{me}[i] + \text{you}[i] \)
- \( \forall i. \text{debt}[i] = \text{mid}[i] - \text{you}[i] \)
- \( \forall i. \text{debt}[i] < 0 \implies \text{color}[i] = \text{black} \)
- \( \forall i. \text{debt}[i] \geq 0 \implies \text{color}[i] = \text{red} \)

#### Code

- Interactive cells
- Backgrounds
Outline

- Data invariant calculus
- Extensions
- Invariants for Security
- GUI invariants
Formalisms
Formalisms

\[ \begin{align*}
\kappa & \in \mathcal{P(Inv)} \\
\downarrow_i & \in \mathcal{P(Inv)} \rightarrow \text{Expr} \\
\downarrow_s & \in (\mathcal{P(Inv)}, \text{Stmt}) \rightarrow \text{Stmt} \\
\text{body} & \in [\text{Vars}] \rightarrow \text{Vars} \rightarrow [\text{Vars}] \rightarrow \text{Inv} \rightarrow \text{Inv} \\
\text{iters} & \in [\text{Vars}] \rightarrow \text{Vars} \rightarrow [\text{Vars}] \rightarrow (\text{Expr, blk, blk, blk})
\end{align*} \]

(a) Declarations for functions used in translation from SPYDER to IMP-\textsc{Verif}.

\[
\begin{align*}
\text{Trans-Cond} & \quad \frac{\kappa \downarrow_i i \quad \kappa, \text{blk}_t \downarrow_s \text{blk}_t \quad \kappa, \text{blk}_f \downarrow_s \text{blk}_f}{\kappa, \text{Cond e blk}_t \text{ blk}_f \downarrow_s \text{Enf i ; Cond e blk}_t \text{ blk}_f ;}
\end{align*}
\]

\[
\begin{align*}
\kappa \downarrow_i i \quad \kappa' \downarrow_i i' & \quad \kappa' = \text{map(body v}_i \text{ idx arr}_i) \kappa \\
(e, \text{blk}_{let}, \text{blk}_{in}, \text{blk}_{out}) & = \text{iters v}_i \text{ idx arr}_i \quad \kappa', \text{blk} \downarrow_s \text{blk}
\end{align*}
\]

(b) Semantic definitions for translating conditional and loop statements in SPYDER to statements in IMP-\textsc{Verif}.

Figure 7. Translation Semantics between SPYDER and IMP-\textsc{Verif}.
Formalisms

Interesting Translation Rule

\[ \kappa \in \mathcal{P}(\text{Inv}) \]

**Trans-Cond**

\[
\kappa, \text{Cond } e \text{ blk}_t \text{ blk}_f \downarrow_s \text{Enf } i; \text{ Cond } e \text{ blk}_t \text{ blk}_f ;
\]

\[
\kappa \downarrow_i i \quad \kappa' \downarrow_i i'
\]

\[
\kappa' = \text{map} (\text{body } v_i \text{ idx } arr_i) \kappa
\]

\[
(e, \text{blk}_\text{let}, \text{blk}_\text{in}, \text{blk}_\text{out}) = \text{iters } v_i \text{ idx } arr_i
\]

\[
\kappa', \text{blk} \downarrow_s \text{blk}
\]

**Trans-For**

\[
\kappa, \text{For } (v_i) \text{ idx } (arr_i) \text{ blk} \downarrow_s \text{Enf } i; \text{ blk}_\text{let}; \text{While } e \text{ i } (\text{blk}_\text{in}; \text{ Assume } i'; \text{ blk}; \text{ Enf } i'; \text{ blk}_\text{out})
\]

(b) Semantic definitions for translating conditional and loop statements in SPYDER to statements in IMP-VERIF.

Figure 7. Translation Semantics between SPYDER and IMP-VERIF.
Formalisms

What we need

- Formal problem definition
- Full translation semantics
- Proof of correctness
Formal Correctness

Problem:

Correctness:
Formal Correctness

Problem: Given a program with arrays and invariants...

Correctness:
Formal Correctness

Problem: Given a program with arrays and invariants…

Correctness: We find a program with augmented semantics…
Formal Correctness

Problem: Given a program with arrays and invariants...

Correctness: We find a program with augmented semantics...

