Path Projection
For User-Centered Static Analysis Tools

Khoo Yit Phang, Jeff Foster, Michael Hicks, Vibha Sazawal
University of Maryland
PASTE 2008, November 10
Success in Static Analysis

- Coverity, Fortify, Grammatech, Klocwork, many others are selling static analysis tools
- Microsoft, Mozilla, and others are integrating static analysis into development
- Very active static analysis research community
Research has focused on static analysis algorithms.

But, programmers use tools, not algorithms.

Static analysis tools are only useful if programmers can understand the results.

Our goal: develop ways to make static analysis tools more user-centered.
Path Projection

- A new UI toolkit for visualizing program paths
  - call stacks and control-flow paths

Paths are very common in static analysis tool output
- Helping users understand paths will help many static analysis tools
- We have applied Path Projection to Locksmith and BLAST

Experimental evaluation
- Task: triaging Locksmith error reports
- Result: 18% improvement in completion time, similar accuracy
Case Study: Locksmith

Polyvios Pratikakis et al. (PLDI 2006)

- Static data race detector for C
  - Data race: Two or more threads access a shared variable at the same time

- Locksmith reports call stacks to possibly-racing dereferences
  - To triage, user must decide whether multiple paths are simultaneously realizable
Locksmith in Standard Viewer

Standard Viewer designed to mimic typical editors/IDEs
Locksmith error report with hyperlinks
Warning: Possible data race of prev (Misc.c:<global>:15)
at:
1. <in main.c>
   main():42 -> pthread_create()
   <in Signal.c>
   signal_waiter():36
   sigalarm_handler():61
   <in Misc.c>
   updateProgressBar():193 -> dereference
   locks: -

2. <in main.c>
   main():117
   <in Aget.c>
   resume_get():171 -> pthread_create()
   <in Download.c>
   http_get():121
   <in Misc.c>
   updateProgressBar():193 -> dereference
   locks: -
Locksmith Error Report

Path Report

Warning: Possible data race of
prev (Misc.c:<global>:15)
at:

1. <in main.c>
   main():42  -> pthread_create()
   <in Signal.c>
   signal_waiter():36
   sigalarm_handler():61
   <in Misc.c>
   updateProgressBar():193  -> dereference

   locks: -

2. <in main.c>
   main():117
   <in Aget.c>
   resume_get():171  -> pthread_create()
   <in Download.c>
   http_get():121
   <in Misc.c>
   updateProgressBar():193  -> dereference

   locks: -
Locksmith Error Report

Path Report

Warning: Possible data race of prev (Misc.c:<global>:15)

at:

1. <in main.c>
   main():42 -> pthread_create()
   <in Signal.c>
   signal_waiter():36
   sigalm_handler():61
   <in Misc.c>
   updateProgressBar():193 -> dereference

   locks: -

2. <in main.c>
   main():117
   <in Aget.c>
   resume_get():171 -> pthread_create()
   <in Download.c>
   http_get():121
   <in Misc.c>
   updateProgressBar():193 -> dereference

   locks: -

Shared variable
Locksmith Error Report

Path Report

Warning: Possible data race of prev (Misc.c:<global>:15)
at:

1. <in main.c>
   main():42 -> pthread_create()
   <in Signal.c>
   signal_waiter():36
   sigalrm_handler():61
   <in Misc.c>
   updateProgressBar():193 -> dereference

   locks: -

2. <in main.c>
   main():117
   <in Aget.c>
   resume_get():171 -> pthread_create()
   <in Download.c>
   http_get():121
   <in Misc.c>
   updateProgressBar():193 -> dereference

   locks: -

Shared variable

Call stacks leading to race
Locksmith Error Report

Path Report

Warning: Possible data race of prev (Misc.c:<global>:15)
at:

1. <in main.c>
   main():42    -> pthread_create()
   <in Signal.c>
   signal_waiter():36
   sigalarm_handler():61
   <in Misc.c>
   updateProgressBar():193 -> dereference

   locks: -

2. <in main.c>
   main():117
   <in Aget.c>
   resume_get():171    -> pthread_create()
   <in Download.c>
   http_get():121
   <in Misc.c>
   updateProgressBar():193 -> dereference

   locks: -

Thread creation
Locksmith Error Report

Path Report

Warning: Possible data race of prev (Misc.c:<global>:15)
at:

