ANALYSIS FOR
TRUSTWORTHY SOFTWARE

DAVID VAN HORN

WITH SUPPORT FROM
NSF, DARPA, CRA, & Google
<table>
<thead>
<tr>
<th>Example Apps (1:2)</th>
<th>Invented in theater</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PLI</strong></td>
<td>Blue-force tracking</td>
</tr>
<tr>
<td><strong>DASH</strong></td>
<td>Messages</td>
</tr>
<tr>
<td><strong>RTC</strong></td>
<td>Chat rooms</td>
</tr>
<tr>
<td><strong>Collab</strong></td>
<td>Live Collaboration</td>
</tr>
<tr>
<td><strong>SAR</strong></td>
<td>Coordinated Search &amp; Rescue</td>
</tr>
<tr>
<td><strong>STREAM</strong></td>
<td>Sensor streaming</td>
</tr>
<tr>
<td><strong>SpeedTest</strong></td>
<td>Network performance</td>
</tr>
<tr>
<td><strong>Maps</strong></td>
<td>High res maps navigation, layers</td>
</tr>
<tr>
<td><strong>Mapdraw</strong></td>
<td>Operational graphics</td>
</tr>
<tr>
<td><strong>TRAQ</strong></td>
<td>GPS trace &amp; media</td>
</tr>
<tr>
<td><strong>Transheat</strong></td>
<td>Patrol heatmap</td>
</tr>
<tr>
<td><strong>Debrief</strong></td>
<td>Patrol review</td>
</tr>
<tr>
<td><strong>TIGR</strong></td>
<td>Events, Places, People, Reports</td>
</tr>
<tr>
<td><strong>Patrolview</strong></td>
<td>Collector &amp; Viewer</td>
</tr>
<tr>
<td><strong>WhoDat</strong></td>
<td>Local population</td>
</tr>
<tr>
<td><strong>Trip Ticket</strong></td>
<td>Personnel &amp; Eqpt mgmt</td>
</tr>
<tr>
<td><strong>Agora</strong></td>
<td>Apps portal &amp; feedback</td>
</tr>
<tr>
<td><strong>Nowtu</strong></td>
<td>User empowered training</td>
</tr>
<tr>
<td><strong>Paranav</strong></td>
<td>Airborne navigation</td>
</tr>
<tr>
<td><strong>WAM</strong></td>
<td>Weapons and Ammunitions</td>
</tr>
<tr>
<td><strong>Dimmer</strong></td>
<td>Night time ops</td>
</tr>
<tr>
<td><strong>Transtalk</strong></td>
<td>Pashto and Dari Translator</td>
</tr>
<tr>
<td><strong>MSD</strong></td>
<td>Minimum safety distance</td>
</tr>
<tr>
<td><strong>Medical training modules</strong></td>
<td>Administer morphine, Apply tourniquet</td>
</tr>
<tr>
<td><strong>Gammapix</strong></td>
<td>Radiation detection</td>
</tr>
<tr>
<td><strong>Transheat</strong></td>
<td>Patrol heatmap</td>
</tr>
<tr>
<td><strong>Debrief</strong></td>
<td>Patrol review</td>
</tr>
<tr>
<td><strong>TIGR</strong></td>
<td>Events, Places, People, Reports</td>
</tr>
<tr>
<td><strong>Julian date converter</strong></td>
<td></td>
</tr>
</tbody>
</table>

Works with network | With or without network | With or without network | Works without network
<table>
<thead>
<tr>
<th><strong>Example Apps (1:2)</strong></th>
<th><strong>Invented in theater</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PLI</strong></td>
<td><strong>WAM</strong></td>
</tr>
<tr>
<td>Blue-force tracking</td>
<td>Weapons and Ammunition</td>
</tr>
<tr>
<td><strong>DASH</strong></td>
<td><strong>Dimmer</strong></td>
</tr>
<tr>
<td>Messages</td>
<td>Night time ops</td>
</tr>
<tr>
<td><strong>RTC</strong></td>
<td><strong>Hello</strong></td>
</tr>
<tr>
<td>Chat rooms</td>
<td>Pashto and Dari Translator</td>
</tr>
<tr>
<td><strong>Collab</strong></td>
<td><strong>MSD</strong></td>
</tr>
<tr>
<td>Live Collaboration</td>
<td>Minimum safety distance</td>
</tr>
<tr>
<td><strong>SAR</strong></td>
<td><strong>Transtalk</strong></td>
</tr>
<tr>
<td>Coordinated Search &amp; Rescue</td>
<td>TransHeat</td>
</tr>
<tr>
<td><strong>STREAM</strong></td>
<td><strong>Paranav</strong></td>
</tr>
<tr>
<td>Sensor streaming</td>
<td>Airborne navigation</td>
</tr>
<tr>
<td><strong>SpeedTest</strong></td>
<td><strong>TIGR</strong></td>
</tr>
<tr>
<td>Network performance</td>
<td>Events, Places, People, Reports</td>
</tr>
<tr>
<td><strong>Maps</strong></td>
<td><strong>TRAQ</strong></td>
</tr>
<tr>
<td>High res maps navigation, layers</td>
<td>GPS trace &amp; media</td>
</tr>
<tr>
<td><strong>Mapdraw</strong></td>
<td><strong>Trip Ticket</strong></td>
</tr>
<tr>
<td>Operational graphics</td>
<td>Personnel &amp; Eqpt mgmt</td>
</tr>
<tr>
<td><strong>WhoDat</strong></td>
<td><strong>Agora</strong></td>
</tr>
<tr>
<td>Local population</td>
<td>Apps portal &amp; feedback</td>
</tr>
<tr>
<td><strong>Patrolview</strong></td>
<td><strong>Nowtu</strong></td>
</tr>
<tr>
<td>Collector &amp; Viewer</td>
<td>User empowered training</td>
</tr>
<tr>
<td><strong>Patrolheatmap</strong></td>
<td><strong>Debrief</strong></td>
</tr>
<tr>
<td><strong>WAM</strong></td>
<td>Patrol review</td>
</tr>
<tr>
<td><strong>Medical training modules</strong></td>
<td><strong>Transheat</strong></td>
</tr>
<tr>
<td>Administer morphine, Apply tourniquet</td>
<td>Patrol heatmap</td>
</tr>
<tr>
<td><strong>GammaPix</strong></td>
<td><strong>TIGR</strong></td>
</tr>
<tr>
<td>Minimum safety distance</td>
<td>Events, Places, People, Reports</td>
</tr>
<tr>
<td><strong>Julian date converter</strong></td>
<td><strong>ACOZ</strong></td>
</tr>
<tr>
<td>Coordinated Search &amp; Rescue</td>
<td>Julian date converter</td>
</tr>
</tbody>
</table>
When is this software trustworthy?

<table>
<thead>
<tr>
<th>Example Apps (1:2)</th>
<th>Invented in theater</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DARPA</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Example Apps (1:2)</strong></td>
<td></td>
</tr>
<tr>
<td>PLI</td>
<td>Blue-force tracking</td>
</tr>
<tr>
<td>DASH</td>
<td>Messages</td>
</tr>
<tr>
<td>RTC</td>
<td>Chat rooms</td>
</tr>
<tr>
<td>Collab</td>
<td>Live Collaboration</td>
</tr>
<tr>
<td>SAR</td>
<td>Coordinated Search &amp; Rescue</td>
</tr>
<tr>
<td>STREAM</td>
<td>Sensor streaming</td>
</tr>
<tr>
<td>SpeedTest</td>
<td>Network performance</td>
</tr>
<tr>
<td>Works with network</td>
<td></td>
</tr>
<tr>
<td><strong>Maps</strong></td>
<td>High res maps navigation, layers</td>
</tr>
<tr>
<td>Mapdraw</td>
<td>Operational graphics</td>
</tr>
<tr>
<td>TRAQ</td>
<td>GPS trace &amp; media</td>
</tr>
<tr>
<td>Transheat</td>
<td>Patrol heatmap</td>
</tr>
<tr>
<td>Debrief</td>
<td>Patrol review</td>
</tr>
<tr>
<td>TIGR</td>
<td>Events, Places, People, Reports</td>
</tr>
<tr>
<td>Works or without network</td>
<td></td>
</tr>
<tr>
<td><strong>Patrolview</strong></td>
<td>Collector &amp; Viewer</td>
</tr>
<tr>
<td>WhoDat</td>
<td>Local population</td>
</tr>
<tr>
<td>Trip Ticket</td>
<td>Personnel &amp; Eqpt mgmt</td>
</tr>
<tr>
<td>Agora</td>
<td>Apps portal &amp; feedback</td>
</tr>
<tr>
<td>Nowtu</td>
<td>User empowered training</td>
</tr>
<tr>
<td>Paranav</td>
<td>Airborne navigation</td>
</tr>
<tr>
<td>Works or without network</td>
<td></td>
</tr>
<tr>
<td><strong>WAM</strong></td>
<td>Weapons and Ammunitions</td>
</tr>
<tr>
<td>Dimmer</td>
<td>Night time ops</td>
</tr>
<tr>
<td>Hello</td>
<td></td>
</tr>
<tr>
<td><strong>Gammapix</strong></td>
<td>Minimum safety distance</td>
</tr>
<tr>
<td>Medical training modules</td>
<td>Administer morphine, Apply tourniquet</td>
</tr>
<tr>
<td>Transheat</td>
<td>Patrol heatmap</td>
</tr>
<tr>
<td>Debrief</td>
<td>Patrol review</td>
</tr>
<tr>
<td>TIGR</td>
<td>Events, Places, People, Reports</td>
</tr>
<tr>
<td>Works without network</td>
<td></td>
</tr>
<tr>
<td><strong>SAR</strong></td>
<td>Coordinated Search &amp; Rescue</td>
</tr>
<tr>
<td>STREAM</td>
<td>Sensor streaming</td>
</tr>
<tr>
<td>SpeedTest</td>
<td>Network performance</td>
</tr>
<tr>
<td>Works with network</td>
<td></td>
</tr>
<tr>
<td><strong>SpeedTest</strong></td>
<td></td>
</tr>
</tbody>
</table>
“When we trust a system, we trust it will behave as we expect it to.”

— Bruce Schneier

Trust “involves the risk of failure or harm to the trustor if the trustee will not behave as desired.”

— Wikipedia, *Trust (social sciences)*
When is this software trustworthy?

<table>
<thead>
<tr>
<th>DARPA Example Apps (1:2)</th>
<th>Invented in theater</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PLI</strong></td>
<td>Blue-force tracking</td>
</tr>
<tr>
<td><strong>DASH</strong></td>
<td>Messages</td>
</tr>
<tr>
<td><strong>RTC</strong></td>
<td>Chat rooms</td>
</tr>
<tr>
<td><strong>Collab</strong></td>
<td>Live Collaboration</td>
</tr>
<tr>
<td><strong>SAR</strong></td>
<td>Coordinated Search &amp; Rescue</td>
</tr>
<tr>
<td><strong>STREAM</strong></td>
<td>Sensor streaming</td>
</tr>
<tr>
<td><strong>SpeedTest</strong></td>
<td>Network performance</td>
</tr>
<tr>
<td><strong>Maps</strong></td>
<td>High res maps navigation, layers</td>
</tr>
<tr>
<td><strong>Mapdraw</strong></td>
<td>Operational graphics</td>
</tr>
<tr>
<td><strong>TRAQ</strong></td>
<td>GPS trace &amp; media</td>
</tr>
<tr>
<td><strong>Transheat</strong></td>
<td>Patrol heatmap</td>
</tr>
<tr>
<td><strong>Debrief</strong></td>
<td>Patrol review</td>
</tr>
<tr>
<td><strong>TIGR</strong></td>
<td>Events, Places, People, Reports</td>
</tr>
<tr>
<td><strong>Patrolview</strong></td>
<td>Collector &amp; Viewer</td>
</tr>
<tr>
<td><strong>WhoDat</strong></td>
<td>Local population</td>
</tr>
<tr>
<td><strong>Trip Ticket</strong></td>
<td>Personnel &amp; Eqpt mgmt</td>
</tr>
<tr>
<td><strong>Agora</strong></td>
<td>Apps portal &amp; feedback</td>
</tr>
<tr>
<td><strong>Nowtu</strong></td>
<td>User empowered training</td>
</tr>
<tr>
<td><strong>Paranav</strong></td>
<td>Airborne navigation</td>
</tr>
<tr>
<td><strong>SAR</strong></td>
<td>Medical training modules</td>
</tr>
<tr>
<td><strong>SAR</strong></td>
<td>Administer morphine, Apply tourniquet</td>
</tr>
<tr>
<td><strong>SAR</strong></td>
<td>Sensor streaming</td>
</tr>
<tr>
<td><strong>STREAM</strong></td>
<td>GPS trace &amp; media</td>
</tr>
<tr>
<td><strong>TIGR</strong></td>
<td>Events, Places, People, Reports</td>
</tr>
<tr>
<td><strong>Debrief</strong></td>
<td>Patrol review</td>
</tr>
<tr>
<td><strong>Nowtu</strong></td>
<td>User empowered training</td>
</tr>
<tr>
<td><strong>Paranav</strong></td>
<td>Airborne navigation</td>
</tr>
<tr>
<td><strong>SAR</strong></td>
<td>Medical training modules</td>
</tr>
<tr>
<td><strong>SAR</strong></td>
<td>Administer morphine, Apply tourniquet</td>
</tr>
<tr>
<td><strong>SAR</strong></td>
<td>Sensor streaming</td>
</tr>
<tr>
<td><strong>STREAM</strong></td>
<td>GPS trace &amp; media</td>
</tr>
<tr>
<td><strong>TIGR</strong></td>
<td>Events, Places, People, Reports</td>
</tr>
<tr>
<td><strong>Debrief</strong></td>
<td>Patrol review</td>
</tr>
<tr>
<td><strong>Nowtu</strong></td>
<td>User empowered training</td>
</tr>
<tr>
<td><strong>Paranav</strong></td>
<td>Airborne navigation</td>
</tr>
</tbody>
</table>

Sunday, March 17, 13
### DARPA Example Apps (1:2)

<table>
<thead>
<tr>
<th>Category</th>
<th>Apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLI</td>
<td>Blue-force tracking</td>
</tr>
<tr>
<td>DASH</td>
<td>Messages</td>
</tr>
<tr>
<td>RTC</td>
<td>Chat rooms</td>
</tr>
<tr>
<td>Collab</td>
<td>Live Collaboration</td>
</tr>
<tr>
<td>SAR</td>
<td>Coordinated Search &amp; Rescue</td>
</tr>
<tr>
<td>STREAM</td>
<td>Sensor streaming</td>
</tr>
<tr>
<td>SpeedTest</td>
<td>Network performance</td>
</tr>
<tr>
<td>Maps</td>
<td>High res maps navigation, layers</td>
</tr>
<tr>
<td>Mapdraw</td>
<td>Operational graphics</td>
</tr>
<tr>
<td>TRAQ</td>
<td>GPS trace &amp; media</td>
</tr>
<tr>
<td>Transheat</td>
<td>Patrol heatmap</td>
</tr>
<tr>
<td>Debrief</td>
<td>Patrol review</td>
</tr>
<tr>
<td>TIGR</td>
<td>Events, Places, People, Reports</td>
</tr>
<tr>
<td>Julian date converter</td>
<td>Coordinated Search &amp; Rescue</td>
</tr>
<tr>
<td>Trip Ticket</td>
<td>Personnel &amp; Eqpt mgmt</td>
</tr>
<tr>
<td>Agora</td>
<td>Apps portal &amp; feedback</td>
</tr>
<tr>
<td>Nowtu</td>
<td>User empowered training</td>
</tr>
<tr>
<td>Paranav</td>
<td>Airborne navigation</td>
</tr>
<tr>
<td>Gammapix</td>
<td>Radiation detection</td>
</tr>
<tr>
<td>ACOZ</td>
<td>Minimum safety distance</td>
</tr>
<tr>
<td>Medical training</td>
<td>Administer morphine, Apply tourniquet</td>
</tr>
</tbody>
</table>

When we can predict it will not misbehave.
When is this software misbehaving?