Thank you, Reviewer A!!!
Stronger Invariants

\[ \forall i, j. \; i < j \implies sorted[i] < sorted[j] \]

Consider Sortedness…
Stronger Invariants

\[ \forall i, j. \ i < j \implies sorted[i] < sorted[j] \]

Inv ::= ForEach \((v_1 \ldots v_n)\) [v] \((v_1 \ldots v_n)\) Inv

| BaseExpr Expr
**Stronger Invariants**

\[ \forall i, j. \ i < j \implies \text{sorted}[i] < \text{sorted}[j] \]

\[
\text{Inv} ::= \text{Foreach} (v_1 \ldots v_n) \ [v] (v_1 \ldots v_n) \ Inv \\
| \text{BaseExpr} \ Expr
\]

Limited to "maplike" properties
Stronger Invariants

\[ \forall i, j. \ i < j \implies \text{sorted}[i] < \text{sorted}[j] \]

Inv ::= **Foreach** \( (v_1 \ldots v_n) \ [v] \ (v_1 \ldots v_n) \) Inv
       | **BaseExpr** Expr

Cannot express :(
Stronger Invariants

Key insight: change of perspective!!

Cannot express :(
Stronger Invariants

\[ \forall i, j. \ i < j \implies \text{sorted}[i] < \text{sorted}[j] \]
\[\forall i, j. \quad i < j \implies \text{sorted}[i] < \text{sorted}[i + 1]\]

\[\forall i. \quad \text{sorted}[i] < \text{sorted}[i + 1]\]
∀i, j.  \( i < j \)  \implies\  \text{sorted}[i] < \text{sorted}[i + 1]

∀i.  \text{sorted}[i] < \text{sorted}[i + 1]
\( \forall i, j. \ P(i, j) \)

\( \forall i. \ P(i, i + 1) \)
Thanks Nadia!!

\[ \forall i. P(i, i + 1) \]

\[ \forall i, j. P(i, j) \]
adjacent (lo, hi) in (sorted) {
  lo < hi
}
Synthesizing Insert

adjacent (lo, hi) in (sorted) {
    lo < hi
}

sorted = sorted + newElem
Synthesizing Insert

adjacent (lo, hi) in (sorted) {
  lo < hi
}

sorted = sorted + newElem

sorted = foldl adder ([], false) sorted
adder (es, d) next = ...
Synthesizing Insert

\[
\text{adjacent (lo, hi) in (sorted) { } lo < hi }
\]

\[
\text{sorted = sorted + newElem}
\]

But in imperative language

\[
\text{sorted = foldl adder ([], false) sorted}
\text{adder (es, d) next = ...}
\]
Implications

- Not clear — need to adjust at least
  - 1) loop translation
  - 2) synthesis
- Formal correctness would help!!
Outline

- Data invariant calculus
- Extensions
- Invariants for Security
- GUI invariants
Outline

- Data invariant calculus
- Extensions
- Invariants for Security
- GUI invariants
Security is really hard
';--have i been pwned?

Check if you have an account that has been compromised in a data breach

jtsarracino@gmail.com

Adobe: In October 2013, 153 million Adobe accounts were breached with each containing an internal ID, username, email, encrypted password and a password hint in plain text. The password cryptography was poorly done and many were quickly resolved back to plain text. The unencrypted hints also disclosed much about the passwords adding further to the risk that hundreds of millions of Adobe customers already faced.

Compromised data: Email addresses, Password hints, Passwords, Usernames

Kickstarter: In February 2014, the crowdfunding platform Kickstarter announced they'd suffered a data breach. The breach contained almost 5.2 million unique email addresses, usernames and salted SHA1 hashes of passwords.

Compromised data: Email addresses, Passwords

LinkedIn: In May 2016, LinkedIn had 164 million email addresses and passwords exposed. Originally hacked in 2012, the data remained out of sight until being offered for sale on a dark market site 4 years later. The passwords in the breach were stored as SHA1 hashes without salt, the vast majority of which were quickly cracked in the days following the release of the data.

Compromised data: Email addresses, Passwords

Ticketfly: In May 2016, the website for the ticket distribution service Ticketfly was defaced by an attacker and was subsequently taken offline. The attacker allegedly requested a ransom to share details of the vulnerability with Ticketfly but did not receive a reply and subsequently posted the breached data online to a publicly accessible location. The data included over 26 million unique email addresses along with names, physical addresses and phone numbers. Whilst there were no
Facebook Network is Breached, Putting 50 Million Users’ Data at Risk

Facebook Faces Potential $1.63 Billion Fine in Europe Over Data Breach

Privacy watchdog looks into whether social network violated European’s Union new privacy law
Manual Security