1. <in main.c>
   main():42  ->  pthread_create()
   <in Signal.c>
   signal_waiter():36
   sigalarm_handler():61
   <in Misc.c>
   updateProgressBar():193  ->  dereference

   locks: -

2. <in main.c>
   main():117
   <in Aget.c>
   resume_get():171  ->  pthread_create()
   <in Download.c>
   http_get():121
   <in Misc.c>
   updateProgressBar():193  ->  dereference

   locks: -

Thread creation
Dereference
Locksmith Error Report

Path Report

Warning: Possible data race of prev (Misc.c:<global>:15)

at:

1. <in main.c>
   main():42       -> pthread_create()
   <in Signal.c>
   signal_waiter():36
   sigalarm_handler():61
   <in Misc.c>
   updateProgressBar():193 -> dereference
   locks: -

2. <in main.c>
   main():117
   <in Aget.c>
   resume_get():171       -> pthread_create()
   <in Download.c>
   http_get():121
   <in Misc.c>
   updateProgressBar():193 -> dereference
   locks: -

Thread creation
Dereference
w/no locks held
# Locksmith Error Report

## Path Report

Warning: Possible data race of prev (Misc.c:<global>:15) at:

1. `<in main.c>`
   - `main():42 -> pthread_create()`
   - `<in Signal.c>`
     - `signal_waiter():36`  
     - `sigalrm_handler():61`  
   - `<in Misc.c>`
     - `updateProgressBar():193 -> dereference`

   locks: -

2. `<in main.c>`
   - `main():117`  
   - `<in Aget.c>`
     - `resume_get():171 -> pthread_create()`
     - `<in Download.c>`
       - `http_get():121`  
   - `<in Misc.c>`
     - `updateProgressBar():193 -> dereference`

   locks: -

- **Thread creation**
- **Dereference**
- **w/no locks held**
### Locksmith Error Report

#### Path Report

**Warning:** Possible data race of prev (Misc.c:<global>:15) at:

1. `<in main.c>`  
   `main():42`  
   `<in Signal.c>`  
   `signal_waiter():36`  
   `sigalrm_handler():61`  
   `<in Misc.c>`  
   `updateProgressBar():193` -> **dereference**

   locks: -

2. `<in main.c>`  
   `main():117`  
   `<in Aget.c>`  
   `resume_get():171`  
   `<in Download.c>`  
   `http_get():121`  
   `<in Misc.c>`  
   `updateProgressBar():193` -> **dereference**

   locks: -
Triaging Locksmith

Triage: are these call stacks simultaneously realizable?
Begin by clicking
Begin by clicking
Thread creation realizable?
Thread creation realizable?

Yes, unconditionally created

```c
16  extern int errno;
17  int main(int argc, char **argv)
18  {
19      extern char *optarg;
20      extern int optind;
21      int c; int error = 0; int ret;
22      struct hist_data h;
23      int retlog;
24
25      /* Allocate heap for download request */
26      /* struct request stores all the information that might be */
27      /* of interest */
28      /*
29          req = (struct request *)calloc(1, sizeof(struct request));
30
31      /* Only some signals will be emitted */
32      sigemptyset(&signal_set);
33      sigaddset(&signal_set, SIGINT);
34      sigaddset(&signal_set, SIGALRM);
35
36      /* Block out all signals */
37      pthread_sigmask(1, &signal_set, NULL);
38
39      /* Create a thread for handling signals */
40      if ((ret = pthread_create(&thread, NULL, signal_waiter, NULL)) != 0) {
41          printf(stderr, "main: cannot create signal_waiter thread: %s, exited\n",
42          exit(-1);
```

Path Report

Warning: Possible data race of prev (Misc.c::<global>:15) at:

1. `<in main.c>
   main():42   -> pthread_create()
   `<in Signal.c>
   signal_waiter():36
   sigalrm_handler():61
   `<in Misc.c>
   updateProgressbar():193 -> dereference

   locks: -

2. `<in main.c>
   main():117
   `<in Aget.c>
   resume_get():171   -> pthread_create()
   `<in Download.c>
   http_get():121
   `<in Misc.c>
   updateProgressbar():193 -> dereference

   locks: -
Keep around context
Triaging Locksmith

Keep around context

Easier than remembering
Triaging Locksmith

Call site of

Call site of

Next call

Warning: Possible data race of prev (Misc.c:<global>:15) at:
> pthread_create()
  signal_waiter():36
  signalrm_handler():61
  updateProgressbar():193 -> dereference

locks: -

2. <in main.c>
  main():117
  resume_get():171 -> pthread_create()
  updateProgressbar():193 -> dereference
  http_get():121
  updateProgressbar():193 -> dereference
  locks: -
the dereference, finally!
/* Create a thread for handling signals */
if ((ret = pthread_create(&hthread, NULL, signal_waiter, NULL)) != 0)
    fprintf(stderr, "main: cannot create signal_waiter thread: %s, exit(-1);

int signal;

exit(0);

void * signal_waiter(void *arg)
{

void sigalarm_handler(void)
{

putchar(' ');
printf("[%d%% completed]\n", ndot);
prev = ndot;
Screen is very cluttered!
/* Create a thread for handling signals */
if ((ret = pthread_create(hthread, NULL, signal_waiter, NULL)) != 0) {
    fprintf(stderr, "main: cannot create signal_waiter thread: %s, exiting!
exit(-1);
}

void * signal_waiter(void *arg)
{
    int signal;
}

void sigalrm_handler(void)
{
}

putchar(' ');
printf("[%d%% completed]\n", ndot);
prev = ndot;
Which function is this?

Old context is hidden!

Where was this called?
A Thousand Cuts
A Thousand Cuts

Many little distractions from actual task

Seemingly straightforward task becomes complex!
Path Projection

Designed for tracing paths
Path Projection

- Designed for tracing paths
- Function call inlining:
  Inline function directly below call site
Path Projection

- Designed for tracing paths
- Function call inlining:
  Inline function directly below call site
- Path-derived code folding:
  Show only implicated lines and lexical control-blocks
Path Projection

- Designed for tracing paths

- Function call inlining:
  Inline function directly below call site

- Path-derived code folding:
  Show only implicated lines and lexical control-blocks

- Show as much code as possible on one screen
int main(int argc, char **argv)
{
    if ((ret = pthread_create(&hthread, NULL, signal_waiter, NULL)) == -1)
    {
        perror("pthread_create error");
        exit(EXIT_FAILURE);
    }

    signal_waiter(void * arg);
    while(1)
    {
        if (signal == SIGINT)
        {
            sigalarm_handler();
        }
        else if (signal == SIGALRM)
        {
            sigalarm_handler();
        }
    }

    signal_handler();
    updateProgress(bwritten, req->length);
}

void sigalarm_handler(void)
{
    updateProgress(bwritten, req->length);
}

void updateProgress(float cur, float tot)
{
    prev = ndot;
    ...
}
Path Projection

Show paths side by side
Path Projection

Show paths side by side
Path Projection

Path 2 of 2

```c
int main(int argc, char **argv) {
    if ((ret = pthread_create(&hthread, NULL, signal_wait)) != -1) {
        resume_get(&h);
    }
    if ((retlog = read_log(&h)) != -1) {
        resume_get(&h);
    }
}
```

```c
void resume_get(struct hist_data *h) {
    for (i = 0; i < nthreads; i++) {
        pthread_create(&wthread[i].tid, NULL, http_get);
    }
}
```

```c
void *http_get(void *arg) {
    pthread_mutex_lock(&bwritten_mutex);
    pthread_mutex_unlock(&bwritten_mutex);
    while (td->offset < offsetset) {
        pthread_mutex_lock(&bwritten_mutex);
        pthread_mutex_unlock(&bwritten_mutex);
        updateProgressBar(bwritten, td->clength);
    }
}
```

```c
void updateProgressBar(float cur, float tot) {
    prev = ndot;
}
```

