

Safely Optimizing Casts between Pointers and Integers

EuroLLVM'19

Juneyoung Lee, Chung-Kil Hur, Ralf Jung, Seoul National University

Zhengyang Liu, John Regehr, University of Utah

Nuno P. Lopes, Microsoft Research

Pointers and Integers

	Assembly (x86-64, ARM, ..)	LLVM IR
Pointer	$[0, 2^{64})$	$[0, 2^{64}) + \textit{provenance}$
Integer	$[0, 2^{64})$	$[0, 2^{64}) + ?$

Casting

Proposed Semantics

```
ptrtoint (addr, prov) := addr
inttoptr (addr)      := (addr, full)
```

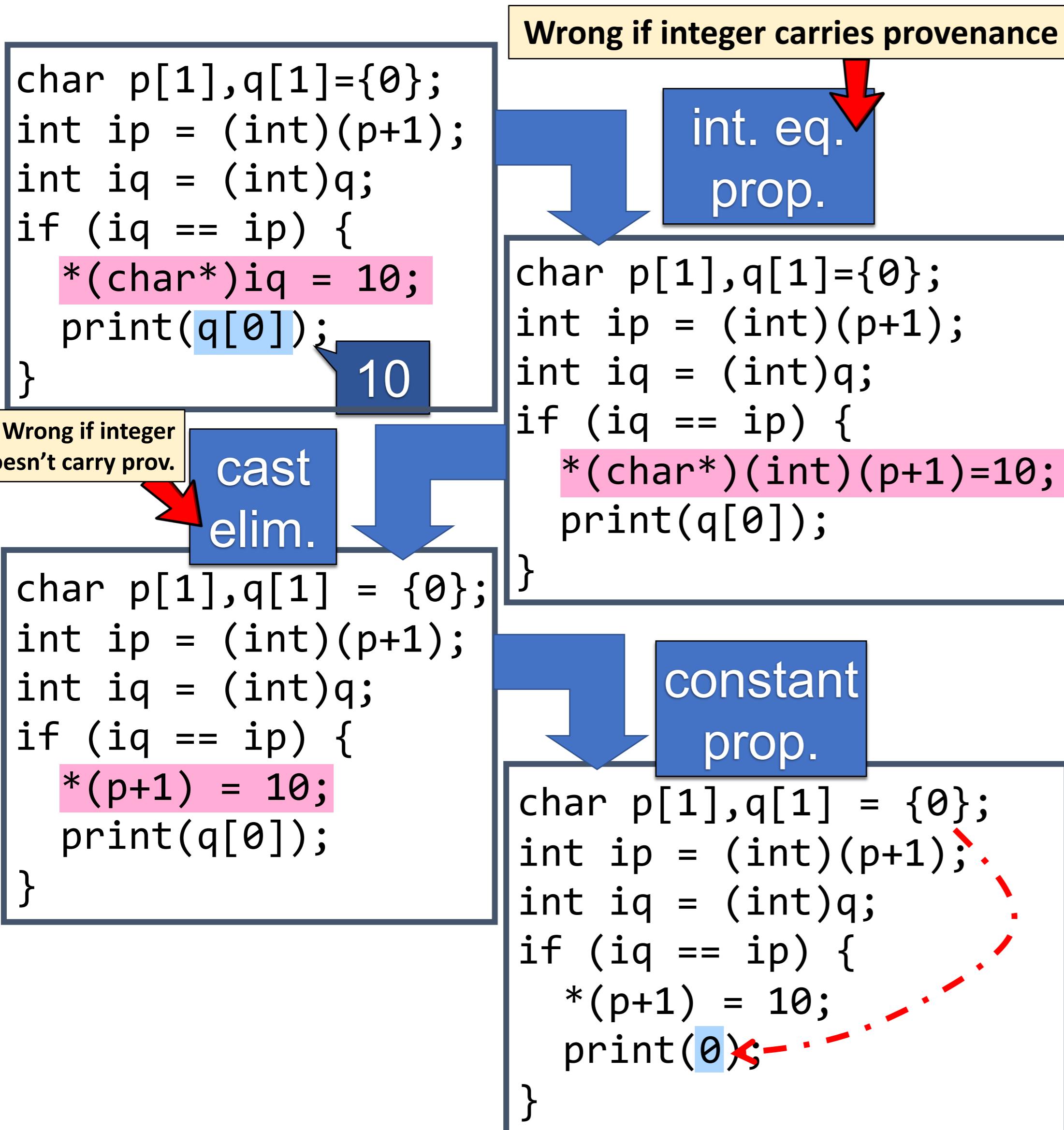
How to Block 'Guessed Access'

```
char p = 0;
f();
*(char*)(0x100)=1;
print(p);
```

```
char p = 0;
f();
print(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

Miscompilation w/ Int-Ptr Casting



Performance Issue

Problem

- $(char*)(int)p \rightarrow p$ removes 13% of ptrtoints, 40% of inttoptrs from SPEC2017rate+test-suite
- Disabling it hurts performance

Solution

1. Make LLVM generate less casts

- Source: pointer subtraction, load/store canon.