<table>
<thead>
<tr>
<th>DARPA Example Apps (1:2)</th>
<th>Invented in theater</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PLI</strong> Blue-force tracking</td>
<td></td>
</tr>
<tr>
<td><strong>DASH</strong> Messages</td>
<td></td>
</tr>
<tr>
<td><strong>RTC</strong> Chat rooms</td>
<td></td>
</tr>
<tr>
<td><strong>Collab</strong> Live Collaboration</td>
<td></td>
</tr>
<tr>
<td><strong>SAR</strong> Coordinated Search &amp; Rescue</td>
<td></td>
</tr>
<tr>
<td><strong>STREAM</strong> Sensor streaming</td>
<td></td>
</tr>
<tr>
<td><strong>SpeedTest</strong> Network performance</td>
<td></td>
</tr>
<tr>
<td><strong>Maps</strong> High res maps navigation, layers</td>
<td></td>
</tr>
<tr>
<td><strong>Mapdraw</strong> Operational graphics</td>
<td></td>
</tr>
<tr>
<td><strong>TRAQ</strong> GPS trace &amp; media</td>
<td></td>
</tr>
<tr>
<td><strong>Transheat</strong> Patrol heatmap</td>
<td></td>
</tr>
<tr>
<td><strong>Debrief</strong> Patrol review</td>
<td></td>
</tr>
<tr>
<td><strong>TIGR</strong> Events, Places People, Reports</td>
<td></td>
</tr>
<tr>
<td><strong>Patrolview</strong> Collector &amp; Viewer</td>
<td></td>
</tr>
<tr>
<td><strong>WhoDat</strong> Local population</td>
<td></td>
</tr>
<tr>
<td><strong>WhoDat</strong> Local population</td>
<td></td>
</tr>
<tr>
<td><strong>WAM</strong> Weapons and Ammunitions</td>
<td></td>
</tr>
<tr>
<td><strong>Dimmer</strong> Night time ops</td>
<td></td>
</tr>
<tr>
<td><strong>Hello بسم الله</strong> Pashto and Dari Translator</td>
<td></td>
</tr>
<tr>
<td><strong>MSD</strong> Minimum safety distance</td>
<td></td>
</tr>
<tr>
<td><strong>Transheat</strong> Patrol heatmap</td>
<td></td>
</tr>
<tr>
<td><strong>Agora</strong> Apps portal &amp; feedback</td>
<td></td>
</tr>
<tr>
<td><strong>Nowtu</strong> User empowered training</td>
<td></td>
</tr>
<tr>
<td><strong>Paranav</strong> Airborne navigation</td>
<td></td>
</tr>
<tr>
<td><strong>Debrief</strong> Patrol review</td>
<td></td>
</tr>
<tr>
<td><strong>TIGR</strong> Events, Places People, Reports</td>
<td></td>
</tr>
<tr>
<td><strong>Patrolview</strong> Collector &amp; Viewer</td>
<td></td>
</tr>
<tr>
<td><strong>WhoDat</strong> Local population</td>
<td></td>
</tr>
<tr>
<td><strong>WAM</strong> Weapons and Ammunitions</td>
<td></td>
</tr>
<tr>
<td><strong>Dimmer</strong> Night time ops</td>
<td></td>
</tr>
<tr>
<td><strong>Hello بسم الله</strong> Pashto and Dari Translator</td>
<td></td>
</tr>
<tr>
<td><strong>MSD</strong> Minimum safety distance</td>
<td></td>
</tr>
<tr>
<td><strong>Transheat</strong> Patrol heatmap</td>
<td></td>
</tr>
<tr>
<td><strong>Agora</strong> Apps portal &amp; feedback</td>
<td></td>
</tr>
<tr>
<td><strong>Nowtu</strong> User empowered training</td>
<td></td>
</tr>
<tr>
<td><strong>Paranav</strong> Airborne navigation</td>
<td></td>
</tr>
<tr>
<td><strong>Debrief</strong> Patrol review</td>
<td></td>
</tr>
<tr>
<td><strong>TIGR</strong> Events, Places People, Reports</td>
<td></td>
</tr>
<tr>
<td><strong>Patrolview</strong> Collector &amp; Viewer</td>
<td></td>
</tr>
<tr>
<td><strong>WhoDat</strong> Local population</td>
<td></td>
</tr>
<tr>
<td><strong>WAM</strong> Weapons and Ammunitions</td>
<td></td>
</tr>
<tr>
<td><strong>Dimmer</strong> Night time ops</td>
<td></td>
</tr>
<tr>
<td><strong>Hello بسم الله</strong> Pashto and Dari Translator</td>
<td></td>
</tr>
<tr>
<td><strong>MSD</strong> Minimum safety distance</td>
<td></td>
</tr>
<tr>
<td><strong>Transheat</strong> Patrol heatmap</td>
<td></td>
</tr>
<tr>
<td><strong>Agora</strong> Apps portal &amp; feedback</td>
<td></td>
</tr>
<tr>
<td><strong>Nowtu</strong> User empowered training</td>
<td></td>
</tr>
<tr>
<td><strong>Paranav</strong> Airborne navigation</td>
<td></td>
</tr>
<tr>
<td><strong>Debrief</strong> Patrol review</td>
<td></td>
</tr>
<tr>
<td><strong>TIGR</strong> Events, Places People, Reports</td>
<td></td>
</tr>
<tr>
<td><strong>Patrolview</strong> Collector &amp; Viewer</td>
<td></td>
</tr>
<tr>
<td><strong>WhoDat</strong> Local population</td>
<td></td>
</tr>
<tr>
<td><strong>WAM</strong> Weapons and Ammunitions</td>
<td></td>
</tr>
<tr>
<td><strong>Dimmer</strong> Night time ops</td>
<td></td>
</tr>
<tr>
<td><strong>Hello بسم الله</strong> Pashto and Dari Translator</td>
<td></td>
</tr>
<tr>
<td><strong>MSD</strong> Minimum safety distance</td>
<td></td>
</tr>
<tr>
<td><strong>Transheat</strong> Patrol heatmap</td>
<td></td>
</tr>
<tr>
<td><strong>Agora</strong> Apps portal &amp; feedback</td>
<td></td>
</tr>
<tr>
<td><strong>Nowtu</strong> User empowered training</td>
<td></td>
</tr>
<tr>
<td><strong>Paranav</strong> Airborne navigation</td>
<td></td>
</tr>
<tr>
<td><strong>Debrief</strong> Patrol review</td>
<td></td>
</tr>
<tr>
<td><strong>TIGR</strong> Events, Places People, Reports</td>
<td></td>
</tr>
<tr>
<td><strong>Patrolview</strong> Collector &amp; Viewer</td>
<td></td>
</tr>
<tr>
<td><strong>WhoDat</strong> Local population</td>
<td></td>
</tr>
<tr>
<td><strong>WAM</strong> Weapons and Ammunitions</td>
<td></td>
</tr>
<tr>
<td><strong>Dimmer</strong> Night time ops</td>
<td></td>
</tr>
<tr>
<td><strong>Hello بسم الله</strong> Pashto and Dari Translator</td>
<td></td>
</tr>
<tr>
<td><strong>MSD</strong> Minimum safety distance</td>
<td></td>
</tr>
<tr>
<td><strong>Transheat</strong> Patrol heatmap</td>
<td></td>
</tr>
<tr>
<td><strong>Agora</strong> Apps portal &amp; feedback</td>
<td></td>
</tr>
<tr>
<td><strong>Nowtu</strong> User empowered training</td>
<td></td>
</tr>
<tr>
<td><strong>Paranav</strong> Airborne navigation</td>
<td></td>
</tr>
<tr>
<td><strong>Debrief</strong> Patrol review</td>
<td></td>
</tr>
<tr>
<td><strong>TIGR</strong> Events, Places People, Reports</td>
<td></td>
</tr>
<tr>
<td><strong>Patrolview</strong> Collector &amp; Viewer</td>
<td></td>
</tr>
<tr>
<td><strong>WhoDat</strong> Local population</td>
<td></td>
</tr>
<tr>
<td><strong>WAM</strong> Weapons and Ammunitions</td>
<td></td>
</tr>
<tr>
<td><strong>Dimmer</strong> Night time ops</td>
<td></td>
</tr>
<tr>
<td><strong>Hello بسم الله</strong> Pashto and Dari Translator</td>
<td></td>
</tr>
<tr>
<td><strong>MSD</strong> Minimum safety distance</td>
<td></td>
</tr>
<tr>
<td><strong>Transheat</strong> Patrol heatmap</td>
<td></td>
</tr>
<tr>
<td><strong>Agora</strong> Apps portal &amp; feedback</td>
<td></td>
</tr>
<tr>
<td><strong>Nowtu</strong> User empowered training</td>
<td></td>
</tr>
<tr>
<td><strong>Paranav</strong> Airborne navigation</td>
<td></td>
</tr>
<tr>
<td><strong>Debrief</strong> Patrol review</td>
<td></td>
</tr>
<tr>
<td><strong>TIGR</strong> Events, Places People, Reports</td>
<td></td>
</tr>
<tr>
<td><strong>Patrolview</strong> Collector &amp; Viewer</td>
<td></td>
</tr>
<tr>
<td><strong>WhoDat</strong> Local population</td>
<td></td>
</tr>
<tr>
<td><strong>WAM</strong> Weapons and Ammunitions</td>
<td></td>
</tr>
<tr>
<td><strong>Dimmer</strong> Night time ops</td>
<td></td>
</tr>
</tbody>
</table>
When is this software misbehaving?

When it sends GPS data to a non-white listed URL.
When is this software misbehaving?

Gammapix
Radiation detection

When it uses the network.
When is this software misbehaving?

When it uses the camera.

*SpeedTest*

Network performance
When is this software misbehaving?

When it raises an uncaught exception.

**Paranav**

Airborne navigation

When it raises an uncaught exception.
When is this software misbehaving?

The navigation software on this device is designed to provide route suggestions and does not replace the driver attentiveness or the requirement to adhere to regulatory requirements and directions.

Sorry!
The application NAVIGON (process com.navigon.navigator_checkout_eu40) has stopped unexpectedly. Please try again.

Do not operate this unit while driving.

OK
To trust software,
we must predict its (mis)behavior.
To trust software,
we must predict its (mis)behavior.

I build tools and techniques for
predicting the behavior of software.
Automated Program Analysis for Cybersecurity (APAC)
Automated Program Analysis for Cybersecurity (APAC)

Result of Abstract Interpretation of all the apks uploaded...

APK | Reports and Graphs
--- | ---
UltraCoolMap2 | Report: Framework APIs dumps
             | Report: Rough abstract profiling
             | Debug information

UltraCoolMap2

Stop and reconfigure
Analyze another apk
Get me out...
Result of Abstract Interpretation of all the apks uploaded...

APK	Reports and Graphs

UltraCoolMap2	Report: Framework APIs dumps
               Report: Rough abstract profiling
               Debug information

Refresh

Stop and reconfigure
Analyze another apk
Get me out...

APIs used in the app

- (#s(compact-meth org/apache/http/impl/client/DefaultHttpClient/ ()). #s(and-perms (android.permission.INTERNET)))
- (#s(compact-meth org/apache/http/client/methods/HttpGet/ ((object java/net/URI))). #f)
- (#s(compact-meth org/apache/http/client/HttpExecute (object org/apache/http/client/methods/HttpUriRequest))). #f)
The Least Privileged Permission System (LPPS) Detection Report

The app asks for the following permissions:

- `android.permission.ACCESS_FINE_LOCATION`
- `android.permission.WRITE_INTERNAL_STORAGE`

Permissions that are used in the app (based on current API knowledge):

- `android.permissionINTERNET`

LPP Violation: permissions requested in the manifest but not used in the app:

- `android.permission.ACCESS_FINE_LOCATION`
- `android.permission.WRITE_INTERNAL_STORAGE`
The Least Privileged Permission System (LPPS) Detection Report

The app asks for the following permissions:

- android.permission.ACCESS_FINE_LOCATION
- android.permission.WRITE_INTERNAL_STORAGE

Permissions that are used in the app (based on current API knowledge):

- android.permissionINTERNET

LPP Violation: permissions requested in the manifest but not used in the app:

- android.permission.ACCESS_FINE_LOCATION
- android.permission.WRITE_INTERNAL_STORAGE
The Least Privileged Permission System (LPPS) Detection Report

The app asks for the following permissions....

State Details

Frame Pointer: #s(frame-pointer com.ultracoolmap/ UltraCoolMapActivity$ReallyBadName/doInBackground())

Time: ((new-instance v0 org.apache/http/impl/client/DefaultHttpClient))

Statements:

- ((invoke-direct (v0) org/apache/http/impl/client/DefaultHttpClient/)
  (line 289))

- (new-instance v1 org/apache/http/client/methods/HttpGet)
  (const/4 v2 0)
  (aget-object v2 v4 v2)
  (invoke-direct
   (v1 v2)
   org/apache/http/client/methods/HttpGet/(object java/net/URI))

<table>
<thead>
<tr>
<th>AndorsTrail</th>
<th>SplitTimer</th>
<th>SMSBackup</th>
</tr>
</thead>
<tbody>
<tr>
<td>AndroidGame</td>
<td>SuperNote</td>
<td>SMSBlocker</td>
</tr>
<tr>
<td>AndroidPrivacyGuard_E</td>
<td>SuperSoduko</td>
<td>SMSPopup</td>
</tr>
<tr>
<td>Butane</td>
<td>SysMon</td>
<td>SysWatcherA</td>
</tr>
<tr>
<td>CalcA</td>
<td>SysWatcherB</td>
<td>SourceViewer</td>
</tr>
<tr>
<td>CalcB</td>
<td>TextSecure</td>
<td>UltraCoolMap</td>
</tr>
<tr>
<td>ConnectBot</td>
<td>TodoList</td>
<td>YARR</td>
</tr>
<tr>
<td>CountdownTimer</td>
<td>Word Helper</td>
<td>AndroidsFortune</td>
</tr>
<tr>
<td>FunDraw</td>
<td>AndBible</td>
<td>CalcC</td>
</tr>
<tr>
<td>MorseCode</td>
<td>AndroidPrivacyGuard_M</td>
<td>CalcE</td>
</tr>
<tr>
<td>MyDrawA</td>
<td>BatteryIndicator</td>
<td>ColorMatcher</td>
</tr>
<tr>
<td>MyDrawC</td>
<td>CalcF</td>
<td>FullControl</td>
</tr>
<tr>
<td>NewsCollator</td>
<td>MediaFun</td>
<td>KitteyKittey</td>
</tr>
<tr>
<td>PasswordSaver</td>
<td>MyDrawD</td>
<td>Orienteering2</td>
</tr>
<tr>
<td>PersistantAssistant</td>
<td>OpenGPSTracker</td>
<td>Sanity</td>
</tr>
<tr>
<td>SmartWebCam</td>
<td>Orienteering1</td>
<td>TomDroid</td>
</tr>
<tr>
<td>SMSReminder</td>
<td>PicViewer</td>
<td>WiFinder</td>
</tr>
<tr>
<td>SourceViewer</td>
<td>Collaboration ShareLoc</td>
<td></td>
</tr>
<tr>
<td>AndorsTrail</td>
<td>SplitTimer</td>
<td>SMSBackup</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td>------------</td>
</tr>
<tr>
<td>AndroidGame</td>
<td>SuperNote</td>
<td>SMSBlocker</td>
</tr>
<tr>
<td>AndroidPrivacyGuard_E</td>
<td>SuperSoduko</td>
<td>SMSPopup</td>
</tr>
<tr>
<td>Butane</td>
<td>SysMon</td>
<td>SysWatcherA</td>
</tr>
<tr>
<td>CalcA</td>
<td>SysWatcherB</td>
<td>SourceViewer</td>
</tr>
<tr>
<td>CalcB</td>
<td>TextSecure</td>
<td>UltraCoolMap</td>
</tr>
<tr>
<td>ConnectBot</td>
<td>TodoList</td>
<td>YARR</td>
</tr>
<tr>
<td>CountdownTimer</td>
<td>Word_Help</td>
<td>AndroidsFortune</td>
</tr>
<tr>
<td>FunDraw</td>
<td>AndBible</td>
<td>CalcC</td>
</tr>
<tr>
<td>MorseCode</td>
<td>AndroidPrivacyGuard_M</td>
<td>CalcE</td>
</tr>
<tr>
<td>MyDrawA</td>
<td>BatteryIndicator</td>
<td>ColorMatcher</td>
</tr>
<tr>
<td>MyDrawC</td>
<td>CalcF</td>
<td>FullControl</td>
</tr>
<tr>
<td>NewsCollator</td>
<td>MediaFun</td>
<td>KitteyKittey</td>
</tr>
<tr>
<td>PasswordSaver</td>
<td>MyDrawD</td>
<td>Orienteering2</td>
</tr>
<tr>
<td>PersistantAssistant</td>
<td>OpenGPSTracker</td>
<td>Sanity</td>
</tr>
<tr>
<td>SmartWebCam</td>
<td>Orienteering1</td>
<td>TomDroid</td>
</tr>
<tr>
<td>SMSReminder</td>
<td>PicViewer</td>
<td>WiFinder</td>
</tr>
<tr>
<td>SourceViewer</td>
<td>Collaboration_ShareLoc</td>
<td></td>
</tr>
<tr>
<td>AndorsTrail</td>
<td>SplitTimer</td>
<td>SMSBackup</td>
</tr>
<tr>
<td>-------------</td>
<td>------------</td>
<td>-----------</td>
</tr>
<tr>
<td>AndroidGame</td>
<td>SuperNote</td>
<td>SMSBlocker</td>
</tr>
<tr>
<td>AndroidPrivacyGuard_E</td>
<td>SuperSoduko</td>
<td>SMSPopup</td>
</tr>
<tr>
<td>Butane</td>
<td>SysMon</td>
<td>SysWatcherA</td>
</tr>
<tr>
<td>CalcA</td>
<td>SysWatcherB</td>
<td>SourceViewer</td>
</tr>
<tr>
<td>CalcB</td>
<td>TextSecure</td>
<td>UltraCoolMap</td>
</tr>
<tr>
<td>ConnectBot</td>
<td>TodoList</td>
<td>YARR</td>
</tr>
<tr>
<td>CountdownTimer</td>
<td>Word Helper</td>
<td>Androids Fortune</td>
</tr>
<tr>
<td>FunDraw</td>
<td>AndBible</td>
<td>CalcC</td>
</tr>
<tr>
<td>MorseCode</td>
<td>AndroidPrivacyGuard_M</td>
<td>CalcE</td>
</tr>
<tr>
<td>MyDrawA</td>
<td>BatteryIndicator</td>
<td>ColorMatcher</td>
</tr>
<tr>
<td>MyDrawC</td>
<td>CalcF</td>
<td>FullControl</td>
</tr>
<tr>
<td>NewsCollator</td>
<td>MediaFun</td>
<td>KitteyKittey</td>
</tr>
<tr>
<td>PasswordSaver</td>
<td>MyDrawD</td>
<td>Orienteering2</td>
</tr>
<tr>
<td>PersistantAssistant</td>
<td>OpenGPSTracker</td>
<td>Sanity</td>
</tr>
<tr>
<td>SmartWebCam</td>
<td>Orienteering1</td>
<td>TomDroid</td>
</tr>
<tr>
<td>SMSReminder</td>
<td>PicViewer</td>
<td>WiFinder</td>
</tr>
<tr>
<td>SourceViewer</td>
<td>Collaboration ShareLoc</td>
<td></td>
</tr>
</tbody>
</table>
I build tools and techniques for predicting the behavior of software.
I build tools and techniques for soundly and effectively predicting the behavior of software.
AnaDroid
Robust, Reliable Software and Trustworthy Systems
OUTLINE:

AnaDroid
OUTLINE:

Understanding prediction

AnaDroid
OUTLINE:

1. Understanding prediction
2. Systematic approach
3. AnaDroid
OUTLINE:

Understanding prediction

Systematic approach

AnaDroid

Results
Part I: Understanding Prediction
To trust software, we must predict its (mis)behavior.

\[ x.m() \]
To trust software, we must predict its (mis)behavior.

```java
class XYZ {
    public int m() {
        return 42;
    }
}

class Example {
    public void f(XYZ x) {
        return x.m();
    }
}
```
To trust software, we must predict its (mis)behavior.

```java
public void f(XYZ x) {
    return x.m();
}
```
To trust software, we must predict its (mis)behavior.

```java
public void f(XYZ x) {
    return x.m();
}
```
Modern software uses computational values.

To predict control flow, we must predict data flow. To predict data flow, we must predict control flow.
Modern software uses computational values.

To predict control flow, we must predict data flow. To predict data flow, we must predict control flow.
public class RemoteCallbackList
    extends Object

java.lang.Object
    --> android.os.RemoteCallbackList<E extends android.os.IInterface>

Class Overview

Takes care of the grunt work of maintaining a list of remote interfaces, typically for the use of performing callbacks from a Service to its clients. In particular, this:

- Keeps track of a set of registered IInterface callbacks, taking care to identify them through their underlying unique IBinder (by calling IInterface.asBinder()).
- Attaches a IBinder.DeathRecipient to each registered interface, so that it can be cleaned out of the list if its process goes away.
- Performs locking of the underlying list of interfaces to deal with multithreaded incoming calls and a thread-safe way to iterate over a snapshot of the list without holding its lock.

To use this class, simply create a single instance along with your service, and call its register(E) and unregister(E) methods as client register and unregister with your service. To call back on to the registered clients, use beginBroadcast(), getBroadcastItem(int), and finishBroadcast().

If a registered callback's process goes away, this class will take care of automatically removing it from the list. If you want to do additional work in this situation, you can create a subclass that implements the onCallbackDied(E) method.
Class Observable

public class Observable extends Object

This class represents an observable object, or "data" in the model-view paradigm. It can be subclassed to represent an object that the application wants to have observed.

An observable object can have one or more observers. An observer may be any object that implements interface Observer. After an observable instance changes, an application calling the observable's notifyObservers method causes all of its observers to be notified.

Constructor Summary

Observable()
Construct an Observable with zero Observers.

Method Summary

void addObserver(Observer o)
Adds an observer to the set of observers for this object, provided that it is not the same as some observer already in the set.

protected void clearChanged()
Indicates that this object has no longer changed, or that it has already notified all of its observers of its most recent change, so that the hasChanged method will now return false.

int countObservers()
Returns the number of observers of this observable object.

void deleteObserver(Observer o)
Deletes an observer from the set of observers of this object.

void deleteObservers()
Clears the observer list so that this object no longer has any observers.

boolean hasChanged()
Tests if this object has changed.

void notifyObservers()
If this object has changed, as indicated by the hasChanged method, then notify all of its observers and then call the clearChanged method to indicate that this object has no longer changed.

void notifyObservers(Object arg)
If this object has changed, as indicated by the hasChanged method, then notify all of its observers and then call the clearChanged method to indicate that this object has no longer changed.

protected void setChanged()
Marks this observable object as having been changed; the hasChanged method will now return true.
1. Introduction

This section is non-normative.

The XMLHttpRequest object implements an interface exposed by a scripting engine that allows scripts to perform HTTP client functionality, such as submitting form data or loading data from a server. It is the ECMAScript HTTP API.

The name of the object is XMLHttpRequest for compatibility with the Web, though each component of this name is potentially misleading. First, the object supports any text based format, including XML. Second, it can be used to make requests over both HTTP and HTTPS (some implementations support protocols in addition to HTTP and HTTPS, but that functionality is not covered by this specification). Finally, it supports "requests" in a broad sense of the term as it pertains to HTTP; namely all activity involved with HTTP requests or responses for the defined HTTP methods.

Some simple code to do something with data from an XML document fetched over the network:

```javascript
function test(data) {
   // taking care of data
}

function handler() {
   if(this.readyState == 4 && this.status == 200) {
      // so far so good
      if(this.responseXML != null && this.responseXML.getElementById('test').firstChild.data) // success!
         test(this.responseXML.getElementById('test').firstChild.data);
      else
         test(null);
   } else if (this.readyState == 4 && this.status != 200) {
      // fetched the wrong page or network error...
      test(null);
   }
}

var client = new XMLHttpRequest();
client.onreadystatechange = handler;
client.open('GET', 'unicorn.xml');
client.send();
```
Modern software is higher-order.

Node.js

Evented I/O for V8 JavaScript.

An example of a web server written in Node which responds with "Hello World" for every request.

```javascript
var http = require('http');
http.createServer(function (req, res) {
    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('Hello World
');
}).listen(8124, '127.0.0.1');
console.log('Server running at http://127.0.0.1:8124/');
```

To run the server, put the code into a file example.js and execute it with the node program:

```bash
% node example.js
Server running at http://127.0.0.1:8124/
```
Object-Oriented Design Libraries

One of the interesting things about Ruby is the high level of abstraction between design and implementation, which is higher than in many other languages. Ruby supports design at the same level as in other languages, thereby simplifying the process of implementing design patterns.

To help in this process, Ruby has support for the following design patterns:

- **The Visitor pattern** (Design Patterns) is a way to implement the strategy pattern without having to know the type of the object being visited.
- **Delegation** is a way of composition in Ruby that can be done using standard methods.
- **The Singleton pattern** is a way to ensure that a particular class exists at a time.
- **The Observer pattern** implements a set of interested objects when a particular event occurs.

Normally, all four of these strategies need to be implemented. With Ruby, they can be implemented and transparently.

---

**Library: observer**

The Observer pattern, also known as Publish/Subscribe, provides a simple mechanism for one object to inform a set of interested third-party objects when its state changes.

In the Ruby implementation, the notifying class mixes in the `observable` module, which provides the methods for managing the associated observer objects.

```ruby
add_observer(obj)
  # Add obj as an observer on this object. obj will now receive notifications.

delete_observer(obj)
  # Delete obj as an observer on this object. It will no longer receive notifications.

delete_observers
  # Delete all observers associated with this object.

countObservers
  # Return the count of observers associated with this object.

changed(newState=true)
  # Set the changed state of this object. Notifications will be sent only if the changed state is true.

changed?
  # Query the changed state of this object.

notifyObservers(*args)
  # If this object's changed state is true, invoke the update method in each currently associated observer in turn, passing it the given arguments. The changed state is then set to false.
```

The observers must implement the `update` method to receive notifications.
1.3 Functions as values

OCaml is a functional language: functions in the full mathematical sense are supported and can be passed around freely just as any other piece of data. For instance, here is a deriv function that takes any float function as argument and returns an approximation of its derivative function:

```ocaml
# let deriv f dx = function x -> (f(x +. dx) -. f(x)) /. dx;
val deriv : (float -> float) -> float -> float -> float = <fun>

# let sin' = deriv sin 1e-6;
val sin' : float -> float = <fun>

# sin' pi;
- : float = -1.000000000013961143
```

Even function composition is definable:

```ocaml
# let compose f g = function x -> f(g(x));;
val compose : ('a -> 'b) -> ('c -> 'a) -> 'c -> 'b = <fun>

# let cos2 = compose square cos;;
val cos2 : float -> float = <fun>
```

Functions that take other functions as arguments are called “functionals”, or “higher-order functions”. Functionals are especially useful to provide iterators or similar generic operations over a data structure. For instance, the standard OCaml library provides a List.map functional that applies a given function to each element of a list, and returns the list of the results:

```ocaml
# List.map (function n -> n * 2 + 1) [0;1;2;3;4];;
- : int list = [1; 3; 5; 7; 9]
```

This functional, along with a number of other list and array functionals, is predefined because it is often useful, but there is nothing magic with it: it can easily be defined as follows.
10.2. **functools** — Higher-order functions and operations on callable objects

**Source code:** Lib/functools.py

The **functools** module is for higher-order functions: functions that act on or return other functions. In general, any callable object can be treated as a function for the purposes of this module.

The **functools** module defines the following functions:

```
functools.cmp_to_key(func)
```

Transform an old-style comparison function to a key function. Used with tools that accept key functions (such as `sorted()`, `min()`, `max()`, `heapq.nlargest()`, `heapq.nsmallest()`, `itertools.groupby()`). This function is primarily used as a transition tool for programs being converted from Python 2 which supported the use of comparison functions.
Modern software uses computational values.
Modern software uses computational values.

To predict its behavior, we need flow analysis.
0. INTRODUCTION

A method is described to extract from an untyped λ-expression information about the sequence of intermediate λ-expressions obtained during its evaluation. This information can be used to give "safe positive answers" to questions involving termination or nontermination of the evaluation, dependence of one subexpression on another and type errors encountered while applying δ rules, thus providing an alternative to techniques of Morris and Levy ([Mor68], [Lev75]). The method works by building a "safe description" of the set of states entered by a call-by-name interpreter and analyzing this description. A similar and more complete analysis of a call-by-value interpreter may be found in [Jon81].

From a flow analysis viewpoint these results extend existing interprocedural analysis methods to include call-by-name and the use of functions both as arguments to other functions and as the results returned by them. Further, the method naturally handles both local and global variables, extending [Cou77a] and [Sha80]. It seems clear that other traditional analyses such as available expressions, constant propagation, etc. can be carried out in this framework.

The main emphasis is on development of the framework and showing its relation to abstract interpretation, rather than on its efficient use in applications. A simplified and optimized version of the method would have applications in the efficient compilation of λ-calculus-based programming languages such as LISP, SCHEME and SASL ([McC63], [Ste75], [Tur76]).

The method provides a general way to find safe approximate descriptions of computations by algorithms which manipulate recursive data structures. It is thus not limited to the λ-calculus, but may be applied to analyze any programming language whose semantics can be implemented by an appropriate definitional interpreter.

Another application would be to extend the method to the flow analysis of denotational definitions of programming languages. This could be used in semantics-directed compiler generation as described in [JoS80], and provided the initial motivation for this study.

Related work

Lambda calculus evaluators have been studied in [Böh72], [Lan64], [McG70], [Plu75], [Roy72], [Sch80] and [Weg68]. Sufficient conditions for termination of
FLOW ANALYSIS OF LAMBDA EXPRESSIONS
(Preliminary Version)

Neil D. Jones
Aarhus University, Denmark

0. INTRODUCTION

A method is described to extract from an untyped λ-expression information about
the sequence of intermediate λ-expressions obtained during its evaluation.

The information can be used to give "safe positive answers" to questions involving
termination or nontermination of the evaluation, dependence of one subexpression
and type errors encountered while applying δ rules, thus providing an elegant
version of techniques of Morris and Levy ([Mor68], [Lev75]). The method works
"safe description" of the set of states entered by a call-by-name interpreter
analyzing this description. A similar and more complete analysis of a call-by-value
interpreter may be found in [Jon81].

From a flow analysis viewpoint these results extend existing techniques
analysis methods to include call-by-name and the use of functions both
and to other functions and as the results returned by them. Further, the method
handles both local and global variables, extending [Cou77a] and [Scho81].
It is clear that other traditional analyses such as available expressions, context,
implementation, etc. can be carried out in this framework.

The main emphasis is on development of the framework and showing
how it can be applied to abstraction interpretation, rather than on its efficient use in applications.
A primitive and optimal version of the method would have applications in
the particularly simple compilation of λ-calculus-based programming languages such as LISP,
and SASL ([McC63], [Ste75], [Turn76]).

The method provides a general way to find safe approximate descriptions
computations by algorithms which manipulate recursive data structures,
not limited to the λ-calculus, but may be applied to analyze any program
language whose semantics can be implemented by an appropriate definition

Another application would be to extend the method to the flow analysis
completely abstract formulations of programming languages. This could be used in
simple graph directed compiler generation as described in [Jo68], and provided the inspiration
for this study.

Related work

Lambda calculus evaluators have been studied in [Boh72], [Lan64],
[Pln75], [Rey72], [Sch80] and [Wie68]. Sufficient conditions for term

Control-Flow Analysis of Functional Programs

JAN MIDTGAARD, Aarhus University

We present a survey of control-flow analysis of functional programs, which has been the subject of extensive
investigation throughout the past 30 years. Analyses of the control flow of functional programs have been
formulated in multiple settings and have led to many different approximations, starting with the seminal
works of Jones, Shivers, and Sestoft. In this article, we survey control-flow analysis of functional programs
by structuring the multitude of formulations and approximations and comparing them.

Categories and Subject Descriptors: D.3.2 [Programming Languages]: Language Classifications—
Applicative functional languages; F.3.1 [Logics and Meanings of Programs]: Specifying and Verifying
Reasoning about Programs

General Terms: Languages, Theory, Verification

Additional Key Words and Phrases: Control-flow analysis, higher-order functions

ACM Reference Format:

2012), 33 pages.

