Safely Optimizing Casts between Pointers and Integers

Juneyoung Lee, Chung-Kil Hur, Ralf Jung, Zhengyang Liu, John Regehr, Nuno P. Lopes
Seoul National University, MPI-SWS, University of Utah, Microsoft Research

Pointers and Integers

<table>
<thead>
<tr>
<th>Assembly (x86-64, ARM, ..)</th>
<th>LLVM IR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pointer</td>
<td>[0, 2^{64}]</td>
</tr>
<tr>
<td>Integer</td>
<td>[0, 2^{64}]</td>
</tr>
</tbody>
</table>

Proposed Semantics

```cpp
ptrint (addr, prov) := addr
inttoptr (addr) := (addr, full)
```

How to Block ‘Guessed Access’

```cpp
char p = 0;
f(); *(char*) (0x100) = 10;
print(q[0]);
```

- Even if &p is 0x100, f() shouldn’t update p to 1
- We define that each allocation creates 2 blocks
1. p creates two blocks 0x100, 0x200
2. One of them (e.g. 0x100) is nondet. Invalidated
3. Now we start twin-execution by run the following code
4. Guessed access like f() raises UB in one of them

Performance Issue

- (char*)int p → p removes 13% of ptrtoint, 40% of inttoptr from SPEC2017rate+Test-suite
- Disabling it hurts performance

1. Make LLVM generate less casts
   - Source: pointer subtraction, load/store canon.
```
v = load 18* p
v2 = load 164* p
v = load 164* p
v2 = inttoptr v
```
- For pointer subtraction: introduce psub
- For load/store canon: use data type (future work)
- Removes 83% of ptrtoint, 89% of inttoptr

2. Conditionally perform cast elimination if sound
   - icmp (i2p (p2i p)), q → icmp p, q if prov(p)=prov(q)
   - ... (list available at https://github.com/aqjune/eurollvm19)

Our Suggestion

- Provenance makes integer optimizations hard to explain.
```
r = (i+j) - k ↔ r = i + (j - k)
```
- OOPSLA’18, Reconciling High-level Optimizations and Low-level Code in LLVM

Evalutation

- Implemented on LLVM 8.0
- SPEC2017rate: <0.2% slowdown
- LLVM test-suite: 0.1% avg slowdown
- https://github.com/aqjune/eurollvm19