```
v = load i8** p
v2= load i8** p
```

→

```
v = load i64* p
v2= load i8** p
```

→

```
v = load i64* p
v2= inttoptr v
```

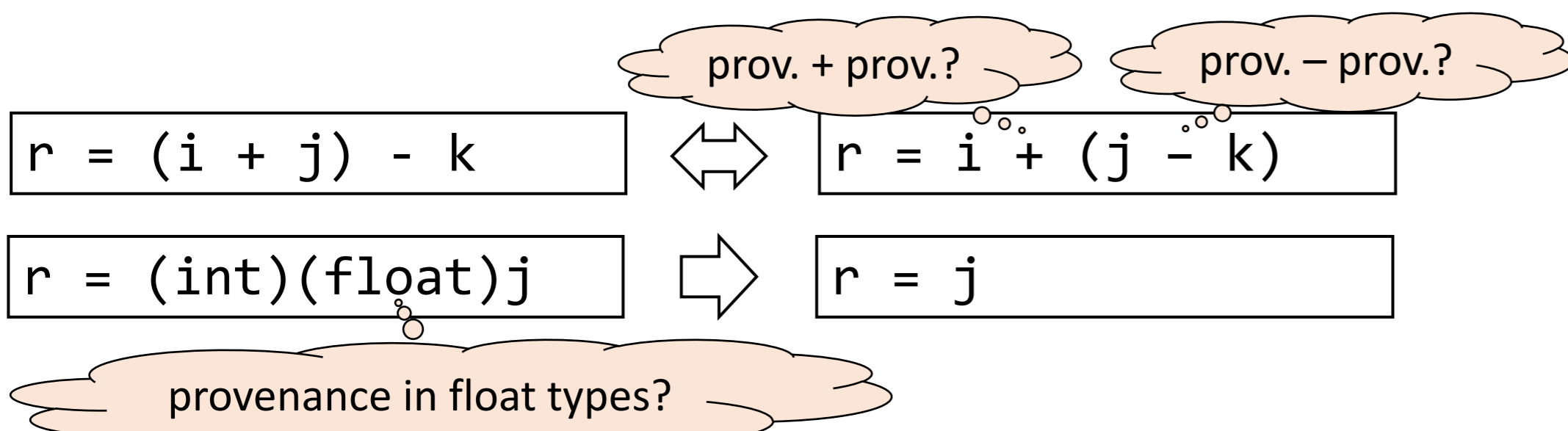
- For pointer subtraction: introduce psub
- For load/store canon: use data type (future work)
- Removes **83% of ptrtoints, 89% of inttoptrs**

2. Conditionally perform cast elimination if sound

- $icmp(i2p(p2i p)), q \rightarrow icmp p, q$ if $prov(p)=prov(q)$
- ... (list available at github.com/aqjune/eurollvm19)

Our Suggestion

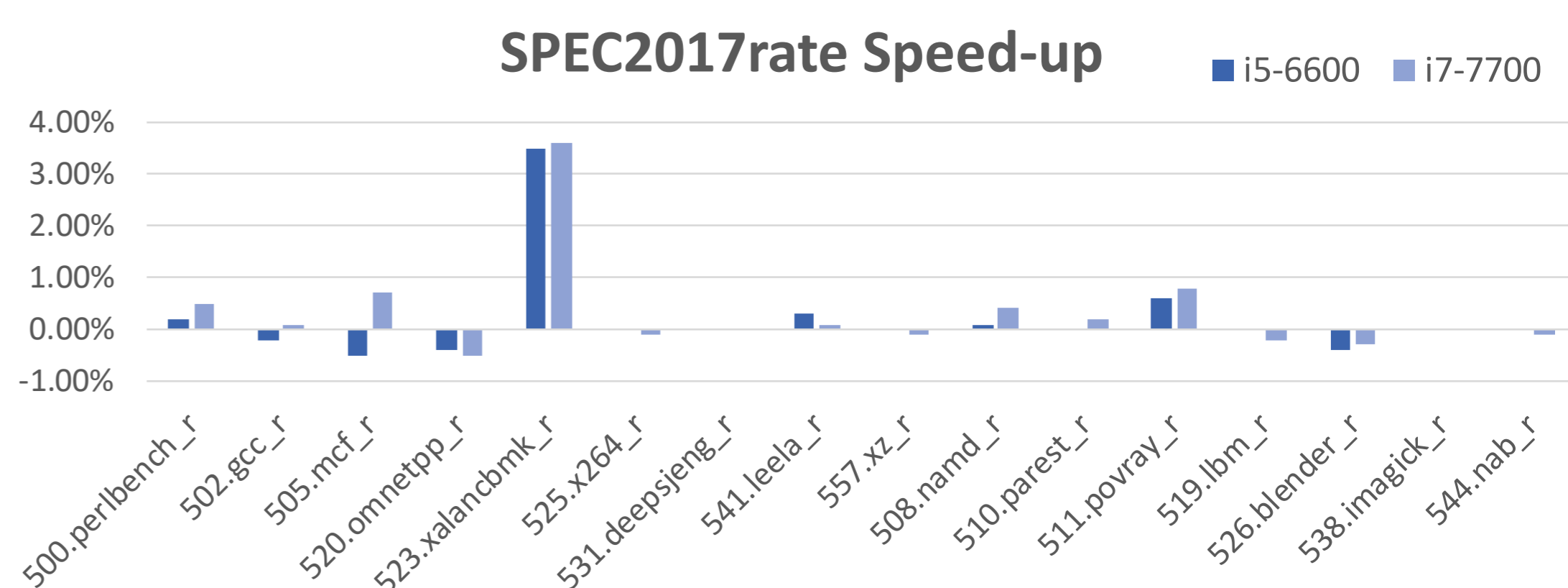
- Provenance makes integer optimizations hard to explain.



	Assembly (x86-64, ARM, ..)	LLVM IR
Pointer	$[0, 2^{64})$	$[0, 2^{64}) + \textit{provenance}$
Integer	$[0, 2^{64})$	$[0, 2^{64})$

- OOPSLA'18, Reconciling High-level Optimizations and Low-level Code in LLVM

Evaluation



Performance Change of SPEC2017rate

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