DOI = 10.1145/2187671.2187672 http://doi.acm.org/10.1145/2187671.2187672

1. INTRODUCTION

Since the introduction of high-level languages and compilers, much work has been
devoted to approximating, at compile time, which values the variables of a given pro-
gram may not be available until runtime. A call-by-value function may not be available until runtime. A
language on the textual structure of the program, since it determines the exact control
flow of the program, for example, as a flow chart. On the other hand, in a language
with higher-order functions, the operator of a function call may not be apparent from
the text of the program: it can be the result of a computation and therefore, the
called function may not be available until runtime. A control-flow analysis approximates at
compile time which functions may be applied at runtime, that is, it determines an
approximate control flow of a given program.

Prerequisites. We assume some familiarity with program analysis in general and
with control-flow analysis in particular. For a tutorial or an introduction to the area, we
refer to Nielson et al. [1999]. We also assume familiarity with functional programming
and a basic acquaintance with continuation-passing style (CPS) [Steele Jr. 1978] and

Part of this work was done with support of the Carlsberg Foundation and an INRIA post-doc grant.
Authors’ addresses: J. Midtgaard, Department of Computer Science, Aarhus University, Aabogade 34,
DK-8200 Aarhus N., Denmark; email: jm0@cs.au.dk.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted
without fee provided that copies are not made or distributed for profit or commercial advantage and that
copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for
components of this work owned by others than ACM must be honored. Abstracting with credit is permitted.
To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this
work in other works requires prior specific permission and/or a fee. Permissions may be requested from
Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212)
869-0481, or permissions@acm.org.
© 2012 ACM 0360-0300/2012/06-ART10 $10.00
DOI 10.1145/2187671.2187672 http://doi.acm.org/10.1145/2187671.2187672
Existing analyses (and their complexities)
function twice(f, x) { return f(f(x)); }; 

twice(sqr, 4);  twice(dbl, 5);
function twice(f, x) { return f(f(x)); };  
twice(sqr, 4);  
twice(dbl, 5);
function twice(f, x) { return f(f(x)); };

twice(sqr, 4);
twice(dbl, 5);
function twice(f,x) { return f(f(x)); };
function twice(f, x) { return f(f(x)); };

twice(sqr, 4);

twice(dbl, 5);
function twice(f,x) { return f(f(x)); };  

twice(sqr,4);   twice(dbl,5);

{4}  {sqr}  

{sqrt(sqrt(4))}  

twice(dbl,5);
function twice(f,x) { return f(f(x)); }; 

twice(sqr,4);  
twice(dbl,5);  

{ sqr }  { 4 }  

{ sqr(sqr(4)) }
function twice(f, x) { return f(f(x)); }; 

twice(sqr, 4); 
twice(dbl, 5); 

{sqr, dbl} {4, 5}
function twice(f, x) { return f(f(x)); }; 

twice(sqr, 4); 

twice(dbl, 5); 

{ sqr(sqr(4)) } 

{ sqr, dbl } { 4, 5 }
function twice(f, x) { return f(f(x)); };  

{dbl(dbl(5))}  

{sqrt(sqr(4))}  

{sqrt, dbl} {4, 5}
function twice(f, x) { return f(f(x)); };

twice(sqr, 4);          twice(dbl, 5);

{sqr(sqr(4))}          {dbl(dbl(5)),
                      sqr(sqr(4)),
                      dbl(sqr(5)), ...}
function twice(f, x) { return f(f(x)); };
function twice(f,x) { return f(f(x)); };

twice(sqr,4); twice(dbl,5);
function twice(f, x) { return f(f(x)); };
function twice(f, x) { return f(f(x));; }

twice(sqr, 4);
twice(dbl, 5);

twice(sqr, 4);  # {sqr}  # {sqr(sqr(4))}
twice(dbl, 5);  # {sqr}  # {4}
function twice(f, x) { return f(f(x)); };

twice(sqr, 4);  // twice(sqr(sqr(4)))
twice(dbl, 5);  // twice(dbl(dbl(5)))
function twice(f, x) { return f(f(x)); };
Precision

SubOCFA
Simple closure
OCFA
ICFA
...

[ICFP'07, SAS'08, ICFP'08]
$k$-CFA

---

1-CFA

---

0-CFA

Simple closure

Sub-0-CFA

---

[ICFP’07, SAS’08, ICFP’08]
Precision

- $k$CFA
- 1CFA
- 0CFA
- Simple closure
- Sub0CFA

EXPTIME

PTIME

[ICFP'07, SAS'08, ICFP'08]
Key insight: analysis is a kind of evaluation
PTIME

EXPTIME

PTIME

[ICFP’07, SAS’08, ICFP’08]
Precision

- kCFA
- mCFA
- ...
- 1CFA
- 1CFA
- EXPTIME
- PTIME
- PTIME
- Simple closure
- ...

[ICFP’07, SAS’08, ICFP’08, PLDI’10]
\begin{align*}
&\text{[con]} \quad \overline{C}, \bar{\rho} \models c' \text{ always} \\
&\text{[var]} \quad \overline{C}, \bar{\rho} \models x^\ell \text{ iff } \bar{\rho}(x) \subseteq \overline{C}(\ell) \\
&\text{[fn]} \quad \overline{C}, \bar{\rho} \models (\text{fn } x \rightarrow e_0)^\ell \text{ iff } \{\text{fn } x \rightarrow e_0\} \subseteq \overline{C}(\ell) \\
&\text{[fun]} \quad \overline{C}, \bar{\rho} \models (\text{fun } f \ x \rightarrow e_0)^\ell \text{ iff } \{\text{fun } f \ x \rightarrow e_0\} \subseteq \overline{C}(\ell) \\
&\text{[app]} \quad \overline{C}, \bar{\rho} \models (t_1^\ell \ t_2^\ell)^\ell \\
&\quad \text{iff } (\overline{C}, \bar{\rho}) \models t_1^0 \land (\overline{C}, \bar{\rho}) \models t_2^0 \land \\
&\quad \{\forall (\text{fun } f \ x \rightarrow t_1^0) \in \overline{C}(\ell_1) : \\
&\quad \overline{C}(\ell_2) \subseteq \bar{\rho}(x) \land \overline{C}(\ell_0) \subseteq \overline{C}(\ell) \land \\
&\quad \{\forall (\text{fun } f \ x \rightarrow t_2^0) \in \overline{C}(\ell_1) : \\
&\quad \overline{C}(\ell_2) \subseteq \bar{\rho}(x) \land \overline{C}(\ell_0) \subseteq \overline{C}(\ell) \land \\
&\quad \{\text{fun } f \ x \rightarrow t_2^0\} \subseteq \overline{\rho}(f)) \\
&\text{[if]} \quad \overline{C}, \bar{\rho} \models (\text{if } t_1^0 \text{ then } t_1^\ell \text{ else } t_2^\ell)^\ell \\
&\quad \text{iff } (\overline{C}, \bar{\rho}) \models t_1^0 \land \\
&\quad \overline{C}(\ell_1) \subseteq \overline{C}(\ell) \land \overline{C}(\ell_2) \subseteq \overline{C}(\ell) \\
&\text{[let]} \quad \overline{C}, \bar{\rho} \models (\text{let } x = t_1^\ell \text{ in } t_2^\ell)^\ell \\
&\quad \text{iff } (\overline{C}, \bar{\rho}) \models t_1^\ell \land (\overline{C}, \bar{\rho}) \models t_2^0 \land \\
&\quad \overline{C}(\ell_1) \subseteq \bar{\rho}(x) \land \overline{C}(\ell_2) \subseteq \overline{C}(\ell) \\
&\text{[op]} \quad \overline{C}, \bar{\rho} \models (t_1^\ell \ op \ t_2^\ell)^\ell \\
&\quad \text{iff } (\overline{C}, \bar{\rho}) \models t_1^\ell \land (\overline{C}, \bar{\rho}) \models t_2^\ell \\
\end{align*}

Table 3.1: Abstract Control Flow Analysis (Subsections 3.1.1 and 3.1.2).

\begin{align*}
&\text{[var]} \quad \rho \vdash x^\ell \rightarrow v^\ell \text{ if } x \in \text{dom}(\rho) \text{ and } v = \rho(x) \\
&\text{[fn]} \quad \rho \vdash (\text{fn } x \rightarrow e_0)^\ell \rightarrow (\text{close } (\text{fn } x \rightarrow e_0) \text{ in } \rho_0)^\ell \\
&\quad \text{ where } \rho_0 = \rho \mid \text{FV}(\text{fn } x \rightarrow e_0) \\
&\text{[fun]} \quad \rho \vdash (\text{fun } f \ x \rightarrow e_0)^\ell \rightarrow (\text{close } (\text{fun } f \ x \rightarrow e_0) \text{ in } \rho_0)^\ell \\
&\quad \text{ where } \rho_0 = \rho \mid \text{FV}(\text{fun } f \ x \rightarrow e_0) \\
&\text{[app]} \quad \rho \vdash i e_1 \rightarrow i e_1^\ell \\
&\quad \rho \vdash (i e_1 i e_2)^\ell \rightarrow (i e_1^\ell i e_2^\ell)^\ell \\
&\text{[app2]} \quad \rho \vdash i e_2 \rightarrow i e_2^\ell \\
&\quad \rho \vdash (v_1^\ell i e_2^\ell)^\ell \rightarrow (v_1^\ell i e_2^\ell)^\ell \\
&\text{[app3]} \quad \rho \vdash ((\text{close } (\text{fn } x \rightarrow e_1) \text{ in } \rho_1)^\ell v_2^\ell)^\ell \rightarrow \\
&\quad (\text{bind } \rho_1[x \mapsto v_2] \text{ in } e_1)^\ell \\
&\text{[appfun]} \quad \rho \vdash ((\text{close } (\text{fun } f \ x \rightarrow e_1) \text{ in } \rho_1)^\ell v_2^\ell)^\ell \rightarrow \\
&\quad (\text{bind } \rho_2[x \mapsto v_2] \text{ in } e_1)^\ell \\
&\quad \text{ where } \rho_2 = \rho_1[f \mapsto \text{close } (\text{fun } f \ x \rightarrow e_1) \text{ in } \rho_1] \\
&\text{[bind]} \quad \rho \vdash \text{bind } \rho_1 \text{ in } i e_1^\ell \rightarrow \text{bind } \rho_1 \text{ in } i e_1^\ell \\
&\text{[bind2]} \quad \rho \vdash \text{bind } \rho_1 \text{ in } v_1^\ell \rightarrow v_1^\ell \\
&\text{[if]} \quad \rho \vdash (i e_0 \text{ then } e_1 \text{ else } e_2)^\ell \rightarrow (i e_0^\ell \text{ then } e_1 \text{ else } e_2^\ell)^\ell \\
&\text{[if2]} \quad \rho \vdash (\text{if } e_0 \text{ then } e_1 \text{ else } e_2)^\ell \rightarrow (\text{if } e_0^\ell \text{ then } e_1 \text{ else } e_2^\ell)^\ell \\
&\text{[if3]} \quad \rho \vdash (\text{if } \text{false} \text{ then } e_1 \text{ else } e_2)^\ell \rightarrow (\text{if } \text{false} \text{ then } e_1 \text{ else } e_2)^\ell \\
&\text{[let]} \quad \rho \vdash \text{let } x = i e_1 \text{ in } e_2^\ell \rightarrow (\text{let } x = i e_1^\ell \text{ in } e_2)^\ell \\
&\text{[let2]} \quad \rho \vdash \text{let } x = v_1^\ell \text{ in } e_2^\ell \rightarrow (\text{bind } \rho_0[x \mapsto v] \text{ in } e_2)^\ell \\
&\quad \text{ where } \rho_0 = \rho \mid \text{FV}(e_2) \\
&\text{[op]} \quad \rho \vdash i e_1 \rightarrow i e_1^\ell \\
&\quad \rho \vdash (i e_1 i e_2)^\ell \rightarrow (i e_1^\ell i e_2^\ell)^\ell \\
&\text{[op2]} \quad \rho \vdash i e_2 \rightarrow i e_2^\ell \\
&\quad \rho \vdash (v_1^\ell i e_2^\ell)^\ell \rightarrow (v_1^\ell i e_2^\ell)^\ell \\
&\text{[op3]} \quad \rho \vdash (v_1^\ell \text{ op } v_2^\ell)^\ell \rightarrow v^\ell \\
&\quad \text{ if } v = v_1 \text{ op } v_2 \\
\end{align*}

Table 3.3: The Structural Operational Semantics of FUN (part 2).
\[ \text{var} \quad \rho \vdash x^\ell \rightarrow v^\ell \quad \text{if } x \in \text{dom}(\rho) \text{ and } v = \rho(x) \]

\[ \text{fn} \quad \rho \vdash (\text{fn } x \Rightarrow e_0)^\ell \rightarrow (\text{close } (\text{fn } x \Rightarrow e_0) \text{ in } \rho_0)^\ell \quad \text{where } \rho_0 = \rho \mid \text{FV}(\text{fn } x \Rightarrow e_0) \]

\[ \text{fun} \quad \rho \vdash (\text{fun } f x \Rightarrow e_0)^\ell \rightarrow (\text{close } (\text{fun } f x \Rightarrow e_0) \text{ in } \rho_0)^\ell \quad \text{where } \rho_0 = \rho \mid \text{FV}(\text{fun } f x \Rightarrow e_0) \]

\[ \text{app}_1 \quad \rho \vdash ie_1 \rightarrow ie_1^\ell \]
\[ \rho \vdash (ie_1 \ ie_2)^\ell \rightarrow (ie_1^\ell \ ie_2^\ell)^\ell \]

\[ \text{app}_2 \quad \rho \vdash ie_2 \rightarrow ie_2^\ell \]
\[ \rho \vdash (v_1^\ell \ ie_2)^\ell \rightarrow (v_1^\ell \ ie_2^\ell)^\ell \]

\[ \text{app}_m \quad \rho \vdash ((\text{close } (\text{fn } x \Rightarrow e_1) \text{ in } \rho_1)^\ell \ v_2^\ell)^\ell \rightarrow (\text{bind } \rho_1[x \mapsto v_2] \text{ in } e_1)^\ell \]

\[ \text{app}_m \quad \rho \vdash ((\text{close } (\text{fun } f x \Rightarrow e_1) \text{ in } \rho_1)^\ell \ v_2^\ell)^\ell \rightarrow (\text{bind } \rho_2[x \mapsto v_2] \text{ in } e_1)^\ell \quad \text{where } \rho_2 = \rho_1[f \mapsto \text{close } (\text{fun } f x \Rightarrow e_1) \text{ in } \rho_1] \]

\[ \text{bind}_1 \quad \rho \vdash (\text{bind } \rho_1 \text{ in } ie_1)^\ell \rightarrow (\text{bind } \rho_1 \text{ in } ie_1^\ell)^\ell \]

\[ \text{bind}_2 \quad \rho \vdash (\text{bind } \rho_1 \text{ in } v_1^\ell)^\ell \rightarrow v_1^\ell \]

\[ \text{if}_1 \quad \rho \vdash ie_0 \rightarrow ie_0^\ell \]
\[ \rho \vdash (\text{if } ie_0 \text{ then } e_1 \text{ else } e_2)^\ell \rightarrow (\text{if } ie_0^\ell \text{ then } e_1 \text{ else } e_2)^\ell \]

\[ \text{if}_2 \quad \rho \vdash (\text{if } \text{true}^\ell \text{ then } t_1^\ell \text{ else } t_2^\ell)^\ell \rightarrow t_1^\ell \]

\[ \text{if}_3 \quad \rho \vdash (\text{if } \text{false}^\ell \text{ then } t_1^\ell \text{ else } t_2^\ell)^\ell \rightarrow t_2^\ell \]

\[ \text{let}_1 \quad \rho \vdash (\text{let } x = ie_1 \text{ in } e_2)^\ell \rightarrow (\text{let } x = ie_1^\ell \text{ in } e_2)^\ell \]

\[ \text{let}_2 \quad \rho \vdash (\text{let } x = v_1^\ell \text{ in } e_2)^\ell \rightarrow (\text{bind } \rho_0[x \mapsto v] \text{ in } e_2)^\ell \quad \text{where } \rho_0 = \rho \mid \text{FV}(e_2) \]

\[ \text{op}_1 \quad \rho \vdash ie_1 \rightarrow ie_1^\ell \]
\[ \rho \vdash (ie_1 \ op \ ie_2)^\ell \rightarrow (ie_1^\ell \ op \ ie_2^\ell)^\ell \]

\[ \text{op}_2 \quad \rho \vdash ie_2 \rightarrow ie_2^\ell \]
\[ \rho \vdash (v_1^\ell \ op \ ie_2)^\ell \rightarrow (v_1^\ell \ op \ ie_2^\ell)^\ell \]

\[ \text{op}_3 \quad \rho \vdash (v_1^\ell \ op \ v_2^\ell)^\ell \rightarrow v^\ell \quad \text{if } v = v_1 \ op v_2 \]

