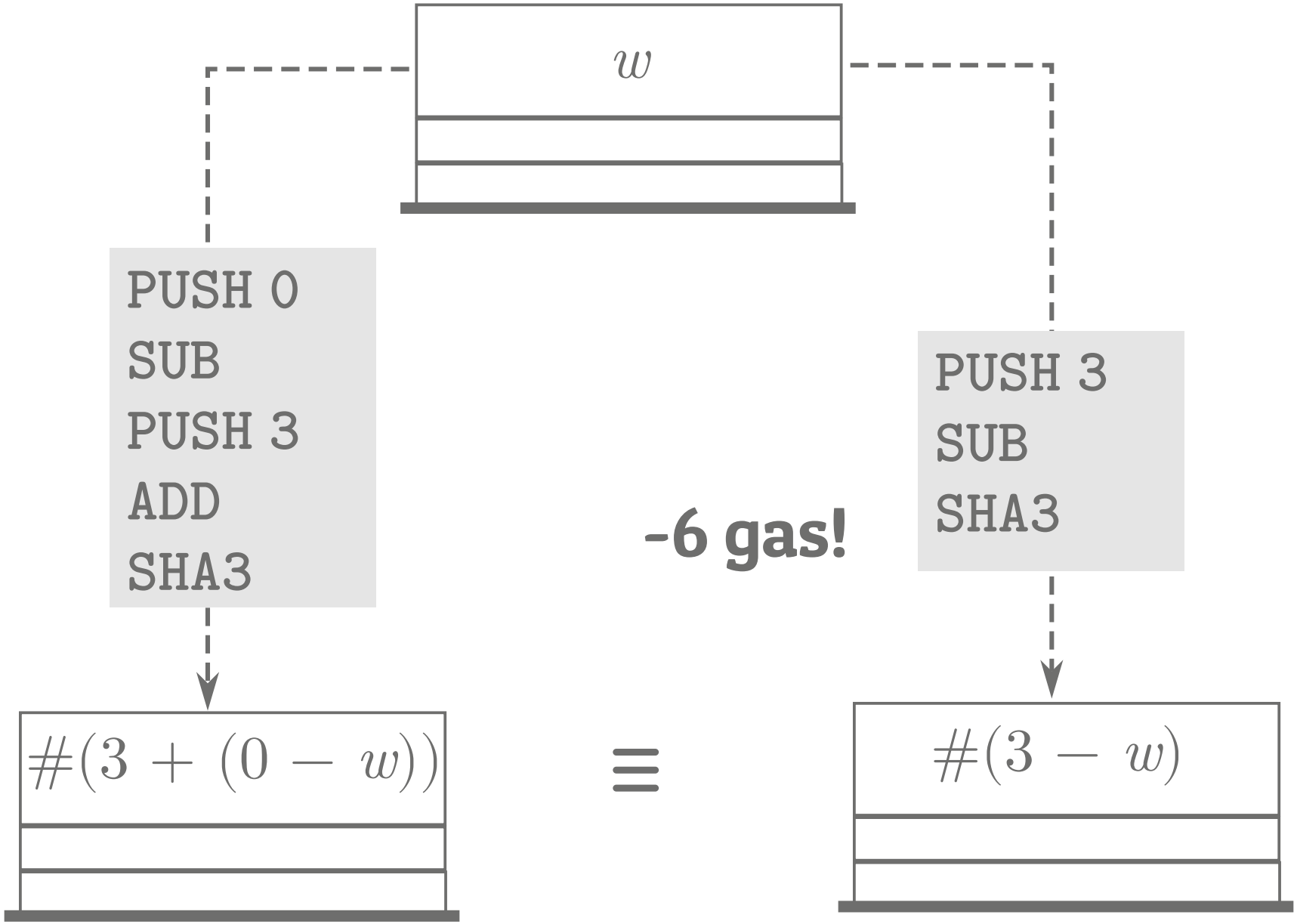
Populating the Peephole Optimizer of a Smart Contract Compiler

