Porting Wasm Semantics to ACL2 With Coding Agents

Ray Myers
The Coding Agency

ACL2 Workshop 2026

~~Porting~~ Wasm Semantics to ACL2 With Coding Agents

Ported!

Why?

Enable practical proofs of
industrial software

Why?

People



Coding Agents

Tools

Why bring Wasm semantics to ACL2 using coding agents?

ACL2

Awesome industrial prover

WebAssembly

Widely-deployed compilation target

Official formal semantics

Coding Agent

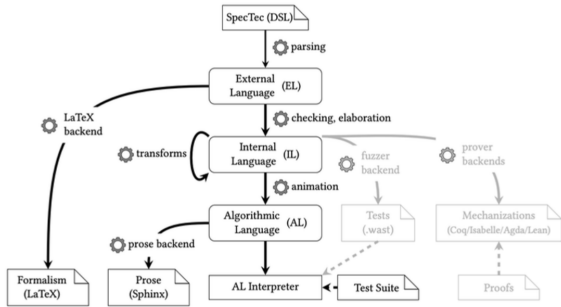
(as currently conceived)

Language Model in a chat loop
that calls tools for developing software

Semantics work is toilsome

Agents can help, *given boundaries*





Bringing the WebAssembly Standard up to Speed with SpecTec

DONGJUN YOUN, WONHO SHIN, JAEHYUN LEE, and SUKYOUNG RYU, KAIST, South Korea
JOACHIM BREITNER, Independent, Germany
PHILIPPA GARDNER, Imperial College London, United Kingdom
SAM LINDLEY, The University of Edinburgh, United Kingdom
MATIJA PRETNAR, University of Ljubljana, Slovenia
XIAOJIA RAO, Imperial College London, United Kingdom
CONRAD WATT, University of Cambridge, United Kingdom
ANDREAS ROSSBERG, Independent, Germany

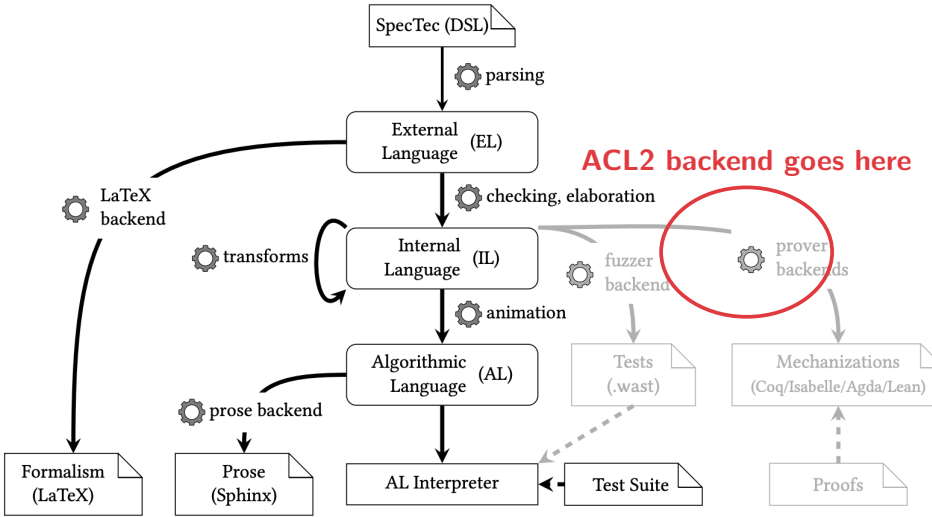
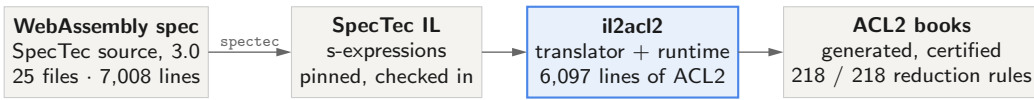
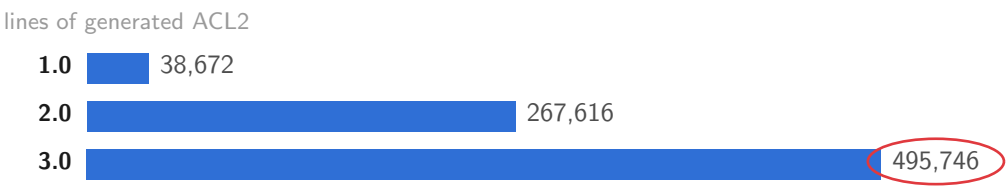
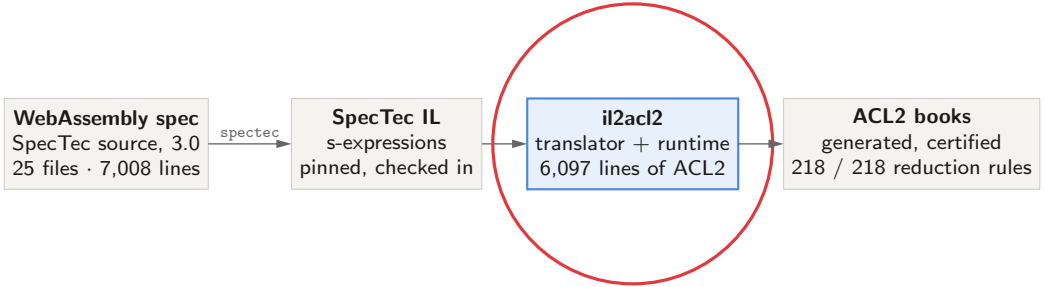