Table 3.3: The Structural Operational Semantics of FUN (part 2).
Table 3.3: The Structural Operational Semantics of Fun (part 2).
\[\begin{align*}
\text{[con]} \quad & (\tilde{C}, \tilde{\rho}) \models c^t \text{ always} \\
\text{[var]} \quad & (\tilde{C}, \tilde{\rho}) \models x^t \text{ if } \tilde{\rho}(x) \subseteq \tilde{C}(\ell) \\
\text{[fn]} \quad & (\tilde{C}, \tilde{\rho}) \models (\text{fn } x \rightarrow e_0)^t \text{ if } \{\text{fn } x \rightarrow e_0\} \subseteq \tilde{C}(\ell) \\
\text{[fun]} \quad & (\tilde{C}, \tilde{\rho}) \models (\text{fun } f x \rightarrow e_0)^t \text{ if } \{\text{fun } f x \rightarrow e_0\} \subseteq \tilde{C}(\ell) \\
\text{[app]} \quad & (\tilde{C}, \tilde{\rho}) \models (t_1^t \ t_2^t)^t \text{ if } (\tilde{C}, \tilde{\rho}) \models t_1^t \land (\tilde{C}, \tilde{\rho}) \models t_2^t \land \\
& \quad \quad \quad (\forall (\text{fn } x \rightarrow t_0^t) \in \tilde{C}(\ell_1) : \\
& \quad \quad \quad \quad \quad \quad (\tilde{C}, \tilde{\rho}) \models t_0^t \land \\
& \quad \quad \quad \quad \quad \quad \tilde{C}(\ell_2) \subseteq \tilde{\rho}(x) \land \tilde{C}(\ell_0) \subseteq \tilde{C}(\ell) \land \\
& \quad \quad \quad \quad \quad \quad (\forall (\text{fun } f x \rightarrow t_0^t) \in \tilde{C}(\ell_1) : \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad (\tilde{C}, \tilde{\rho}) \models t_0^t \land \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \tilde{C}(\ell_2) \subseteq \tilde{\rho}(x) \land \tilde{C}(\ell_0) \subseteq \tilde{C}(\ell) \land \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \{\text{fun } f x \rightarrow t_0^t\} \subseteq \tilde{\rho}(f) \\
\text{[if]} \quad & (\tilde{C}, \tilde{\rho}) \models (\text{if } t_0^t \text{ then } t_1^t \text{ else } t_2^t)^t \text{ if } (\tilde{C}, \tilde{\rho}) \models t_0^t \land \\
& \quad \quad \quad \quad \quad \quad (\tilde{C}, \tilde{\rho}) \models t_1^t \land (\tilde{C}, \tilde{\rho}) \models t_2^t \land \\
& \quad \quad \quad \quad \quad \quad \tilde{C}(\ell_1) \subseteq \tilde{C}(\ell) \land \tilde{C}(\ell_2) \subseteq \tilde{C}(\ell) \\
\text{[let]} \quad & (\tilde{C}, \tilde{\rho}) \models (\text{let } x = t_1^t \text{ in } t_2^t)^t \text{ if } (\tilde{C}, \tilde{\rho}) \models t_1^t \land (\tilde{C}, \tilde{\rho}) \models t_2^t \land \\
& \quad \quad \quad \quad \quad \quad \tilde{C}(\ell_1) \subseteq \tilde{\rho}(x) \land \tilde{C}(\ell_2) \subseteq \tilde{C}(\ell) \\
\text{[op]} \quad & (\tilde{C}, \tilde{\rho}) \models (t_1^t \ \text{op} \ t_2^t)^t \text{ if } (\tilde{C}, \tilde{\rho}) \models t_1^t \land (\tilde{C}, \tilde{\rho}) \models t_2^t \\
\end{align*}\]

Table 3.1: Abstract Control Flow Analysis (Subsections 3.1.1 and 3.1.2).
\[
\begin{align*}
((\lambda x^\ell e) v^\ell) & \ell_a \quad \rightarrow \quad e[v^\ell / x^\ell] & \text{SUBST} \\
(n^\ell e) & \ell_a \quad \rightarrow \quad (\text{blame } \lambda R)^\ell_a & \text{APP-ERROR} \\
(\text{if}0 \; v^\ell e_1 e_2) & \ell \quad \rightarrow \quad e_1 & \text{IF0-TRUE} \\
(\text{if}0 \; v^\ell e_1 e_2) & \ell \quad \rightarrow \quad e_2 & \text{IF0-FALSE} \\
(\text{any} f^\ell \equiv v^\ell e) & \ell_e \quad \rightarrow \quad v^\ell f & \text{ANY} \\
((e_1 e_2 \text{ and} f^\ell \equiv f^\ell e) & \ell_f^+ \ell_f^- \equiv e^\ell c) \ell_c \quad \rightarrow \quad e_1[v^\ell c_2, e^\ell f] & \text{ANY-TRIP} \\
(\text{int} f^\ell & \equiv n^\ell e) \ell_e \quad \rightarrow \quad n^\ell & \text{INT-INT} \\
(\text{int} f^\ell & \equiv v^\ell f) \ell_e \quad \rightarrow \quad (\text{blame } f R)^\ell e & \text{INT-LAM} \\
((e_1 e_2 \text{ int } f^\ell) & \ell_f^+ \ell_f^- \equiv e^\ell c_2) \ell_c \quad \rightarrow \quad (e_1 n^\ell c_2, e^\ell f) & \text{INT-TRIP-INT} \\
((e_1 e_2 \text{ int } f^\ell) & \ell_f^+ \ell_f^- \equiv v^\ell f) \ell_e \quad \rightarrow \quad (\text{blame } f R)^\ell e & \text{INT-TRIP-LAM} \\
((e_1 \rightarrow e_2) & \ell_f \equiv \ell_{\text{fail}}) \ell_e \quad \rightarrow \quad ((e_1 \rightarrow e_2) \ell_f \equiv \ell_{\text{fail}} e) \ell_e & \text{LAM-LAM} \\
((e_1 \rightarrow e_2) & \ell_f \equiv \ell_{\text{fail}}) \ell_e \quad \rightarrow \quad (\text{blame } f R)^\ell e & \text{LAM-TRIP-LAM} \\
((e_1 e_2 \; (e_1 \rightarrow e_2) & \ell_f^+ \ell_f^- \equiv \ell_{\text{fail}} e) \ell_e \quad \rightarrow \quad (e_1 \rightarrow (e_1 \rightarrow e_2) \ell_f \equiv \ell_{\text{fail}} e) \ell_e / e^\ell f) & \text{LAM-TRIP-INT} \\
((e_1 e_2 \; (e_1 \rightarrow e_2) & \ell_f^+ \ell_f^- \equiv \ell_{\text{fail}} e) \ell_e \quad \rightarrow \quad (\text{blame } f R)^\ell e & \text{LAM-TRIP-INT} \\
(((e_1 \rightarrow e_2) & \ell_f \equiv \ell_{\text{fail}} e) \; w^\ell v) \ell_e \quad \rightarrow \quad (e_2 \equiv (\ell_{\text{fail}} (e_1 \equiv w^\ell v) \; \text{lub} + (e_1) \; \text{lub} - (e_2)) \; \text{lub} + (e_2)) & \text{SPLIT}
\end{align*}
\]

Figure 7. Reduction rules.

Semantics of contracts
### Table 1. Constraints creation for source-sink pairs.

<table>
<thead>
<tr>
<th>Source \ Sink</th>
<th>(\ell_n \ell_1 \cdots)</th>
<th>(\ell_1 \ell_2 \cdots)</th>
<th>(\ell_1 \ell_2 \cdots)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\ell_n \ell_1 \cdots)</td>
<td>(\ell_n \ell_1 \cdots)</td>
<td>&amp;</td>
<td>&amp;</td>
</tr>
<tr>
<td>(\ell_1 \ell_2 \cdots)</td>
<td>&amp;</td>
<td>(</td>
<td>\ell_1</td>
</tr>
<tr>
<td>(\ell_1 \ell_2 \cdots)</td>
<td>&amp;</td>
<td>(</td>
<td>\ell_1</td>
</tr>
<tr>
<td>(\ell_1 \ell_2 \cdots)</td>
<td>&amp;</td>
<td>(</td>
<td>\ell_1</td>
</tr>
<tr>
<td>(\ell_1 \ell_2 \cdots)</td>
<td>&amp;</td>
<td>(</td>
<td>\ell_1</td>
</tr>
<tr>
<td>(\ell_1 \ell_2 \cdots)</td>
<td>&amp;</td>
<td>(</td>
<td>\ell_1</td>
</tr>
<tr>
<td>(\ell_1 \ell_2 \cdots)</td>
<td>&amp;</td>
<td>(</td>
<td>\ell_1</td>
</tr>
<tr>
<td>(\ell_1 \ell_2 \cdots)</td>
<td>&amp;</td>
<td>(</td>
<td>\ell_1</td>
</tr>
<tr>
<td>(\ell_1 \ell_2 \cdots)</td>
<td>&amp;</td>
<td>(</td>
<td>\ell_1</td>
</tr>
</tbody>
</table>

### Analysis of contracts
Table 1. Constraints creation for source-sink pairs.

<table>
<thead>
<tr>
<th>Source-Sink Pairs</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a, b)</td>
<td>(a, b) ⊆ (a, b)</td>
</tr>
<tr>
<td>(c, d)</td>
<td>(c, d) ⊆ (c, d)</td>
</tr>
<tr>
<td>(e, f)</td>
<td>(e, f) ⊆ (e, f)</td>
</tr>
</tbody>
</table>

Constraints creation for source-sink pairs.
### Op & Format

<table>
<thead>
<tr>
<th>Op Format</th>
<th>Mnemonic / Syntax</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>76:</td>
<td>invoke-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>direct/range</td>
<td></td>
<td></td>
</tr>
<tr>
<td>77:</td>
<td>invoke-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>static/range</td>
<td></td>
<td></td>
</tr>
<tr>
<td>78:</td>
<td>invoke-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>interface/range</td>
<td></td>
<td></td>
</tr>
<tr>
<td>79..7a</td>
<td>unused</td>
<td>(unused)</td>
<td></td>
</tr>
<tr>
<td>7b..8f</td>
<td>unused</td>
<td>(unused)</td>
<td></td>
</tr>
</tbody>
</table>

A: destination register or pair (4) Perform the identified unary operation on

<table>
<thead>
<tr>
<th>Op Format</th>
<th>Mnemonic / Syntax</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0d 11x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>09 32x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01 12x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1e 11x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1a 21c</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0d 11x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>09 32x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01 12x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1e 11x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1a 21c</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Op & Format

<table>
<thead>
<tr>
<th>Op Format</th>
<th>Mnemonic / Syntax</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>d0..d7</td>
<td>bnpop/lit16</td>
<td>A: destination register (4 bits)</td>
<td>Perform the indicated binary op on the indicated register (first argument) and literal value (second argument), storing the result in the destination register. Note: rsub-int does not have a suffix since this version is the main opcode of its family. Also, see below for details on its semantics.</td>
</tr>
<tr>
<td>d0..d7</td>
<td></td>
<td>B: source register (4 bits)</td>
<td></td>
</tr>
<tr>
<td>d0..d7</td>
<td></td>
<td>C: signed int constant (16 bits)</td>
<td></td>
</tr>
<tr>
<td>d8..e2</td>
<td>bnpop/lit8</td>
<td>A: destination register (8 bits)</td>
<td>Perform the indicated binary op on the indicated register (first argument) and literal value (second argument), storing the result in the destination register. Note: See below for details on the semantics of rsub-int.</td>
</tr>
<tr>
<td>d8..e2</td>
<td></td>
<td>B: source register (8 bits)</td>
<td></td>
</tr>
<tr>
<td>d8..e2</td>
<td></td>
<td>C: signed int constant (8 bits)</td>
<td></td>
</tr>
<tr>
<td>e3..ff</td>
<td></td>
<td>(unused)</td>
<td></td>
</tr>
</tbody>
</table>

### Op & Format

<table>
<thead>
<tr>
<th>Op Format</th>
<th>Mnemonic / Syntax</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b0..cf</td>
<td>bnpop/2addr</td>
<td>A: destination and first source register or pair (4 bits)</td>
<td>Perform the identified binary operation on the two source registers, storing the result in the first source register.</td>
</tr>
<tr>
<td>b0..cf</td>
<td></td>
<td>B: second source register or pair (4 bits)</td>
<td></td>
</tr>
<tr>
<td>b0..cf</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b0..cf</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b0..cf</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b0..cf</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
We need a systematic approach.
Evaluator → Analysis
PART II:
A SYSTEMATIC APPROACH
Abstracting Abstract Machines
\[
e ::= x \mid e \ e \ e \mid \lambda x. e
\]
\[ e ::= x \mid e\ e\ e \mid \lambda x.e \]
$e ::= x \mid e\ e\ e \mid \lambda x.e$
\[
e \ ::= \ x \ | \ e \ e \ | \ \lambda x.e
\]
\(e, \rho, \sigma, \kappa\)
\[ \rho : \text{Var} \rightarrow \text{Addr} \]
\( \rho : \text{Var} \rightarrow \text{Addr} \)

\( \sigma : \text{Addr} \rightarrow \text{Val} \)

\( e, \rho, \sigma, \kappa \)
\[
\begin{align*}
\langle x, \rho, \sigma, \kappa \rangle & \rightarrow \langle v, \rho, \sigma, \kappa \rangle \quad \text{if } v = \sigma(\rho(x)) \\
\langle e_0 \ e_1, \rho, \sigma, \kappa \rangle & \rightarrow \langle e_0, \rho, \sigma, (e_1, \rho) \cdot \kappa \rangle \\
\langle v, \sigma, (e, \rho) \cdot \kappa \rangle & \rightarrow \langle e, \rho, \sigma, v \cdot \kappa \rangle \\
\langle v, \sigma, (\lambda x.e, \rho) \cdot \kappa \rangle & \rightarrow \langle e, \rho[x \mapsto a], \sigma[a \mapsto v], \kappa \rangle
\end{align*}
\]

CESK machine
Felleisen & Friedman, ’88
\[\langle x, \rho, \sigma, \kappa \rangle \mapsto \langle v, \rho, \sigma, \kappa \rangle \text{ if } v = \sigma(\rho(x))\]
\[\langle e_0 \ e_1, \rho, \sigma, \kappa \rangle \mapsto \langle e_0, \rho, \sigma, (e_1, \rho) \cdot \kappa \rangle\]
\[\langle v, \sigma, (e, \rho) \cdot \kappa \rangle \mapsto \langle e, \rho, \sigma, v \cdot \kappa \rangle\]
\[\langle v, \sigma, (\lambda x.e, \rho) \cdot \kappa \rangle \mapsto \langle e, \rho[x \mapsto a], \sigma[a \mapsto v], \kappa \rangle\]
\begin{align*}
\langle x, \rho, \sigma, \kappa \rangle & \rightarrow \langle v, \rho, \sigma, \kappa \rangle \quad \text{if } v = \sigma(\rho(x)) \\
\langle e_0 \ e_1, \rho, \sigma, \kappa \rangle & \rightarrow \langle e_0, \rho, \sigma, (e_1, \rho) \cdot \kappa \rangle \\
\langle v, \sigma, (e, \rho) \cdot \kappa \rangle & \rightarrow \langle e, \rho, \sigma, v \cdot \kappa \rangle \\
\langle v, \sigma, (\lambda x.\ e, \rho) \cdot \kappa \rangle & \rightarrow \langle e, \rho[x \mapsto a], \sigma[a \mapsto v], \kappa \rangle
\end{align*}

CESK machine
Felleisen & Friedman, ’88
\[
\begin{align*}
\langle x, \rho, \sigma, \kappa \rangle & \rightsquigarrow \langle v, \rho, \sigma, \kappa \rangle \quad \text{if } v = \sigma(\rho(x)) \\
\langle e_0 \; e_1, \rho, \sigma, \kappa \rangle & \rightsquigarrow \langle e_0, \rho, \sigma, (e_1, \rho) \cdot \kappa \rangle \\
\langle v, \sigma, (e, \rho) \cdot \kappa \rangle & \rightsquigarrow \langle e, \rho, \sigma, v \cdot \kappa \rangle \\
\langle v, \sigma, (\lambda x.e, \rho) \cdot \kappa \rangle & \rightsquigarrow \langle e, \rho[x \mapsto a], \sigma[a \mapsto v], \kappa \rangle
\end{align*}
\]

CESK machine
Felleisen & Friedman, ’88
$e, \rho, \sigma, \kappa$
\[ e ::= x \mid e \cdot e \mid \lambda x. e \]

\[ \rho : \text{Var} \rightarrow \text{Addr} \]

\[ \sigma : \text{Addr} \rightarrow \text{Val} \]

\[ v ::= (\lambda x. e, \rho) \]

\[ \kappa ::= \text{nil} \mid ([\ ] e, \rho) : : \kappa \mid ([v]) : : \kappa \]

\[ e, \rho, \sigma, \kappa \]
κ ::= nil

⟨ e, σ | nil σ τ ⟩ ↦− → ⟨ e, σ τ ⟩ ↦− → ⟨ e, σ τ ⟩ ↦− → ⟨ e, σ τ ⟩ ↦− → ⟨ e, σ τ ⟩
We cannot predict because the future is undecidable.
We cannot predict because the future is undecidable.