Maria A Schett & Julian Nagele

mail@ {maria-a-schett.net, jnagele.net}

FMBC @ CAV 2020

2020 | 07 | 20


w
PUSH 0
SUB
PUSH 3
ADD
SHA3
PUSH 3
SUB
SHA3
-6 gas!
$\#(3 + (0 - w))$
$\equiv$
$\#(3 - w)$

```
PUSH 0
SUB
PUSH 3
ADD
SHA3
```

$\equiv$

```
PUSH 3
SUB
SHA3
```

generate **peephole optimization rule**

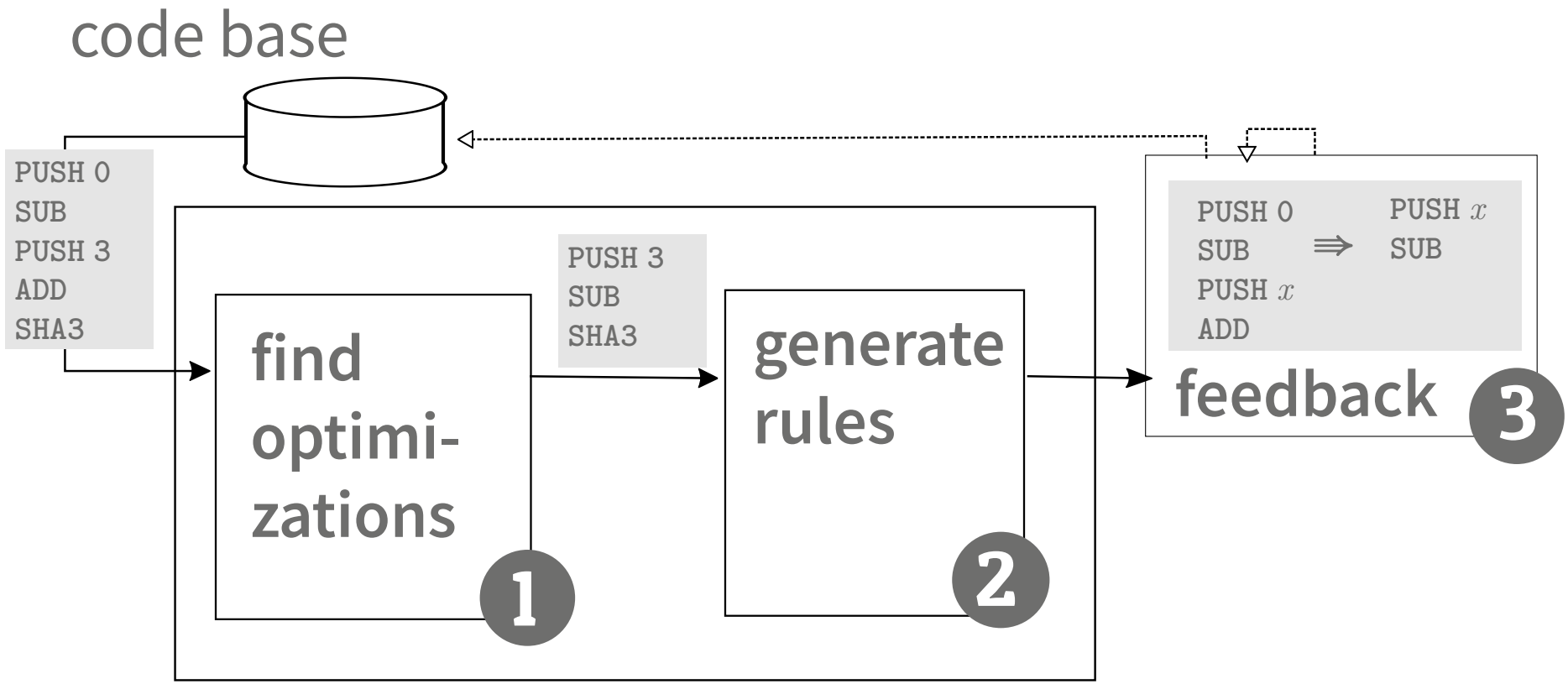
```
PUSH 0
SUB
PUSH x
ADD
```

$\Longrightarrow$

```
PUSH x
SUB
```

automatically!