Invariants

```
for i in people.data.users:
    response = client.api.statuses.user_timeline.get(screen_name=i.screen_name)
    print 'Got', len(response.data), 'tweets from', i.screen_name
    if len(response.data) != 0:
        ldate = response.data[0]['created_at']
        ldate2 = datetime.strptime(ldate, '%a %b %d %H:%M:%S +0000 %Y')
        today = datetime.now()
        howlong = (today-ldate2).days
        if howlong < daywindow:
            print i.screen_name, 'has tweeted in the past', daywindow,
            totaltweets += len(response.data)
            for j in response.data:
                if j.entities.Urls:
                    for k in j.entities.urls:
                        newurl = k['expanded_url']
                        urlset.add((newurl, j.user.screen_name))
                else:
                    print i.screen_name, 'has not tweeted in the past', daywindow
```

Invariant:

“All urls are in the urlset”
Manual Security

Invariants

```
for i in people.data.users:
    response = client.apistatuses.user_timeline.get(screen_name=i.screen_name)
    print 'Got', len(response.data), 'tweets from', i.screen_name
    if len(response.data) != 0:
        ldate = response.data[0]['created_at']
        ldate2 = datetime.strptime(ldate, '%a %b %d %H:%M:%S +0000 %Y')
        today = datetime.now()
        howlong = (today-lbirthdate).days
        if howlong < daywindow:
            print i.screen_name, 'has tweeted in the past', daywindow,
            totaltweets += len(response.data)
            for j in response.data:
                if j.entities.urls:
                    for k in j.entities.urls:
                        newurl = k['expanded_url']
                        urlset.add((newurl, j.user.screen_name))
                else:
                    print i.screen_name, 'has not tweeted in the past', daywindow,
```

Invariant:

“All urls are in the urlset”

“All connections use the urlset”
Manual Security

Invariant:

“All urls are in the urlset”
“All connections use the urlset”
Security Synthesis

Policy

Algorithm

Code
Security Synthesis

Catch policy errors at compile time
Security Policy Spectrum

Easy  Hard  Insane
Security Policy Spectrum

“all urls are safe”

Permissions

Easy  Hard  Insane
Security Policy Spectrum

“the secret is not leaked”

Information Flow

Easy  Hard  Insane
Security Policy Spectrum

“no buffers overflow”

Runtime Systems

Easy  Hard  Insane
Security Policy Spectrum

"all urls are safe"

Permissions

Easy  Hard  Insane
Permission Invariants

foreach (u1, u2) in (Users, Users) {
    u1 can view info(u2) <=> u1 in Doctors && u2 in patients(u1)
}

render(u: User) {
    for (record) in (records) {
        display record
    }
}
Permission Invariants

foreach (u1, u2) in (Users, Users) {
    u1 can view info(u2) \iff\ u1 in Doctors && u2 in patients(u1)
}

render(u: User) {
    for (record) in (records) {
        if (u in Doctors && patient(record) in patients(u)) {
            display record
        }
    }
}
Outline

- Data invariant calculus
- Extensions
- Invariants for Security
- GUI invariants
Outline

- Data invariant calculus
- Extensions
- Invariants for Security
- GUI invariants
Model-View Synchronization

Model

View

render

update
Synchronization as Invariants
Synchronization as Invariants
GUI Invariants

“all text has sufficient contrast”

“append post_content to the end of tweet_body”
GUI Invariants

“all text has sufficient contrast”

“append post_content to the end of tweet_body”

Inspired by Cassius, PLDI ’18: Verifying that Web Pages have Accessible Layout
GUI Invariants

```javascript
foreach txtBlock in texts(document) {
  foreach block in blocks(document) {
    block isBehind txtBlock ==> goodContrast(txtBlock, block)
  }
}
```

```javascript
tweet_body.appendChild(post_content);
```
GUI Invariants

```javascript
tweet_body.appendChild(post_content);

```
GUI Invariants

```javascript
// Accessibility-agnostic programming!!

// Code snippet

t1 = text(txtBlock[0:mid], "blue");
t2 = text(txtBlock[mid:end], "black");
t3 = text(txtBlock[end:], "white");
spn.appendChild(t1)
    .appendChild(t2)
    .appendChild(t3);
tweet_body.appendChild(spn);
```

```javascript
// Code snippet

// Accessibility-agnostic programming!!

t1 = text(txtBlock[0:mid], "blue");
t2 = text(txtBlock[mid:end], "black");
t3 = text(txtBlock[end:], "white");
spn.appendChild(t1)
    .appendChild(t2)
    .appendChild(t3);
tweet_body.appendChild(spn);
```
Outline

- Data invariant calculus
- Extensions
- Invariants for Security
- GUI invariants
My thesis proposal

Compiling and generating invariants is useful, novel, and broadly applicable.

I propose a general framework for maintaining invariants