Program analysis

= sound, computable approximations
Key idea:

Evaluator

Analysis
Key idea:

Evaluator

infinite, deterministic transition system

Analysis
Key idea:

Evaluator

infinite, deterministic transition system

finite, non-deterministic transition system
$e, \rho, \sigma, \kappa$
Idea: make it finite
Idea: make it finite

\[ e, \rho, \sigma, \kappa \]
Idea: make it finite

\[ e, \rho, \sigma, \kappa \]
Idea: make it finite

\[ e, \rho, \sigma, \kappa \]

\[ \sigma : \text{Addr} \rightarrow \text{Val} \]
Idea: make it finite

$e, \rho, \tilde{\sigma}, \tilde{\kappa}$

$\epsilon | (e, \rho) \cdot \kappa | v \cdot \kappa$

$\sigma : \text{Addr} \rightarrow \text{Val}$
Idea: make it finite
Idea: make it finite
\[
\begin{align*}
\langle x, \rho, \sigma, \kappa \rangle & \mapsto \langle v, \rho, \sigma, \kappa \rangle & \text{if } v = \sigma(\rho(x)) \\
\langle e_0 \ e_1, \rho, \sigma, \kappa \rangle & \mapsto \langle e_0, \rho, \sigma, (e_1, \rho) \cdot \kappa \rangle \\
\langle v, \sigma, (e, \rho) \cdot \kappa \rangle & \mapsto \langle e, \rho, \sigma, v \cdot \kappa \rangle \\
\langle v, \sigma, (\lambda x. e, \rho) \cdot \kappa \rangle & \mapsto \langle e, \rho[x \mapsto a], \sigma[a \mapsto v], \kappa \rangle
\end{align*}
\]
\[ \langle x, \rho, \sigma, \kappa \rangle \implies \langle v, \rho, \sigma, \kappa \rangle \quad \text{if } v = \sigma(\rho(x)) \]
\[ \langle e_0 \ e_1, \rho, \sigma, \kappa \rangle \implies \langle e_0, \rho, \sigma, (e_1, \rho) \cdot \kappa \rangle \]
\[ \langle v, \sigma, (e, \rho) \cdot \kappa \rangle \implies \langle e, \rho, \sigma, v \cdot \kappa \rangle \]
\[ \langle v, \sigma, (\lambda x.e, \rho) \cdot \kappa \rangle \implies \langle e, \rho[x \mapsto a], \sigma[a \mapsto v], \kappa \rangle \]

\[ \langle x, \rho, \hat{\sigma}, \hat{\kappa} \rangle \implies \langle v, \rho, \hat{\sigma}, \hat{\kappa} \rangle \quad \text{if } v \in \hat{\sigma}(\rho(x)) \]
\[ \langle e_0 \ e_1, \rho, \hat{\sigma}, \hat{\kappa} \rangle \implies \langle e_0, \rho, \hat{\sigma} \sqcup [a \mapsto \hat{\kappa}], (e_1, \rho), a \rangle \]
\[ \langle v, \hat{\sigma}, (e, \rho), a \rangle \implies \langle e, \rho, \hat{\sigma}, v, a \rangle \]
\[ \langle v, \hat{\sigma}, (\lambda x.e, \rho), a \rangle \implies \langle e, \rho[x \mapsto a'], \hat{\sigma} \sqcup [a' \mapsto v], \hat{\kappa} \rangle \quad \text{if } \hat{\kappa} \in \hat{\sigma}(a) \]
\[
\begin{align*}
\langle x, \rho, \sigma, \kappa \rangle & \mapsto \langle v, \rho, \sigma, \kappa \rangle, \\
\langle e_0 \ e_1, \rho, \sigma, \kappa \rangle & \mapsto \langle e_0, \rho, \sigma, (e_1, \rho), \kappa \rangle, \\
\langle v, \sigma, (e, \rho) \cdot \kappa \rangle & \mapsto \langle e, \rho, \sigma, v \cdot \kappa \rangle, \\
\langle v, \sigma, (\lambda x. e, \rho) \cdot \kappa \rangle & \mapsto \langle e, \rho[x \mapsto a], \sigma \sqcup [a \mapsto v], \kappa \rangle, \\
\langle x, \rho, \hat{\sigma}, \hat{\kappa} \rangle & \mapsto \langle v, \rho, \hat{\sigma}, \hat{\kappa} \rangle, \text{ if } v \in \hat{\sigma}(\rho(x)) \\
\langle e_0 \ e_1, \rho, \hat{\sigma}, \hat{\kappa} \rangle & \mapsto \langle e_0, \rho, \hat{\sigma} \sqcup [a \mapsto \hat{\kappa}], (e_1, \rho), a \rangle, \\
\langle v, \hat{\sigma}, (e, \rho), a \rangle & \mapsto \langle e, \rho, \hat{\sigma}, v, a \rangle, \\
\langle v, \hat{\sigma}, (\lambda x. e, \rho), a \rangle & \mapsto \langle e, \rho[x \mapsto a'], \hat{\sigma} \sqcup [a' \mapsto v], \hat{\kappa} \rangle, \\
& \text{ if } \hat{\kappa} \in \hat{\sigma}(a)
\end{align*}
\]
\[ (e, \rho, \hat{\sigma}, \hat{\kappa}) \mapsto (v, \rho, \sigma, \kappa) \]

if

\[ v = \sigma(\rho(x)) \]

\[ (e_0, e_1, \rho, \sigma, \kappa) \mapsto (e_0, \rho, \sigma, \hat{\kappa}) \]

\[ (v, \sigma, \lambda x. e, \rho) \mapsto (e, \rho, \sigma, (v) \hat{\kappa}) \]

\[ \kappa \in \sigma(a) \]

\[ (e_0, \rho, \sigma, \kappa) \mapsto (e_0, \rho, \sigma, \nu a \mapsto v) \]

\[ (v, \sigma, \lambda x. e, \rho) \mapsto (e, \rho, \sigma, v) \hat{\kappa} \]

\[ e ::= x | ee | \lambda x. e \]

\[ \rho : \text{Var} \rightarrow \text{Addr} \]

\[ \sigma : \text{Addr} \rightarrow \text{Val} \]

\[ \hat{\sigma} : \hat{\text{Addr}} \rightarrow \mathbb{P}(\text{Val}) \]

\[ v ::= (\lambda x. e, \rho) \]

\[ \kappa ::= \text{nil} | \nu e, \rho \hat{\kappa} | (v) \hat{\kappa} \]

\[ \hat{\kappa} ::= \text{nil} | \nu e, \rho a | (v) a \]
\[ v, \sigma \langle \sigma, v, \langle \sigma_0 \sigma x.e, x.e, e \rangle \rangle \mapsto \rho, \sigma, a \kappa \sigma \kappa \kappa \mapsto \rho : v, | v \]
Soundness
(the safety of predictions)
\( e, \rho, \sigma, \kappa \equiv e, \rho, \hat{\sigma}, \hat{k} \)
\( e, \rho, \sigma, \kappa \)

\( e, \rho, \sigma, \kappa \)

\( e, \rho, \hat{\sigma}, \hat{\kappa} \)

\( e, \rho, \hat{\sigma}, \hat{\kappa} \)
The navigation software on this device is designed to provide route suggestions but does not replace the need for driver attentiveness and compliance with road signs and regulations. Please ensure to pay attention to the road and follow directions. The application NAVIGON (process com.navigon.navigator_checkout_eu40) has stopped unexpectedly. Please try again. Do not operate this unit while navigating.
If it doesn’t misbehave in the abstract, it doesn’t misbehave.
\[ e ::= x \mid ee \mid \lambda x.e \]
-------------- The Least Privileged Permission System (LPPS) Detection Report --------------

The app asks for the following permissions:

```
(
    android.permission.ACCESS_FINE_LOCATION
    android.permissionINTERNET
    android.permission.WRITE_INTERNAL_STORAGE
)
```

-------------------- Permissions that are used in the app (based on current API knowledge):--------------------

```
    android.permissionINTERNET
```

LPPV Violation: permissions requested in the manifest but not used in the app:

```
    android.permission.ACCESS_FINE_LOCATION
    android.permission.WRITE_INTERNAL_STORAGE
```
Improving precision
\begin{align*}
f(x); \\
\text{function } f(\text{z}) \{ \\
\quad \ldots \\
\quad \text{return;}
\} \\
f(y);
\end{align*}
f(x);

function f(z) {
  ...  
  return;
}

f(y);
f(x);

function f(z) {
...
return;
}

f(y);
function f(z) {
    ...
    return;
}

f(x);

f(y);
f(x);  

function f(z) {
  ...
  return;
}

f(y);
function f(z) {
    ...
    return;
}
\[ \langle e_0 e_1, \rho, \sigma, \kappa \rangle \rightarrow \langle e_0, \rho, \sigma, (\langle e_1, \rho \rangle : : \kappa) \rangle \]

\[ \langle v, \sigma, (\langle e, \rho \rangle : : \kappa) \rangle \rightarrow \langle e, \rho, \sigma, (v[A]) : : \kappa \rangle \]

\[ \langle v, \sigma, (\lambda x.e, \rho) : : \kappa \rangle \rightarrow \langle e, \rho \{ x \mapsto a \}, \sigma \{ a \mapsto v \}, \kappa \rangle \] if \( \kappa \in \sigma(\lambda x.e) \)

\[ e ::= x | ee | \lambda x.e \]

\[ \rho : \text{Var} \rightarrow \text{Addr} \]

\[ \sigma : \text{Addr} \rightarrow \text{Val} \]

\[ \hat{\sigma} : \hat{\text{Addr}} \rightarrow \mathbb{P}(\text{Val}) \]

\[ v ::= (\lambda x.e, \rho) \]

\[ \kappa ::= \text{nil} | (\langle e, \rho \rangle : : \kappa) | (v[A]) : : \kappa \]

\[ \hat{\kappa} ::= \text{nil} | (\langle e, \rho \rangle, a) | (v[A], a) \]
call to $f$ from $\kappa$
return from $f$ to $\kappa$

return from $f$ to $\kappa'$
return from \( f \) to \( \kappa \)

return from \( f \) to \( \kappa' \)
Idea: make it finite

\[ e, \rho, \sigma, \kappa \]

\[ \varepsilon \mid (e, \rho) \cdot \kappa \mid \nu \cdot \kappa \]

\[ \varepsilon \mid (e, \rho), \ a \mid \nu, \ a \]
Idea: make it finite
Idea: make it finite
Idea: make it finite

\[
e, \rho, \hat{\sigma}, \hat{\kappa}
\]

\[
\epsilon \mid (e, \rho) \cdot \kappa \mid v \cdot \kappa
\]

\[
\epsilon \mid (e, \rho), a \mid v, a
\]

\[
\hat{\sigma} : \text{Addr} \rightarrow \mathcal{P}(\text{Val})
\]

\[
\sigma : \text{Addr} \rightarrow \text{Val}
\]
Decidable

Idea: make it finite
\[ \langle x, \rho, \hat{\sigma}, \hat{\kappa} \rangle \longrightarrow \langle v, \rho, \hat{\sigma}, \hat{\kappa} \rangle \quad \text{if } v \in \hat{\sigma}(\rho(x)) \]

\[ \langle e_0 \ e_1, \rho, \hat{\sigma}, \hat{\kappa} \rangle \longrightarrow \langle e_0, \rho, \hat{\sigma} \sqcup [a \mapsto \hat{\kappa}], (e_1, \rho), a \rangle \]

\[ \langle v, \hat{\sigma}, (e, \rho), a \rangle \longrightarrow \langle e, \rho, \hat{\sigma}, v, a \rangle \]

\[ \langle v, \hat{\sigma}, (\lambda x.e, \rho), a \rangle \longrightarrow \langle e, \rho[x \mapsto a'], \hat{\sigma} \sqcup [a' \mapsto v], \hat{\kappa} \rangle \quad \text{if } \hat{\kappa} \in \hat{\sigma}(a) \]
\[ \langle x, \rho, \hat{\sigma}, \hat{\kappa} \rangle \quad \mapsto \quad \begin{align*}
\langle x, \rho, \hat{\sigma}, \hat{\kappa} \rangle & \quad \text{if } v \in \hat{\sigma}(\rho(x)) \\
\langle e_0 \ e_1, \rho, \hat{\sigma}, \hat{\kappa} \rangle & \quad \mapsto \quad \langle e_0, \rho, \hat{\sigma} \cup [a \mapsto \hat{\kappa}], (e_1, \rho), a \rangle \\
\langle v, \hat{\sigma}, (e, \rho), a \rangle & \quad \mapsto \quad \langle e, \rho, \hat{\sigma}, v, a \rangle \\
\langle v, \hat{\sigma}, (\lambda x.e, \rho), a \rangle & \quad \mapsto \quad \langle e, \rho[x \mapsto a'], \hat{\sigma} \cup [a' \mapsto v], \hat{\kappa} \rangle \\
& \quad \text{if } \hat{\kappa} \in \hat{\sigma}(a)
\end{align*} \]
\[ \langle x, \rho, \hat{\sigma}, \hat{\kappa} \rangle \quad \overset{\text{if } v \in \hat{\sigma}(\rho(x))}{\longrightarrow} \quad \langle v, \rho, \hat{\sigma}, \hat{\kappa} \rangle \]
\[ \langle e_0 \ e_1, \rho, \hat{\sigma}, \hat{\kappa} \rangle \quad \overset{}{\longrightarrow} \quad \langle e_0, \rho, \hat{\sigma} \sqcup \left[ \mathsf{a} \mapsto \hat{\kappa} \right], (e_1, \rho), \mathsf{a} \rangle \]
\[ \langle v, \hat{\sigma}, (e, \rho), \mathsf{a} \rangle \quad \overset{}{\longrightarrow} \quad \langle e, \rho, \hat{\sigma}, v, \mathsf{a} \rangle \]
\[ \langle v, \hat{\sigma}, (\lambda x. e, \rho), \mathsf{a} \rangle \quad \overset{}{\longrightarrow} \quad \langle e, \rho \left[ x \mapsto a' \right], \hat{\sigma} \sqcup \left[ a' \mapsto v \right], \hat{\kappa} \rangle \quad \text{if } \hat{\kappa} \in \hat{\sigma}(\mathsf{a}) \]
\[ \langle x, \rho, \hat{\sigma}, \kappa \rangle \quad \overset{\text{if } v \in \hat{\sigma}(\rho(x))}{\longrightarrow} \quad \langle v, \rho, \hat{\sigma}, \kappa \rangle \]
\[ \langle e_0 \ e_1, \rho, \hat{\sigma}, \kappa \rangle \quad \overset{}{\longrightarrow} \quad \langle e_0, \rho, \hat{\sigma}, (e_1, \rho) \cdot \kappa \rangle \]
\[ \langle v, \hat{\sigma}, (e, \rho) \cdot \kappa \rangle \quad \overset{}{\longrightarrow} \quad \langle e, \rho, \hat{\sigma}, v \cdot \kappa \rangle \]
\[ \langle v, \hat{\sigma}, (\lambda x. e, \rho) \cdot \kappa \rangle \quad \overset{}{\longrightarrow} \quad \langle e, \rho \left[ x \mapsto a \right], \hat{\sigma} \sqcup \left[ a \mapsto v \right], \kappa \rangle \]
f(x);
f(y);

function f(z) {
  ...
  return;
}

f(x);
f(y);
f(x);

function f(z) {
  ...
  return;
}

f(y);
let's Use P.D.A. & G.C.