How?



case study for **Ethereum bytecode**

Why?

for **1000** most-called contracts on **Ethereum**

We could save

56 500 $ of execution cost*

~4.5 % of space

* assuming avg. cost of 27.6 gwei, 200.2$/ETH, 10% of contract is executed

❶ find optimizations

for a program ρ find an
observationally equivalent program τ
where cost of τ < cost of ρ

Ingredients

- constraint solver Z3
- unbounded superoptimization*

* Jangda & Yorsh, Onward! 2017

❶ find optimizations

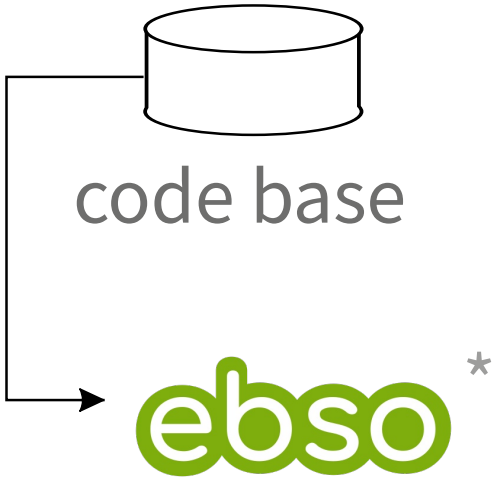
$$\exists n, \exists \tau, \forall \vec{x}.\ \mathsf{init}(\vec{x}, \sigma_0) \wedge \sigma_0 \overset{\rho}{\twoheadrightarrow} \sigma_{|\rho|+1} \wedge$$

$$\sigma_0 \equiv \sigma'_0 \wedge \sigma_{|\rho|+1} \equiv \sigma'_n \wedge$$

$$\forall j < n.\ \bigwedge_{\iota \in \mathcal{I}} \tau(j) = \iota \implies \sigma'_j \xrightarrow{\iota} \sigma'_{j+1} \wedge \bigvee_{\iota \in \mathcal{I}} \tau(j) = \iota \wedge$$

$$\mathsf{cost}(\rho, \sigma_0) > \mathsf{cost}(\tau, \sigma'_0)$$

❶ find optimizations



250 most-called smart contracts

106 798 "basic+" blocks

split to blocks of 6 instructions

54 301 blocks

on a cluster w/ 15 min/2 GB

1580 optimizations

* Nagele & Schett, pre-proc. LOPSTR'19

❷ generate rules

```
PUSH 0
SUB
PUSH 3
ADD
SHA3
```

≡

```
PUSH 3
SUB
SHA3
```

ⓘ generalize

ⓘⓘ strip

```
PUSH 0
SUB
PUSH x
ADD
```

⟹

```
PUSH x
SUB
```

❷ generate rules

ⓘ generalize

≈ finding a substitution

ⓘⓘ strip

≈ finding a context

exhaustive search with pruning

$$\exists \vec{x}.\ \mathsf{init}(\vec{x}, \sigma_0) \wedge \sigma_0 \equiv \sigma'_0 \wedge$$

$$\sigma_0 \xrightarrow{\rho}\!\!\!\!\rightarrow \sigma_{|\rho|+1} \wedge \sigma'_0 \xrightarrow{\tau}\!\!\!\!\rightarrow \sigma'_{|\tau|+1} \wedge$$

