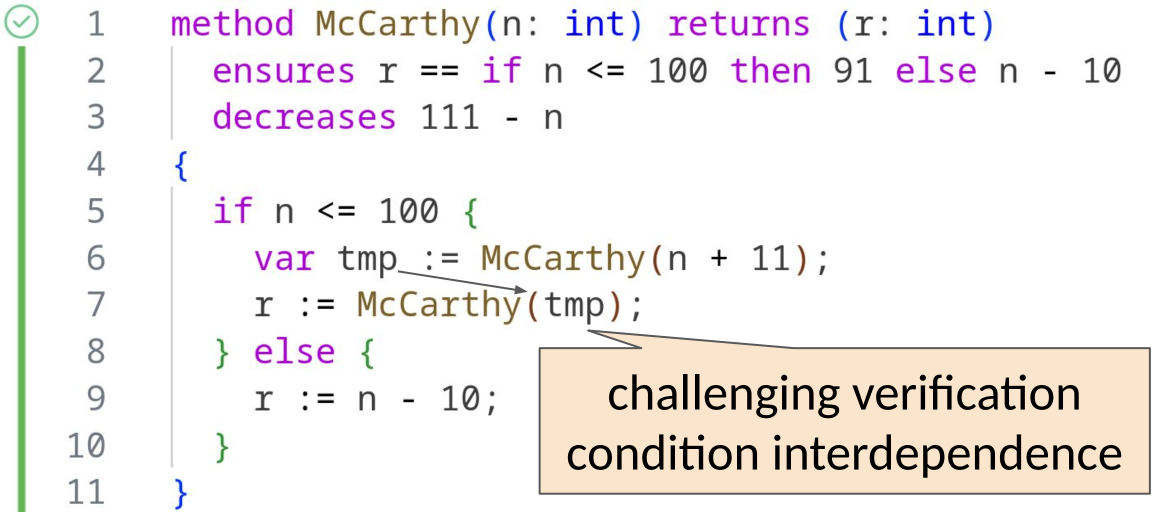
Verified VCG and Verified Compiler for Dafny

Daniel Nezamabadi
ETH Zurich

Magnus Myreen
Chalmers University of Technology and University of Gothenburg

Yong Kiam Tan
I^{2}R, A*STAR and NTU Singapore

What does the checkmark actually mean?



```
method McCarthy(n: int) returns (r: int)
  ensures r == if n <= 100 then 91 else n - 10
  decreases 111 - n
{
  if n <= 100 {
    var tmp := McCarthy(n + 11);
    r := McCarthy(tmp);
  } else {
    r := n - 10;
  }
}
```

challenging verification condition interdependence

McCarthy’s 91 function in Dafny (VSCode)

Our Answer:

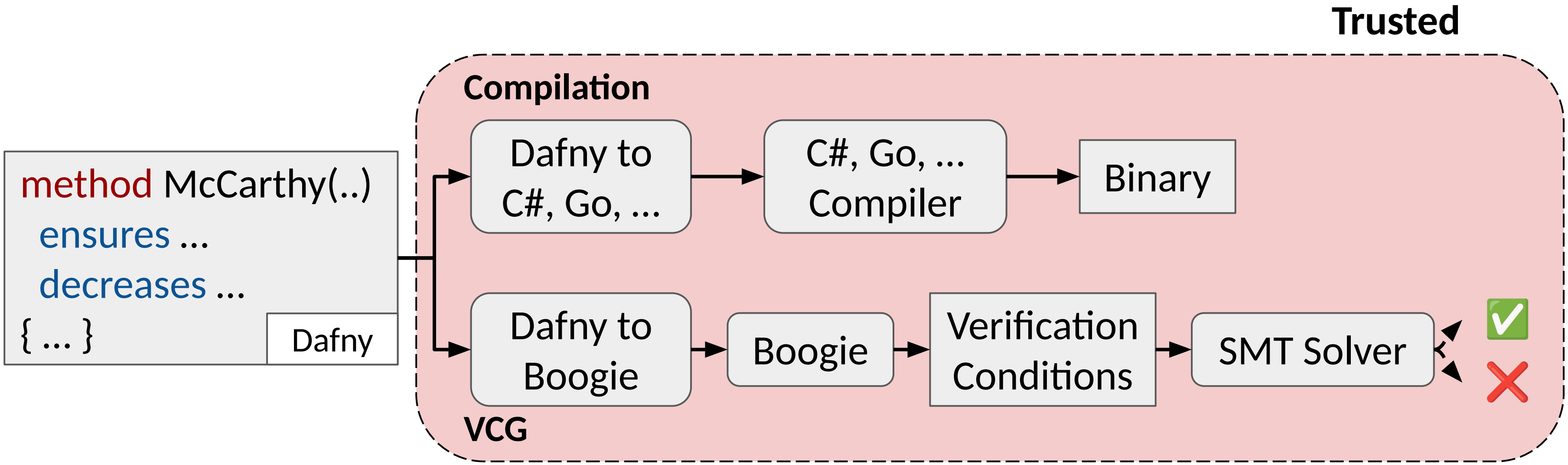
compile mccarthy = inr *mccarthy_cml* ∧ … ⇒
AppReturns (INT *n*) (… [*mccarthy_cml*] …)
(INT (if *n* <= 100 then 91 else *n* - 10))