69
613
47
8
92
472
22

(3) with GC only: 105 states

52
58
566
136
142
97
375
383
504
61

order functions, but for short durations. The abstract-GC-only is

directly into the abstract semantics, which are in turn, phrased in
enabling pushdown analysis without continuation passing style.

the frames; it only needs the
confused by non-tail-recursive loop structure. With both techniques

(2) with pushdown only: 139 states

There are three strong secondary motiva-
tions for this work: (1) bringing context-sensitivity to pushdown

C(ontrol), E(nvironment) and S(tore) portions of the machine while
retains just enough restrictions to compute reachable control states,

pushdown systems—which

pushdown systems—which

pushdown systems—which

pushdown systems—which

pushdown systems—which

pushdown systems—which

pushdown systems—which
We generated an abstract transition graph for the same infinite-state abstract interpretation, constructed by bounding the abstract GC, the abstract transition graph shrinks by an order of magnitude. In CFA2, monovariant (OCFA-like) context-sensitivity is etched directly into the abstract semantics, which are in turn, phrased in terms of abstract garbage collection; (2) with only abstract garbage collection, the abstract transition graph shrinks by an order of magnitude.

We first review preliminaries to set a consistent feel for terminology. For the purposes of this work: (1) bringing context-sensitivity to pushdown systems, and recast abstract garbage collection.
\[ v ::= (\lambda x.e, \rho), \sigma, \kappa \]

\[ \kappa ::= \text{nil} \mid ([e, \rho] : : \kappa) \mid (v[]) : : \kappa \]

\[ \hat{\kappa} ::= \text{nil} \mid ([e, \rho]), a \mid (v[]), a \]

\[ e, \rho, \hat{\sigma}, \kappa \]

\[ e, \rho, \hat{\sigma}, \kappa \]
\[
\begin{align*}
\hat{\sigma} &::= (\lambda x.e, \rho) \\
\kappa &::= \text{nil} | ([ ] e, \rho): : \kappa | (v[]): : \kappa \\
\hat{\kappa} &::= \text{nil} | ([ ] e, \rho), a | (v[]), a \\
\end{align*}
\]
\[ e, \rho, \hat{\sigma} \] 

Finite control state
$v ::= (\lambda x. e, \rho)$

$\kappa ::= \text{nil} | ([ ] e, \rho) : : \kappa | ([ ] v) : : \kappa$

$\hat{\kappa} ::= \text{nil} | ([ ] e, \rho), a | ([ ] v), a$

$e, \rho, \sigma, \hat{\kappa}$

Stack, with finite alphabet

Finite control state
Stack, with finite alphabet

Finite control state

PDA

$e, \rho, \hat{\sigma}$

$e, \rho, \hat{\sigma}$
Part III: Results
<table>
<thead>
<tr>
<th>AndorsTrail</th>
<th>SplitTimer</th>
<th>SMSBackup</th>
</tr>
</thead>
<tbody>
<tr>
<td>AndroidGame</td>
<td>SuperNote</td>
<td>SMSBlocker</td>
</tr>
<tr>
<td>AndroidPrivacyGuard_E</td>
<td>SuperSoduko</td>
<td>SMSPopup</td>
</tr>
<tr>
<td>Butane</td>
<td>SysMon</td>
<td>SysWatcherA</td>
</tr>
<tr>
<td>CalcA</td>
<td>SysWatcherB</td>
<td>SourceViewer</td>
</tr>
<tr>
<td>CalcB</td>
<td>TextSecure</td>
<td>UltraCoolMap</td>
</tr>
<tr>
<td>ConnectBot</td>
<td>TodoList</td>
<td>YARR</td>
</tr>
<tr>
<td>CountdownTimer</td>
<td>Word Helper</td>
<td>AndroidsFortune</td>
</tr>
<tr>
<td>FunDraw</td>
<td>AndBible</td>
<td>CalcC</td>
</tr>
<tr>
<td>MorseCode</td>
<td>AndroidPrivacyGuard_M</td>
<td>CalcE</td>
</tr>
<tr>
<td>MyDrawA</td>
<td>BatteryIndicator</td>
<td>ColorMatcher</td>
</tr>
<tr>
<td>MyDrawC</td>
<td>CalcF</td>
<td>FullControl</td>
</tr>
<tr>
<td>NewsCollator</td>
<td>MediaFun</td>
<td>KitteyKittey</td>
</tr>
<tr>
<td>PasswordSaver</td>
<td>MyDrawD</td>
<td>Orienteering2</td>
</tr>
<tr>
<td>PersistantAssistant</td>
<td>OpenGPSTracker</td>
<td>Sanity</td>
</tr>
<tr>
<td>SmartWebCam</td>
<td>Orienteering1</td>
<td>TomDroid</td>
</tr>
<tr>
<td>SMSReminder</td>
<td>PicViewer</td>
<td>WiFinder</td>
</tr>
<tr>
<td>SourceViewer</td>
<td>Collaboration ShareLoc</td>
<td>DARPA</td>
</tr>
</tbody>
</table>
Improving Exception-flow analysis
a.foo()

try {
  b.foo()
} catch {
  ...
}

method foo() {
  ...
  return ...
  ...
  throw ...
}

...
try {
    b.foo()
} catch {
    ...
}
try {
    b.foo()
} catch {
    ...
    ...
    ...
    ... throw ...
    ...
    ...
}
a.foo()

try {
    b.foo()
} catch {
    ...
}

method foo() {
    ...
    return ...
    ...
    throw ...
}


try {
    b.foo()
} catch {
    ...
}
try {
    b.foo()
} catch {
    ...
    ...
    throw ...
    ...
}

method foo() {
    ...
    return ...
    ...
}
a.foo()

try {
    b.foo()
} catch {
    ...
}

The navigation software on this device is designed to provide route suggestions and does not replace the driver attentiveness or the requirement to adhere to regulations, and posted road signs and directions should always take precedence.

⚠️ Sorry!
The application NAVIGON (process com.navigon.navigator_checkout_eu40) has stopped unexpectedly. Please try again.

Force close

Do not operate this unit while driving.
try {
    b.foo()
} catch {
    method foo() {
        ... return ...
        ... throw ...
    }
}
a.foo()

try {
    b.foo()
} catch {
    method foo() {
        ... return ...
        ... throw ...
    }
}

try {
    b.foo()
} catch {
}
a.foo()

try {
    b.foo()
} catch { }

method foo() {
    ... return ...
    ... throw ...
}
<table>
<thead>
<tr>
<th>Program</th>
<th>Variable points-to</th>
<th>Throw-Catch edges</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>antlr</td>
<td>614</td>
<td>2277</td>
<td>&gt;4 hours</td>
</tr>
<tr>
<td>35KLOC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lusearch</td>
<td>348</td>
<td>2378</td>
<td>46 minutes</td>
</tr>
<tr>
<td>87KLOC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pmd</td>
<td>343</td>
<td>2284</td>
<td>56 minutes</td>
</tr>
<tr>
<td>55KLOC</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bravenboer & Smaragdakis, ISSTA’09
<table>
<thead>
<tr>
<th>Program</th>
<th>Variable points-to</th>
<th>Throw-Catch edges</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>antlr</td>
<td>614</td>
<td>2277</td>
<td>&gt;4 hours</td>
</tr>
<tr>
<td>35KLOC</td>
<td>2</td>
<td>65</td>
<td>1.1 hours</td>
</tr>
<tr>
<td>lusearch</td>
<td>348</td>
<td>2378</td>
<td>46 minutes</td>
</tr>
<tr>
<td>87KLOC</td>
<td>2</td>
<td>59</td>
<td>46 minutes</td>
</tr>
<tr>
<td>pmd</td>
<td>343</td>
<td>2284</td>
<td>56 minutes</td>
</tr>
<tr>
<td>55KLOC</td>
<td>2</td>
<td>38</td>
<td>22 minutes</td>
</tr>
</tbody>
</table>

Pushdown Exception Flow Analysis
Run-time Techniques at Analysis-time
Abstract Models of Memory Management

Greg Morrisett
Carnegie Mellon
jmorris@cs.cmu.edu

Matthias Felleisen
Rice University
matthias@cs.rice.edu

Robert Harper
Carnegie Mellon
rwh@cs.cmu.edu

Abstract

Most specifications of garbage collectors concentrate on the low-level algorithmic details of how to find and preserve accessible objects. Often, they focus on bit-level manipulations such as "scanning stack frames," "marking objects," "tagging data," etc. While these details are important in some contexts, they often obscure the more fundamental aspects of memory management: what objects are garbage and why.

We develop a series of calculi that are just low-level enough that we can express allocation and garbage collection, yet are sufficiently abstract that we may formally prove the correctness of various memory management strategies. By making the heap of a program syntactically apparent, we can specify memory actions as rewriting rules that allocate values on the heap and automatically dereference pointers to such objects when needed. This formulation permits the specification of garbage collection as a relation that removes portions of the heap without affecting the outcome of the evaluation.

Our high-level approach allows us to specify in a compact manner a wide variety of memory management techniques, including standard trace-based garbage collection (i.e., the family of copying and mark/sweep collection algorithms), generational collection, and type-based, tag-free collection. Furthermore, since the definition of garbage is based on the semantics of the underlying language instead of the conservative approximation of inaccessibility, we are able to specify and prove that type inference can be used to collect objects that are accessible but never used.

1 Memory Safety

Advanced programming languages manage memory allocation and deallocation automatically. Automatic memory managers, or garbage collectors, significantly facilitate the programming process because programmers can rely on the language implementation for the delicate tasks of finding and freeing unused objects. Indeed, the presence of a garbage collector ensures memory safety in the same way that a type system guarantees type safety: no program written in an advanced programming language will crash due to dangling pointer problems while allocation, access, and deallocation are transparent. However, in contrast to type systems, memory management strategies and particularly garbage collectors rarely come with a compact formulation and a formal proof of soundness. Since garbage collectors work on the machine representations of abstract values, the very idea of providing a proof of memory safety sounds unrealistic given the lack of simple models of memory operations.

The recently developed syntactic approaches to the specification of language semantics by Felleisen and Heib [11] and Mason and Talcott [18, 19] are the first execution models that are intensional enough to permit the specification of memory management actions and yet are sufficiently abstract to permit compact proofs of important properties. Starting from the $\lambda$–$\Sigma$ calculus of Felleisen and Heib, we design compact specifications of a number of memory management ideas and prove several correctness theorems.

The basic idea underlying the development of our garbage collection calculi is the representation of a program’s run-time memory as a global series of syntactic declarations. The program evaluation rules allocate large objects in the global declaration, which represents the heap, and automatically dereference pointers to such objects when needed. As a result, garbage collection can be specified as any relation that removes portions of the current heap without affecting the result of a program’s execution.

In Section 2, we present a small functional programming language, Age, with a rewriting semantics that makes allocation explicit. We define a semantic notion of garbage collection for Age and prove that there is no optimal collection strategy that is computable. In Section 3, we specify the "free-variable" garbage collection rule which models trace-based collectors including mark/sweep and copying collectors. We prove that the free-variable rule is correct and provide two "implementations" at the syntactic level: the first corresponds to a copying collector, the second to a generational one.

In Section 4, we formalize so-called "tag-free" collection algorithms for explicitly-typed, monomorphic languages such as Pascal and Algol [7, 29, 8]. We show how to recover...
Improved precision and efficiency via abstract GC
function twice(f, x) { return f(f(x)); };

twice(sqr, 4);  twice(dbl, 5);
function twice(f, x) { return f(f(x)); };

twice(sqr, 4);  
twice(dbl, 5);

twice(sqr, 4);
		\{sqr(sqr(4))\}

twice(dbl, 5);
	\{sqr\}  \{4\}
function twice(f, x) { return f(f(x)); }

twice(sqr, 4);
{ sqr(sqr(4)) }  twice(dbl, 5);
function twice(f, x) { return f(f(x)); };

twice(sqr, 4);
{ sqr(sqr(4)) }

twice(dbl, 5);
function twice(f, x) { return f(f(x)); }; 

{ 
  sqr(sqr(4)) 
}

twice(sqr, 4); 

twice(dbl, 5); 

{ 
  {5} 
} 

{ 
  {dbl1} 
}
function twice(f,x) { return f(f(x)); }; 

twice(sqr,4);          twice(dbl,5);

{ sqr(sqr(4)) }        { dbl(dbl(5)) }
e, \hat{\rho}, \hat{\sigma}, \hat{\kappa}
$e, \hat{\rho}, \hat{\sigma}, \hat{\kappa}$
Analysis time

- Normal
- With abstract garbage collection
Analysis time

- earley
- fringe
- stream
- lattice
- nboyer
- perm
- doubler
- sboyer

Normal

With abstract garbage collection
Normal

With abstract garbage collection

Analysis time
Problem: Needless non-determinism
Problem: needless non-determinism

Solution: lazy non-determinism
Problem: Needless non-determinism

Solution: Lazy non-determinism
**Problem: Long Corridors**
**Problem:** Long Corridors

**Solution:** Abstract Compilation
Problem: Long Corridors

Solution: Abstract Compilation
Factor improvement of Peak Memory usage
Factor improvement of speed of transitions
Factor improvement of Analysis Time
Behavioral Software Contract Verification
/**
 * @param left a sorted list of elements
 * @param right a sorted list of elements
 * @return the contents of the two lists, merged, sorted
 */
 List merge(List left, List right);

@Requires(
 "Collections.isSorted(left)",
 "Collections.isSorted(right)"
)
@Ensures(
 "Collections.containsSame(result, Lists.concatenate(left, right))",
 "Collections.isSorted(result)"
)
 List merge(List left, List right);
/**
 * @param left a sorted list of elements
 * @param right a sorted list of elements
 * @return the contents of the two lists, merged, sorted
 */

List merge(List left, List right);

@Requires({
"Collections.isSorted(left)",
"Collections.isSorted(right)"
})
@Ensures({
"Collections.containsSame(result, Lists.concatenate(left, right))",
"Collections.isSorted(result)"
})
List merge(List left, List right);

@Requires({
"Collections.isSorted(left)",
"Collections.isSorted(right)"
})
@Ensures({
"Collections.containsSame(result, Lists.concatenate(left, right))",
"Collections.isSorted(result)"
})
List merge(List left, List right);

@Requires({
"Collections.isSorted(left)",
"Collections.isSorted(right)"
})
@Ensures({
"Collections.containsSame(result, Lists.concatenate(left, right))",
"Collections.isSorted(result)"
})
List merge(List left, List right);
/**
 * @param left a sorted list of elements
 * @param right a sorted list of elements
 * @return the contents of the two lists, merged, sorted
 */
List merge(List left, List right);
@Requires({
"Collections.isSorted(left)",
"Collections.isSorted(right)"
})
@Ensures({
"Collections.containsSame(result, Lists.concatenate(left, right))",
"Collections.isSorted(result)"
})
List merge(List left, List right);

@Requires({
"Collections.isSorted(left)",
"Collections.isSorted(right)"
})
@Ensures({
"Collections.containsSame(result, Lists.concatenate(left, right))",
"Collections.isSorted(result)"
})
List merge(List left, List right);
Contracts for Higher-Order Functions

Robert Bruce Findler 1  Matthias Felleisen
Northeastern University
College of Computer Science
Boston, Massachusetts 02115, USA

Abstract

Assertions play an important role in the construction of robust software. Their use in programming languages dates back to the 1970s. Eiffel, an object-oriented programming language, wholeheartedly adopted assertions and developed the “Design by Contract” philosophy. Indeed, the entire object-oriented community recognizes the value of assertions-based contracts on methods.