$$\sigma_{|\rho|+1} \not\equiv \sigma'_{|\tau|+1}$$

❷ generate rules

1580
optimizations

generalize & strip

on a cluster w/ 15 min/2 GB

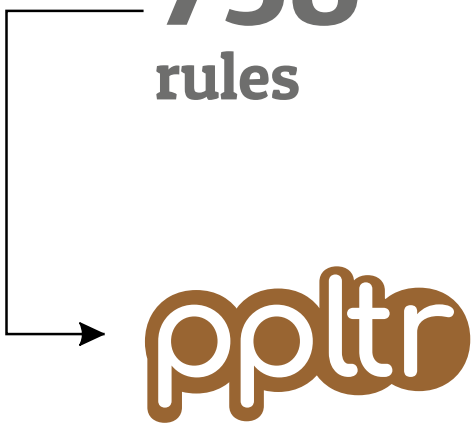
remove **duplicate** rules

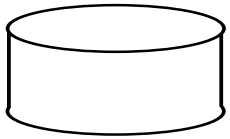
sorg

758 rules

❸ feedback

758
rules





code base

ⓐ apply to **rules**

reduced **4** rules

ⓑ apply to **code base**

changed 17 255 blocks

repeat: ❶ & ❷ to 435 rules

993 rules

Branch: master

ppltr / eval / **17-reduced-rules.csv**

mariaschett Add final right reduced rules

1 contributor

1391 lines (1391 sloc) | 205 KB

Search this file…

rule lhs	rule rhs
DUP1 SWAP3 SWAP2 POP POP	SWAP1 POP SWAP1
PUSH cw_2 DUP1 SWAP2 PUSH cw_2 SWAP1 DUP2	PUSH cw_2 DUP1 DUP1 SWAP3 DUP3
DUP3 MLOAD DUP4 SWAP1	DUP3 DUP4 MLOAD
DUP1 MLOAD SWAP3 SWAP2 SWAP1 POP	MLOAD SWAP2 SWAP1
DUP2 DUP1 DUP4 SUB DUP2	DUP2 PUSH 0x00 DUP4
SWAP2 SWAP1 PUSH cw_2 DUP1 DUP5 SWAP3	PUSH cw_2 PUSH cw_2 DUP3 SWAP5 SWAP4
DUP1 MLOAD SWAP2 LT POP	SWAP1 POP MLOAD
DUP2 SWAP1 PUSH cw_1 SHA3 SSTORE POP	PUSH cw_1 SHA3 SSTORE

```
ISZERO
ISZERO
ISZERO
```

$\Longrightarrow$

```
ISZERO
```

```
PUSH x
PUSH y
SWAP1
```

$\Longrightarrow$

```
PUSH y
PUSH x
```

```
CALLVALUE
DUP1
```

$\Longrightarrow$

```
CALLVALUE
CALLVALUE
```

each applied × **7700+** in **1000** most-called smart contracts

```
PUSH 0
DUP6
DUP5
SUB
LT
ISZERO
```

$\Longrightarrow$

```
PUSH 1
```

saves **15 g** and **5** instructions

Rewriting

can rules just be applied exhaustively?

yes, termination prover **WANDA** shows termination

does result depend on which rules are applied?

yes, confluence prover **CSI** shows non-confluence

```
PUSH 1
PUSH 1
```

vs

```
PUSH 1
DUP1
```

can conflicts between rules be resolved?

future work: apply **completion**

Outlook

how do we get these rules into a compiler?

- use a peephole optimizer **DSL** like GCC,
- or have pplr generate code

how can we apply this to Michelson/Move/...?

- encode operational semantics in SMT
- use existing code base to start

Advertisement

"Synthesis of Super-Optimized Smart Contracts using Max-SMT"

Elvira Albert, Pablo Gordillo, Maria Schett, Albert Rubio

@ CAV | Session 5A | Wed, 07/22

```
PUSH 0          PUSH x
SUB      ⇒      SUB
PUSH x
ADD
```

56 500 $ of execution cost

~4.5 % of space

1. find optimizations
2. generate rules
3. feedback

ebso sorg ppltr

... available on **github!**

questions? mail@ {maria-a-schett.net, jnagele.net}