Fig. 3. An overview of the Wasm SpecTec architecture

Pipeline


WebAssembly spec
SpecTec source, 3.0
25 files · 7,008 lines
spectec
SpecTec IL
s-expressions
pinned, checked in
il2acl2
translator + runtime
6,097 lines of ACL2
ACL2 books
generated, certified
218 / 218 reduction rules


lines of generated ACL2
1.0
38,672
2.0
267,616
3.0
495,746



Requires trusting $\approx$ 6K lines

SpecTec rule

```
rule Step_pure/if-true:
  (CONST I32 c) (IF bt instr_1* ELSE instr_2*)  ~>  (BLOCK bt instr_1*)
  -- if c =/= 0
```

ACL2 rule

(cleaned up)

```
(defun if-true (instrs)
  (case-match instrs
    (((:const (:i32) c) (:ifelse bt then else))
     (and (true-listp then)
          (true-listp else)
          (not (equal c 0))
          (list (list (list :block bt then)))))))
```

SpecTec rule

```
rule Step_pure/if-true:
  (CONST I32 c) (IF bt instr_1* ELSE instr_2*)  ~>  (BLOCK bt instr_1*)
  -- if c =/= 0
```

Generated ACL2 rule

```
(AND (TRUE-LISTP %A0)
     (EQUAL (ACL2::LEN %A0) 2)
     (LET* ((%T16 (CAR %A0)))
       (AND (CONSP %T16)
            (EQ (CAR %T16) :CONST)
            (TRUE-LISTP (CDR %T16))
            (EQUAL (ACL2::LEN (CDR %T16)) 2)
            (LET* ((%T17 (CAR (CDR %T16))))
              (AND (CONSP %T17)
                   (EQ (CAR %T17) :I32)
                   (TRUE-LISTP (CDR %T17))
                   (EQUAL (ACL2::LEN (CDR %T17)) 0)
                   (LET* ((C (CAR (CDR (CDR %T16))))
                          (%T18 (CAR (CDR %A0))))
                     (AND (CONSP %T18)
                          (EQ (CAR %T18) :IFELSE)
                          (TRUE-LISTP (CDR %T18))
                          (EQUAL (ACL2::LEN (CDR %T18)) 3)
                          (LET* ((BT (CAR (CDR %T18)))
                                 (%T19 (CAR (CDR (CDR %T18)))))
                            (AND (TRUE-LISTP %T19)
                                 (LET* ((INSTR_1-STAR %T19)
                                        (%T20 (CAR (CDR (CDR (CDR %T18))))))
                                   (AND (TRUE-LISTP %T20)
                                        (LET* ((INSTR_2-STAR %T20))
                                          (AND (NOT (EQUAL C 0))
                                               (LIST (%RULE "Step_pure/if-true"
                                                            (LIST (LIST :BLOCK
                                                                        BT
                                                                        INSTR_1-STAR)))))))))))))))))
```

Process

Phases

Reference: SpecTec DSL, Kestrel books

1. Direct agentic port of Wasm 1.0
2. Redo 1.0 as translator layer
3. Upgrade translator for Wasm 3.0

Tools

Wasm 1.0 Port

Models: Claude Opus 4.6

Agent: OpenHands, no MCP

Token cost: $300

Wasm 3.0 Translator

Models: Claude Opus 5.5, Fable 5.2

Agent: Claude Code, no MCP

Token cost: $300

Agent orchestration

Single agent, repeated

Ralph Wiggum pattern

Agent plans and logs available

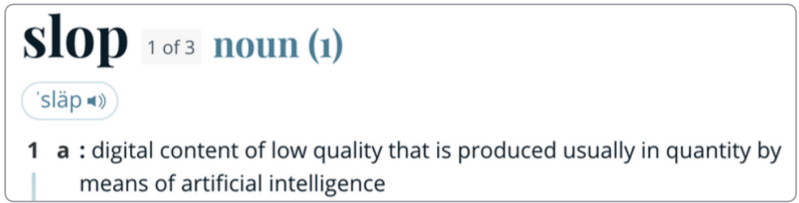
Confession

I don't know what I'm doing



Beginner

- 20 years developing industrial software
- Only 2 years with formal methods
- Compiler literate, new to semantics
- No PhD



