Fast verified computation for BIR

Jules TIMMERMAN
Supervised by Karl PALMSKOG and Mads DAM

August 19th 2024

Table of Contents

What is HOL4 ?

- Proof Assistant
- Based on Higher-Order Logic
- Mainly developed at Cambridge
- Meta-language : SML

HolBA

- Library of HOL4
- Binary Analysis
- Weakest precondition, Side-channels, Contracts, Out-of-order, Symbolic execution...
- Made in Stockholm

What is BIR ?

- BIR : Binary Intermediate Representation
- Machine independent
- Represents programs in HOL
- Usually, BIR generated (lifter...)

What is BIR ?

- BIR : Binary Intermediate Representation
- Machine independent
- Represents programs in HOL
- Usually, BIR generated (lifter...)

Example (BIR Expression)

```
BExp_BinExp BIExp_Plus
  (BExp_Den (BVar "r0"))
  (BExp_Const (Imm64 1w))
```

Overview of the Syntax

- Expressions
  - Constants
  - Variable Environment Read
  - Operations (Unary / Binary)
  - If Then Else / Predicates
  - Memory Operations (Store / Load)
- Statements
  - Assign in environment
  - (Conditional) Jumps
- Programs / Blocks / Labels

cv_compute library

- Fast computation library for *ground terms*
- Translate to a type called cv
- `cv ::= Num | Pair cv cv`

fact n for different values of n

n	Candle	HOL4	H.Light	Isabelle
256	<1 ms	2.3 s	0.6 s	14 s
512	<1 ms	4.1 s	3.5 s	202 s
1024	<1 ms	127 s	17.6 s	2451 s
2048	11 ms	684 s	86.1 s	—
32768	0.9 s	—	—	—

primes_upto n for different values of n

n	Candle	HOL4	H.Light	Isabelle
256	<1 ms	0.5 s	1.3 s	2.6 s
512	<1 ms	1.6 s	5.2 s	9.8 s
1024	2 ms	6.3 s	20.7 s	35.6 s
2048	9 ms	24.2 s	83.4 s	132 s
32768	1.7 s	—	—	—

Automation

Manually converting to cv can be tedious...

Automation

Manually converting to cv can be tedious...

- Automatic translation using cv_transLib
- Also support *deep embedding* terms

Automation

Manually converting to `cv` can be tedious...

- Automatic translation using `cv_transLib`
- Also support *deep embedding* terms

Limitations

- Higher-order
- Free variables

Motivation

Why recreate two semantics ?

Motivation

Why recreate two semantics ?

- `eval` : Easier to understand
- Smaller : easier to test `cv`
- Fairly close to the original

Key differences with HolBA

- Typing less enforced
  - Environments
  - If / Then / Else
- Instead, Typing relation
- Fewer operations (ex : $+$ and bitwise `AND` for binary operations)

Alternative representation

Limitation of translation...

Alternative representation

Limitation of translation $\Rightarrow$ Alternate Syntax

Alternative representation

Limitation of translation $\Rightarrow$ Alternate Syntax

Datatype	BIR	CV
Environment	`ident -> val option`	`(ident # val) list`
Memory maps	`num |-> num`	`(num # num) list`
Program Counter State Block	Records	Tuples

Benchmarks

Example	`EVAL`	`cv_compute`		
		Compute	Embedding	Translation
Increment	6.7	0.2	4.4	11.5
Mem Incr	32.8	0.06	7.4	24.6
Sum List	0.1	0.2	0.3 + 0.2	106
Jump Chain	0.3	0.02[1]	90 + 0.1	0.4

- Embedding for programs : expressions + statements + state
- Good Results for expressions
- Program stepping need some work...
- Embedding for programs : Program + State

[1] Rewrite time : 14

Some issues I faced

- Low performance using `cv` initially
  $\Rightarrow$ Use `cv_trans_deep_embedding`
- Weird errors regarding non `cv` type
  $\Rightarrow$ Don't use record types
- Preconditions with `cv_auto_trans`
  $\Rightarrow$ Not propagated. Translate the problematic function yourself.

Implementation in HolBA

Two possibilities :

- Keep Translation
  - Less work
  - Less performance
  - More flexibility for datatypes in theory
- Change HolBA types
  - Big refactor
  - More performance
  - Less flexibility

Future work

- Program stepping performance
  - What to embed ?
  - Measure without translation to `bir_cv`
- Program multi-stepping
  - Embed state ?
- Other operations (cf. binary operations / statements)