Multi-query

- pthread_create
- pthread_join
- pthread_mutex_lock
- pthread_mutex_unlock

```
main():42 -> pthread_create()
Signal.c:36
Sigalm_handler():61
Misc.c:193 -> dereference
```

```
2. <in main.c>
main():117
Aget.c:94
resume_get():171 -> pthread_create()
Download.c:121
http_get():121
Misc.c:193 -> dereference
```

locks: -
Path Projection

Multiple searches (despite folds)
Path Projection

Continuing example...

Path Report

Warning: Possible data race of prev (Misc.c:<global>:15) at:
1. <in main.c>
   main():42 -> pthread_create()
   <in Signal.c>
   signal_waiter():36
   sigalm_handler():61
   <in Misc.c>
   updateProgressBar():193 -> dereference
   locks: -

2. <in main.c>
   main():117
   <in Aget.c>
   resume_get():171 -> pthread_create()
   <in Download.c>
   http_get():121
   <in Misc.c>
   updateProgressBar():193 -> dereference
   locks: -
Path Projection

from 1st call stack
Path Projection

from 2nd call stack
Path Projection

from 2nd call stack
dereference
Path Projection

**Path Report**

- Possible data race of `prev (Misc.c:<global>:15)` at:
  1. `<in main.c>`
     - `main()`: Dereference
     - `pthread_create()`
   - `signal_waiter()`: Dereference
     - `sigalm_handler()`
   - `updateProgressbar()`

- `updateProgressbar()` calls `pthread_mutex_lock()` and `pthread_mutex_unlock()`.

**Code Snippet**

```c
19 int main(int argc, char **argv)
20 {
21   if (ret = pthread_create(&hthread, NULL)) {
22     exit(1);
23   }
24   if ((retlog = read_log(&h)) != -1)
25     resume_get(&h);
26 }
27
37 void * http_get(void * arg)
38 {
39   pthread_mutex_lock(&wbwritten_mutex);
40   pthread_mutex_unlock(&wbwritten_mutex);
41   while (td->offset < foffset) {
42     pthread_mutex_lock(&wbwritten_mutex);
43     pthread_mutex_unlock(&wbwritten_mutex);
44     updateProgressBar(wbwritten, td->length);
45   }
46   pthread_mutex_lock(&http_get_ok_mutex);
47   pthread_mutex_unlock(&http_get_ok_mutex);
48   return (void *) 0;
49 }
50
133 void resume_get(struct hist_data * h)
134 {
135   for (i = 0; i < nthreads; i++) {
136     pthread_mutex_lock(&wthread[i].tid);
137     pthread_mutex_unlock(&wthread[i].tid);
138   }
139   for (i = 0; i < nthreads; i++) {
140     pthread_join(wthread[i].tid, NULL);
141     if (http_get_ok == nthreads)
142       break;
143   }
144 }
145
172 for (i = 0; i < nthreads; i++)
173   pthread_join(wthread[i].tid, NULL);
174 if (http_get_ok == nthreads)
175     exit(1);
176 }
177 void updateProgressBar(float cur, float tot)
178 {
179   prev = ndot;
180 }
181
182 if (td->offset == td->foffset) {
183   pthread_mutex_lock(&wbwritten_mutex);
184   pthread_mutex_unlock(&wbwritten_mutex);
185 } else {
186   pthread_mutex_lock(&wbwritten_mutex);
187   pthread_mutex_unlock(&wbwritten_mutex);
188 }
```
Path Projection

not a single click or scroll!
Path Projection

not a single click or scroll!

no need to look here too!
What's foffset?
Path Projection

Path 2 of 2

Path Visualizer

Multi-query

foffset

Path Report

Warning: Possible data race of prev (Misc.c:<global>:15)
at:
1. <in main.c>
   main():42 -> pthread_create()
   <in Signal.c>
   signal_waiter():36
   sigalm_handler():61
   <in Misc.c>
   updateProgressBar():193 -> dereference
   locks: -

2. <in main.c>
   main():117
   <in Aget.c>
   resume_get():171 -> pthread_create()
   <in Download.c>
   http_get():121
   <in Misc.c>
   updateProgressBar():193 -> dereference
   locks: -
Path Projection

What's in read_log?