Merriam-Webster
2025 Word of the Year

I don't want to drop slop at your doorstep

I welcome guidance to make
this a proper contribution

Evidence

ACL2 books certify

SpecTest3 suite: 99.98%

64,005 examples pass, 13 fail

Demo

Module | Edit source

Example: Factorial (i64)

WAT → 107 bytes

printed by wabt from the bytes

highlight: a display aid, not part of the semantics

```
(module
  (type (;0;) (func (param i64) (result i64)))
  (func (;0;) (type 0) (param i64) (result i64)
    (local i64)
    i64.const 1
    local.set 1
    block  ;; label = @1
      loop  ;; label = @2
        local.get 0
        i64.const 1
        i64.le_u
        br_if 1 (;@1;)
        local.get 1
        local.get 0
        i64.mul
        local.set 1
        local.get 0
        i64.const 1
        i64.sub
        local.set 0
        br 0 (;@2;)
      end
    end
    local.get 1)
  (func (;1;) (type 0) (param i64) (result i64)
    local.get 0
```

fac (i64) → i64 (arg 0 : i64 20) fuel 200000

Step | Step × 10 | Next line | Run to end | Restart | Stop

ACL2 semantics returned

i64 **2432902008176640000**

steps	run	load
240	211 ms (0.88 ms/step)	21 ms

Native WebAssembly V8 returned

i64 **2432902008176640000**

steps	run	load
not observable	0.10 ms	0.10 ms

Configuration after 240 steps top level

Rules of the last step, outermost first — WebAssembly/spec@943c5df

Step/pure › Step_pure/frame-vals

Instructions ■ operands ■ redex □ rest

i64 2432902008176640000

normal form: no step applies

Locals innermost frame

none

Memory changed by the last 240 steps

no byte changed

Module | Edit source

Example: Factorial (i64)

WAT → 107 bytes

printed by wabt from the bytes

highlight: a display aid, not part of the semantics

```
1  (module
2    (type (;0;) (func (param i64) (result i64)))
3    (func (;0;) (type 0) (param i64) (result i64)
4      (local i64)
5      i64.const 1
6      local.set 1
7      block  ;; label = @1
8        loop  ;; label = @2
9          local.get 0
10         i64.const 1
11         i64.le_u
12         br_if 1 (;@1;)
13         local.get 1
14         local.get 0
15         i64.mul
16         local.set 1
17         local.get 0
18         i64.const 1
19         i64.sub
20         local.set 0
21         br 0 (;@2;)
22       end
23     end
24     local.get 1)
25   (func (;1;) (type 0) (param i64) (result i64)
26     local.get 0
```

fac (i64) → i64 (arg 0 : i64 20) fuel 200000

Step | Step × 10 | Next line | Run to end | Restart | Stop

ACL2 semantics stepping

in progress

steps	run	load
4	5 ms	8 ms

Native WebAssembly V8 returned

i64 **2432902008176640000**

steps	run	load
not observable	0.20 ms	< 0.1 ms

Configuration after 4 steps frame{1} › label{1} › label{0} › label{0}

Rules of the last step, outermost first — WebAssembly/spec@943c5df

Step/ctxt-frame › Step/ctxt-label › Step/ctxt-instrs › Step/ctxt-label › Step/read › **Step_read/loop**

Instructions ■ operands ■ redex □ rest

```
local.get 0
i64.const 1
i64.le_u
br_if 1
local.get 1
local.get 0
i64.mul
local.set 1
local.get 0
i64.const 1
```

Locals innermost frame

local 0	i64 20
local 1	i64 1

Memory changed by the last step

no byte changed

Module | Edit source

Example: Factorial (i64)

WAT → 107 bytes

printed by wabt from the bytes

highlight: a display aid, not part of the semantics

```
1  (module
2    (type (;0;) (func (param i64) (result i64)))
3    (func (;0;) (type 0) (param i64) (result i64)
4      (local i64)
5      i64.const 1
6      local.set 1
7      block  ;; label = @1
8        loop  ;; label = @2
9          local.get 0
10         i64.const 1
11         i64.le_u
12         br_if 1 (;@1;)
13         local.get 1
14         local.get 0
15         i64.mul
16         local.set 1
17         local.get 0
18         i64.const 1
19         i64.sub
20         local.set 0
21         br 0 (;@2;)
22       end
23     end
24     local.get 1)
25   (func (;1;) (type 0) (param i64) (result i64)
26     local.get 0
```