In contrast, languages with higher-order functions do not support assertion-based contracts. Because predicates on functions are, in general, undecidable, specifying such predicates appears to be meaningless. Instead, the functional languages community developed type systems that statically approximate interesting predicates.

In this paper, we show how to support higher-order function contracts in a theoretically well-founded and practically viable manner. Specifically, we introduce \( \lambda^{\text{C}} \), a typed lambda calculus with assertions for higher-order functions. The calculus models the assertion monitoring system that we employ in DrScheme. We establish basic properties of the model (type soundness, etc.) and illustrate the usefulness of contract checking with examples from DrScheme’s code base.

We believe that the development of an assertion system for higher-order functions serves two purposes. On one hand, the system has strong practical potential because existing type systems simply cannot express many assertions that programmers would like to state. On the other hand, an inspection of a large base of invariants may provide inspiration for the direction of practical future type system research.

Categories & Subject Descriptors: D.3.3, D.2.1; General Terms: Design, Languages, Reliability; Keywords: Contracts, Higher-order Functions, Behavioral Specifications, Polymorphic Typing, Software Reliability

1 Introduction

Dynamically enforced pre- and post-condition contracts have been widely used in procedural and object-oriented languages [11, 14, 17, 20, 21, 22, 25, 31]. As Rosenblum [27] has shown, for example, these contracts have great practical value in improving the robustness of systems in procedural languages. Eiffel [22] even developed an entire philosophy of system design based on contracts (“Design by Contract”). Although Java [12] does not support contracts, it is one of the most requested extensions. 3

With one exception, higher-order languages have mostly ignored assertion-style contracts. The exception is Bigloo Scheme [28], where programmers can write down first-order, type-like constraints on procedures. These constraints are used to generate more efficient code when the compiler can prove they are correct and are turned into runtime checks when the compiler cannot prove them correct.

First-order procedural contracts have a simple interpretation. Consider this contract, written in an ML-like syntax:

\[
\text{val rec } f : \text{int} \rightarrow \text{int} = \text{rec } x. f \cdot \text{int} > 9 \rightarrow \text{int}(0.99) \\
\text{val rec } f = x. x.
\]

It states that the argument to \( f \) must be an int greater than 9 and that \( f \) produces an int between 0 and 99. To enforce this contract, a contract compiler inserts code to check that \( x \) is in the proper range when \( f \) is called and that \( f \)'s result is in the proper range when \( f \) returns. If \( x \) is not in the proper range, \( f \)'s caller is blamed for a contractual violation. Symmetrically, if \( f \)'s result is not in the proper range, the blame falls on \( f \) itself. In this world, detecting contractual violations and assigning blame merely means checking appropriate predicates at well-defined points in the program’s evaluation.

This simple mechanism for checking contracts does not generalize to languages with higher-order functions. Consider this contract:

\[
\text{val rec } g : \text{int} \rightarrow \text{int}(0.99) = \text{int}(0.99) \\
\text{val rec } g = x. x. \\
\]

The contract’s domain states that \( g \) accepts \text{int} \rightarrow \text{int} functions and must apply them to \text{int}s larger than 9. In turn, these functions must produce \text{int}s between 0 and 99. The contract’s range obliges \( g \) to produce \text{int}s between 0 and 99.

Contracts for Higher-Order Functions

Abstract

In this paper, we show how to support higher-order function contracts in a purely functional setting. We develop a calculus for first-order procedural contracts that is parameterized by a type system. We then extend this framework to handle higher-order contracts via a novel approach to contract monitoring. We illustrate the usefulness of contract checking with examples from the DrScheme code base.

We believe that the development of an assertion system for higher-order functional languages is essential for the future of software development. The functional languages community has adopted assertions and developed the "Design by Contract" philosophy. Indeed, the entire object-oriented community recognizes the importance of assertions in the construction of robust software.

In contrast, languages with higher-order functions do not support assertions very well. In general, undecidable, specifying such predicates appears to be meaningless. Instead, the functional languages community developed type systems that statically approximate interesting predicates. With one exception, higher-order languages have mostly ignored this one of the most requested extensions.

First-order procedural contracts have a simple interpretation. Contrary to what one might expect, they are expressive enough to establish basic properties of the model (type soundness, etc.), and membership of a type system serves two purposes. On one hand, the system has established basic properties of the model (type soundness, etc.) and membership of a type system serves two purposes. On one hand, the system has established basic properties of the model (type soundness, etc.) and membership of a type system.

On the other hand, an inspection of a large base of invariants may not express many assertions that programmers would like to state. Instead, the functional languages community developed type systems that statically approximate interesting predicates. With one exception, higher-order languages have mostly ignored this one of the most requested extensions.

We introduce \( \lambda \text{CON} \), a typed lambda calculus with contracts. It states that the argument to \( f \) must be an integer between \( 0 \) and \( 99 \) for \( f \) to produce its result. If \( x \) is not in the proper range, the blame falls on \( f \) itself. In this world, detecting contractual violations and assigning blame merely means checking constraints on procedures. These constraints are used to generate more appropriate predicates at well-defined points in the program's evaluation.

The contract's domain states that \( f(\cdot) \) must apply them to \( n \) and \( m \) when \( n \) is less than \( m \) and \( m \) is less than \( \lambda \). To enforce this contract, a \( n \) is called and that \( f \) must be an integer and \( g \) accepts \( m \) and \( \lambda \) to \( n \) and \( m \). In turn, these functions must produce an integer and \( \lambda \) when \( \lambda \) is not in the proper range.

First-order procedural contracts have a simple interpretation. Contractual violations and assigning blame merely means checking constraints on procedures. These constraints are used to generate more appropriate predicates at well-defined points in the program's evaluation. Dynamically enforced pre- and post-condition contracts have been widely used in procedural and object-oriented languages [11, 14, 17, 20, 21, 22, 25, 31]. As Rosenblum [27] has shown, for example, these contracts have great practical value in improving the robustness of systems in procedural languages. Eiffel [22] even developed an entire philosophy of system design based on contracts ("Design by Contract"). Although Java [12] does not support contracts, it is adopted assertions and developed the "Design by Contract" philosophy. Indeed, the entire object-oriented community recognizes the importance of assertions in the construction of robust software.

Keywords: Contracts, Higher-order Functions, Lambda Calculus, Type System, Assertion Monitoring

Robert Bruce Findler
Boston, Massachusetts 02115, USA
Matthias Felleisen
College of Computer Science
Northeastern University

1 Introduction

In general, undecidable, specifying such predicates appears to be meaningless. Instead, the functional languages community developed type systems that statically approximate interesting predicates. With one exception, higher-order languages have mostly ignored this one of the most requested extensions.

Assertions play an important role in the construction of robust software. In general, undecidable, specifying such predicates appears to be meaningless. Instead, the functional languages community developed type systems that statically approximate interesting predicates. With one exception, higher-order languages have mostly ignored this one of the most requested extensions.

17, 20, 21, 22, 25, 31. As Rosenblum [27] has shown, for example, these contracts have great practical value in improving the robustness of systems in procedural languages. Eiffel [22] even developed an entire philosophy of system design based on contracts ("Design by Contract"). Although Java [12] does not support contracts, it is adopted assertions and developed the "Design by Contract" philosophy. Indeed, the entire object-oriented community recognizes the importance of assertions in the construction of robust software.

Abstract

In this paper, we show how to support higher-order function contracts in a purely functional setting. We develop a calculus for first-order procedural contracts that is parameterized by a type system. We then extend this framework to handle higher-order contracts via a novel approach to contract monitoring. We illustrate the usefulness of contract checking with examples from the DrScheme code base.

We believe that the development of an assertion system for higher-order functional languages is essential for the future of software development. The functional languages community has adopted assertions and developed the "Design by Contract" philosophy. Indeed, the entire object-oriented community recognizes the importance of assertions in the construction of robust software.

In contrast, languages with higher-order functions do not support assertions very well. In general, undecidable, specifying such predicates appears to be meaningless. Instead, the functional languages community developed type systems that statically approximate interesting predicates. With one exception, higher-order languages have mostly ignored this one of the most requested extensions.

First-order procedural contracts have a simple interpretation. Contrary to what one might expect, they are expressive enough to establish basic properties of the model (type soundness, etc.), and membership of a type system serves two purposes. On one hand, the system has established basic properties of the model (type soundness, etc.) and membership of a type system serves two purposes. On one hand, the system has established basic properties of the model (type soundness, etc.) and membership of a type system.
### Semantics for Symbolic PCF with Contracts

\[ E \mapsto E' \]

- if \( V \, E_1 \, E_2 \mapsto E_1 \) if \( \delta(\text{false}?, \, V) \ni \text{ff} \)
- if \( V \, E_1 \, E_2 \mapsto E_2 \) if \( \delta(\text{false}?, \, V) \ni \text{tt} \)
- \( (\lambda X : T. \, E) \, V \mapsto [V/X] \, E \)
- \( \mu X : T. \, E \mapsto [\mu X : T. \, E/X] \, E \)
- \( O(\vec{V}) \mapsto A \) if \( \delta(O, \vec{V}) \ni A \)
- \( (\bullet^{T \rightarrow T'}/C) \, V \mapsto \bullet^{T'}/\{[V/X] \, C_2 \mid C_1 \mapsto \lambda X : T. \, C_2 \in C\} \)
- \( (\bullet^{T \rightarrow T'}/C) \, V \mapsto \text{havoc}_T \, V \)

[OOPSLA’12]
Semantics for Symbolic PCF with Contracts

\[
\begin{align*}
\# V : E_1 \rightarrow E_2 : \# (\# \# \text{false}, V) \rightarrow \# & \\
\# V : E_1 \rightarrow E_2 : \# (\# \# \text{false}, V) \rightarrow \# \text{tt} & \\
(\lambda X : T . E) : V \rightarrow [V/X]E & \\
\mu X : T . E \rightarrow [\mu X : T . E] & \\
O(V) \rightarrow A \# (O, \# \# V) \rightarrow A & \\
(\varphi^{T \rightarrow T} / C) : V \rightarrow \varphi^{T / [V/X]C} & \\
(\varphi^{T \rightarrow T} / C) : V \rightarrow \text{havoc}_{C} & \\
\end{align*}
\]
Abstract (or synonymously: symbolic) values to take the values of CPCF as “pre”-values to the values of the same program after replacing the contractum with: (C \vdash A). This enables multiple results when a symbolic value does not produce an answer. If the value is returned. Otherwise, a contract error is signaled with appropriate blame.

The high-level goal of the following semantics is to establish basic properties of the model (type soundness, etc.) and developed type systems that statically approximate interesting predicates on functions, e.g., from languages with higher-order functions do not support value of assertion-based contracts on methods. In contrast, languages with higher-order functions do not support assertions and developed the “Design by Contract” philosophy. Assertions play an important role in the construction of robust software systems.

We believe that the development of an assertion system for higher-order functions allows for the establishment of basic properties of the model (type soundness, etc.) and the development of type systems that statically approximate interesting predicates on functions, e.g., from languages with higher-order functions. In contrast, languages with higher-order functions do not support assertions and developed the “Design by Contract” philosophy. Assertions play an important role in the construction of robust software systems.

We believe that the development of an assertion system for higher-order functions allows for the establishment of basic properties of the model (type soundness, etc.) and the development of type systems that statically approximate interesting predicates on functions, e.g., from languages with higher-order functions. In contrast, languages with higher-order functions do not support assertions and developed the “Design by Contract” philosophy. Assertions play an important role in the construction of robust software systems.
Conclusion & Perspective
To trust software, we must predict its (mis)behavior.
AnaDroid
AnaDroid
AnaDroid

Factor Improvement of Analysis Time

<table>
<thead>
<tr>
<th>Program</th>
<th>Variable points-to</th>
<th>Throw-Catch edges</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>antlr</td>
<td>(40503, 614)</td>
<td>2277</td>
<td>&gt;4 hours</td>
</tr>
<tr>
<td>35KLOC</td>
<td>(681, 2)</td>
<td>65</td>
<td>1.1 hours</td>
</tr>
<tr>
<td>lusearch</td>
<td>(22970, 348)</td>
<td>2378</td>
<td>46 minutes</td>
</tr>
<tr>
<td>87KLOC</td>
<td>(709, 2)</td>
<td>59</td>
<td>46 minutes</td>
</tr>
<tr>
<td>pmd</td>
<td>(25286, 343)</td>
<td>2284</td>
<td>56 minutes</td>
</tr>
<tr>
<td>55KLOC</td>
<td>(1017, 2)</td>
<td>38</td>
<td></td>
</tr>
</tbody>
</table>
Robust, Reliable Software and Trustworthy Systems
\[a[i]\]

\[i < a.\text{length}()\]

\[\neg (i < a.\text{length}())\]

\[\text{bad}\]
a[i]

i < a.length()

!(i < a.length())

v

bad
a[i] \quad \text{i < a.length()}

Z3

v
open(f);

---

close(f);
Temporal Higher-Order Contracts

Tim Disney
University of California, Santa Cruz

Cormac Flanagan
University of California, Santa Cruz

Jay McCarthy
Brigham Young University

Abstract
Behavioral contracts are embraced by software engineers because they document module interfaces, detect interface violations, and help identify faulty modules (packages, classes, functions, etc.). This paper extends prior higher-order contract systems to also express and enforce temporal properties, which are common in software systems with imperative state, but which are mostly left implicit or are at best informally specified. The paper presents both a programmatic contract API as well as a temporal contract language, and reports on experience and performance results from implementing these contracts in Racket.

Our development formalizes module behavior as a trace of events such as function calls and returns. Our contract system provides both non-instrumentable (where contracts cannot influence execution) and also a notion of completeness (where contracts can enforce any decidable, prefix-closed predicate on event traces).

Categories and Subject Descriptors D.3.1 [Formal Definitions and Theory]: Semantics; D.3.3 [Language Constructs and Features]: Contracts

General Terms Languages, Reliability, Security, Verification.

Keywords Higher-order Programming, Temporal Contracts

Temporal contracts are supported by existing contract systems.

1. The assert function takes two arguments, an array of positive integers and a comparison function cmp.

2. This standard, first-order precondition constrains how assert should be called, that is, what arguments are valid. These kinds of basic first-order contracts are supported by most contract systems, for example, Eiffel [40].

3. The assert function requires two arguments, both positive integers,

This higher-order precondition constrains how the assert module can call the function argument cmp, and so is a guarantee provided by assert rather than an obligation on the client. Higher-order contract systems [19, 15, 22, 24, 45] support such preconditions by wrapping the cmp argument to enforce this property dynamically.

The assert function is not re-entrant—it can only be called after all previous assert invocations have completed.

Unlike the previous contracts that constrain how functions may be called, this temporal contract constrains when assert can be called [12, 13]. This constraint implies that assert must be used

assert(ψ)
Robust, Reliable Software and Trustworthy Systems

Thank you
Robust, Reliable Software and Trustworthy Systems
Robust, Reliable Software and Trustworthy Systems

Thank you
Robust, Reliable Software and Trustworthy Systems

Thank you
Robust, Reliable Software and Trustworthy Systems
What about numbers, strings, arrays, etc.?

\[
\begin{array}{c}
  T \\
  \cdots \quad -1 \quad -2 \quad 0 \quad 1 \quad 2 \quad \cdots \\
  \cdots \\
  \cdots \\
  \bot
\end{array}
\]

\[
sqr(5) = T
\]
What about numbers, strings, arrays, etc.?

\[
\begin{array}{c}
\ldots \\
\ldots -1 -2 0 1 2 \ldots \\
\ldots \\
\bot
\end{array}
\]

\[
\begin{array}{c}
Neg \\
\text{sqr}(5) = \text{Pos}
\end{array}
\]

\[
\begin{array}{c}
\uparrow \\
\text{Pos} \\
\downarrow \\
\bot
\end{array}
\]

\[
\begin{array}{c}
\uparrow \\
\text{Neg} \\
\downarrow \\
\bot
\end{array}
\]