Path Report

Warning: Possible data race of prev (Misc.c: <global>:15)
at:
1. <in main.c>
   main():42 -> pthread_create()
   <in Signal.c>
   signal_waiter():36
   sigalm_handler():61
   <in Misc.c>
   updateProgressBar():193 -> dereference
   locks: -

2. <in main.c>
   main():117
   <in Aget.c>
   resume_get():171 -> pthread_create()
   <in Download.c>
   http_get():121
   <in Misc.c>
   updateProgressBar():193 -> dereference
   locks: -
We discovered that static analysis is...
In our pilot studies, non-expert users had great trouble triaging Locksmith error reports:

- ad hoc, inconsistent procedure
- neglected some causes of false positives
- sidetracked by non-causes of false positives

Even with extensive tutorials!
Our solution: triaging checklist

Checklists are tool-/error-specific
- Different tools have different imprecision & error reports

Anecdotally, 41% faster at triaging using checklist
Locksmith Triaging Checklist

To triage Locksmith:
check if any pair of paths are simultaneously realizable
different cases: threads in loop, parent-child, child-child

For example:
Source of imprecision: Locksmith is path-insensitive
Possible false positive: child-child threads may be mutually exclusive

---

For threads leading to dereferences in Paths 1 and 2:
Are they parent-child (or child-parent), or child-child?

- Parent-child / Child-child

---

Child-child threads.
Are the children mutually exclusive (i.e., only one can be spawned by their common parent/ancestor)?
If no, there is likely a race. Are there reasons to show otherwise?
Explain: ????
Locksmith Triaging Checklist

To triage Locksmith:
check if any pair of paths are simultaneously realizable
different cases: threads in loop, parent-child, child-child

For example:
Source of imprecision: Locksmith is path-insensitive
Possible false positive: child-child threads may be mutually exclusive
Locksmith Triaging Checklist

To triage Locksmith:
check if any pair of paths are simultaneously realizable
different cases: threads in loop, parent-child, child-child

For example:
Source of imprecision: Locksmith is path-insensitive
Possible false positive: child-child threads may be mutually exclusive

For threads leading to dereferences in Paths 1 and 2:
Are they parent-child (or child-parent), or child-child?
- Parent-child / Child-child

Child-child threads.
Are the children mutually exclusive (i.e., only one can be spawned by their common parent/ancestor)?
- If no, there is likely a race. Are there reasons to show otherwise? ???
  Explain: ???
User Study

Which is better: Standard Viewer (SV) or Path Projection (PP)?
- Quantitatively: completion time
- Qualitatively: user ratings

Data race triaging task using Locksmith
User Study Issues

- Large variance between participants
  - Participants have different skill level
  - Are differences due to participant or UI?

- Within-subjects: each participant use both interfaces
  - Compare UI results for each participant
User Study Issues

- **Order and carryover effect**
  - Participants get better over time (learning)
  - Participants biased by initial UI or problem

- **Counter-balance: divide participants into two groups**
  - SV-PP: Standard Viewer, then Path Projection
  - PP-SV: Path Projection, then Standard Viewer
User Study: Locksmith Task

- 6 trials from Locksmith corpus (unfamiliar to users)
- One warning per trial
  - no need to manage warnings
- Only verify that paths are simultaneously realizable
  - No aliasing/imprecise lock state (future work)
User Study: Misc.