fac (i64) → i64 (arg 0 : i64 20) fuel 200000

Step | Step × 10 | Next line | Run to end | Restart | Stop

one step of the reduction relation

ACL2 semantics stepping

in progress

steps	run	load
5	7 ms	8 ms

Native WebAssembly V8 returned

i64 **2432902008176640000**

steps	run	load
not observable	0.20 ms	< 0.1 ms

Configuration after 5 steps

frame{1} › label{1} › label{0} › label{0}

Rules of the last step, outermost first — WebAssembly/spec@943c5df

Step/ctxt-frame › Step/ctxt-label › Step/ctxt-instrs › Step/ctxt-label › Step/ctxt-label › Step/ctxt-instrs › Step/read › **Step_read/local.get**

Instructions ■ operands ■ redex □ rest

```
i64 20
i64 1
i64.le_u
br_if 1
local.get 1
local.get 0
i64.mul
local.set 1
local.get 0
i64.const 1
```

Locals innermost frame

local 0	i64 20
local 1	i64 1

Memory changed by the last step

no byte changed

Module | Edit source

Example: Factorial (i64)

WAT → 107 bytes

printed by wabt from the bytes

highlight: a display aid, not part of the semantics

```
(module
  (type (;0;) (func (param i64) (result i64)))
  (func (;0;) (type 0) (param i64) (result i64)
    (local i64)
    i64.const 1
    local.set 1
    block  ;; label = @1
      loop  ;; label = @2
        local.get 0
        i64.const 1
        i64.le_u
        br_if 1 (;@1;)
        local.get 1
        local.get 0
        i64.mul
        local.set 1
        local.get 0
        i64.const 1
        i64.sub
        local.set 0
        br 0 (;@2;)
      end
    end
    local.get 1)
  (func (;1;) (type 0) (param i64) (result i64)
```

fac (i64) → i64 (arg 0 : i64 20) fuel 200000

Step | Step × 10 | Next line | Run to end | Restart | Stop

ACL2 semantics stepping

in progress

steps	run	load
6	9 ms	8 ms

Native WebAssembly V8 returned

i64 **2432902008176640000**

steps	run	load
not observable	0.20 ms	< 0.1 ms

Configuration after 6 steps

frame{1} › label{1} › label{0} › label{0}

Rules of the last step, outermost first — WebAssembly/spec@943c5df

Step/ctxt-frame › Step/ctxt-label › Step/ctxt-instrs › Step/ctxt-label › Step/ctxt-label › Step/ctxt-instrs › Step/pure › **Step_pure/relop**

Instructions ■ operands ■ redex □ rest

```
i32 0
br_if 1
local.get 1
local.get 0
i64.mul
local.set 1
local.get 0
i64.const 1
i64.sub
local.set 0
```

Locals innermost frame

```
local 0   i64 20
local 1   i64 1
```

Memory changed by the last step

no byte changed

Acknowledgements

Jim White, testing and encouragement

OpenHands, my employer during phase 1

One more thing...

The Real Demo

This presentation was rendered in ACL2 Wasm

```
# TeX, compiled to WebAssembly
$ file ../build/dl/tex.wasm
../build/dl/tex.wasm: WebAssembly (wasm) binary version 0x1 (MVP module)

# acl2-wasm-tex: a saved ACL2 image = our engine's books + tex.wasm instantiated
$ time ../build/acl2-wasm-tex deck.tex
ACL2 Version 8.7: WebAssembly engine WE::FAST-STEP (certified book engine/fast/fast)
running tex.wasm (517,692 bytes) on deck.tex
This is e-TeX, Version 3.14159265-2.6 (preloaded format=latex 2022.5.1)
(deck.tex LaTeX2e <2020-02-02> patch level 2 (slides.tex) (figdims.tex) [1]
[100 M WebAssembly instructions, 31 s]
 [2] [3] [4] [5] [6] [7]
[200 M WebAssembly instructions, 62 s]
 [8] [9] [10] [11]
[300 M WebAssembly instructions, 93 s]
 [12] [13]
[400 M WebAssembly instructions, 126 s]
 [14] [15] [16] [17] [18] [19] [20]
[500 M WebAssembly instructions, 157 s]
 [21] [22] [23] [24] [25] [26] [27] [28] [29] [30]
Output written on deck.dvi (30 pages, 18360 bytes).
tex.wasm returned: 570,806,146 WebAssembly instructions in 178.5 s (3.20 M/s), 60,964 host calls
real    2m59.322s

# ordinary tools from here: DVI to PDF, fonts and images embedded
$ dvipdfmx deck.dvi
deck.dvi -> deck.pdf
2240323 bytes written
```

Thank you

github.com/raymyers/wasm-acl2

raymyers.org