- 8 student participants
- 3 undergraduates, 5 graduates
- Prior experience in C, multithreading (not necessarily C)
- Self-rated 3-4 (1: no experience to 5: very experienced)
- 2 had experience in Locksmith and Eraser
Quantitative (Chart guide)

![Completion time (sec)]

- Session 1: Standard Viewer
- Session 2: Path Projection
Quantitative (Chart guide)
Quantitative (Chart guide)
Faster Completion Time

Completion time (sec)

- Standard Viewer
- Path Projection

Session 1

Session 2
Learning effect
all improved in Session 2*

*statistically significant (p<0.05)
Faster Completion Time

Learning effect
  all improved in Session 2*

SV-PP improved by 188s*
  (effect size d=1.276)

*statistically significant (p<0.05)
**Learning effect**
- all improved in Session 2*
- SV-PP improved by 188s*  
  (effect size d=1.276)
- PP-SV improved by 55s*  
  (effect size d=0.375)

*statistically significant (p<0.05)
Faster Completion Time

Learning effect
- all improved in Session 2*

SV-PP improved by 188s*
  (effect size d=1.276)

PP-SV improved by 55s*
  (effect size d=0.375)

Similar # mistakes
- 10 in PP (10.9%), 9 in SV (9.8%)

18% faster on average

*statistically significant (p<0.05)
Less Use of Error Report

Duration where pointer is over error report (e.g., using hyperlinks)

Duration in error report (sec)

- Standard Viewer
- Path Projection

Session 1
Session 2
Less Use of Error Report

- Duration where pointer is over error report (e.g., using hyperlinks)
- On average, only 20s with PP vs. 94s with SV*

*statistically significant (p<0.05)
Less Use of Error Report

- Duration where pointer is over error report (e.g., using hyperlinks)
- On average, only 20s with PP vs. 94s with SV*
- “Necessary for [SV], but just a convenience in [PP].”

*statistically significant (p<0.05)
Qualitative (Boxplot Guide)

We asked participants to rate on a 1-5 scale.
Qualitative (Boxplot Guide)

We asked participants to rate on 1-5 scale

Results summarized in boxplots
Prefer Path Projection

Overall impression

<table>
<thead>
<tr>
<th></th>
<th>Quick to learn</th>
<th>Confident of answer</th>
<th>Easy to verify race</th>
<th>Prefer Path Projection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Viewer</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Path Projection</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Strongly agree

Strongly disagree
Quick/confident/easy: not statistically significant
Prefer Path Projection

Quick/confident/easy: not statistically significant

Preference: all but one preferred PP to SV*

*statistically significant (p<0.05)
## Path Projection Features

### Usefulness of features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Very useful</th>
<th>Not useful</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error report</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Checklist</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function inlining</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code folding</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi-query</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Query reveals folded code</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Error report: 4
- Checklist: 3
- Function inlining: 5
- Code folding: 4
- Multi-query: 2
- Query reveals folded code: 3
Path Projection Features

**Usefulness of features**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Usefulness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error report</td>
<td>Very useful</td>
</tr>
<tr>
<td>Checklist</td>
<td>Not useful</td>
</tr>
<tr>
<td>Function inlining</td>
<td>Not useful</td>
</tr>
<tr>
<td>Code folding</td>
<td>Not useful</td>
</tr>
<tr>
<td>Multi-query</td>
<td>Not useful</td>
</tr>
<tr>
<td>Query reveals folded code</td>
<td>Very useful</td>
</tr>
</tbody>
</table>

*Generally favorable towards PP features*

*statistically significant (p<0.05)*
Path Projection Features

Usefulness of features

- Error report
- Checklist
- Function inlining
- Code folding
- Multi-query
- Query reveals folded code

Checklist: “saved me from having to memorize rules”
Path Projection Features

Checklist: “saved me from having to memorize rules”

Surprisingly, favored function inlining/code folding

code folding was “the best feature” or “my favorite feature”
Threats To Validity

- Experimental design limitations
  - small number of users and trials
  - not static analysis experts, unfamiliar programs
  - statistically significant despite limitations

- Standard Viewer not “real” editor
  - deliberate choice to avoid bias from prior experience

- The checklist might bias users
  - checklist designed for Locksmith, not SV or PP
  - both interfaces use the same checklist
Conclusion

Path Projection: a new UI toolkit for visualizing program paths

Can be used with any static analysis tools
  Takes an XML path report as input

Our study showed that it improves completion time (18%) with similar accuracy and users liked it