McCarthy compiled to CakeML

Hoare triple

Paper: Section 1 (Introduction)

The Life of a Dafny Program

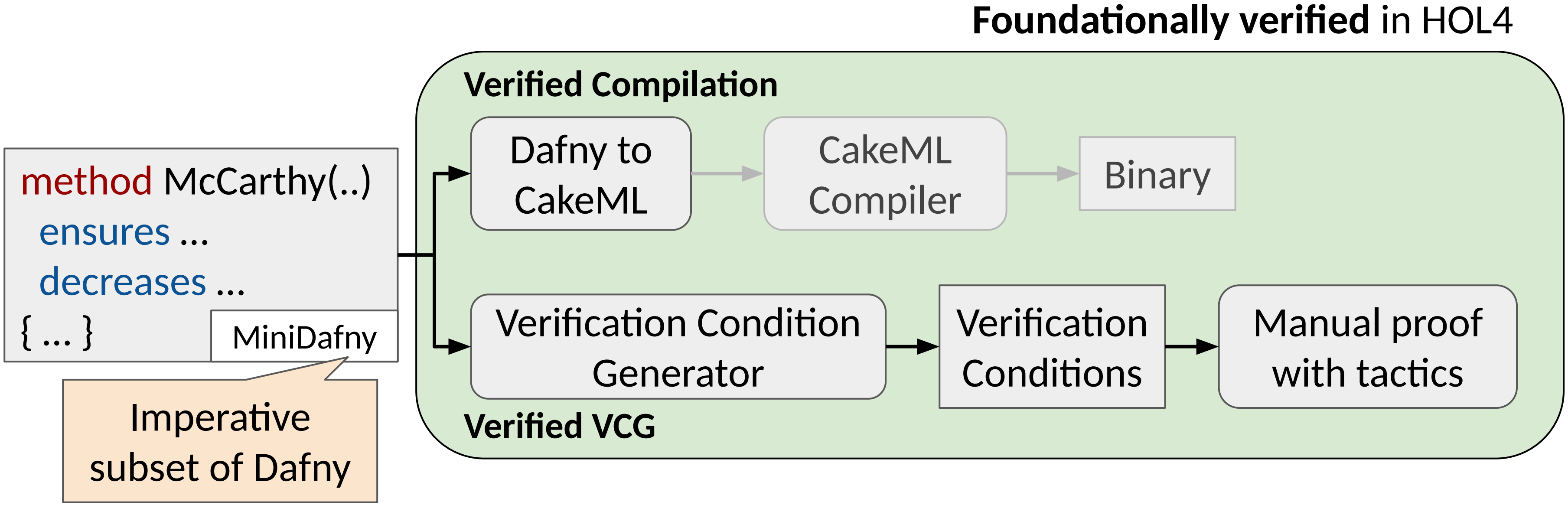


```
method McCarthy(..)
  ensures …
  decreases …
{ … }
```

"[...] report 24 previously-unknown Dafny compiler bugs [...], of which 9 are soundness issues."*

* A.F. Donaldson et al., "Randomised Testing of the Compiler for a Verification-Aware Programming Language", IEEE ICST, 2024

Our Work



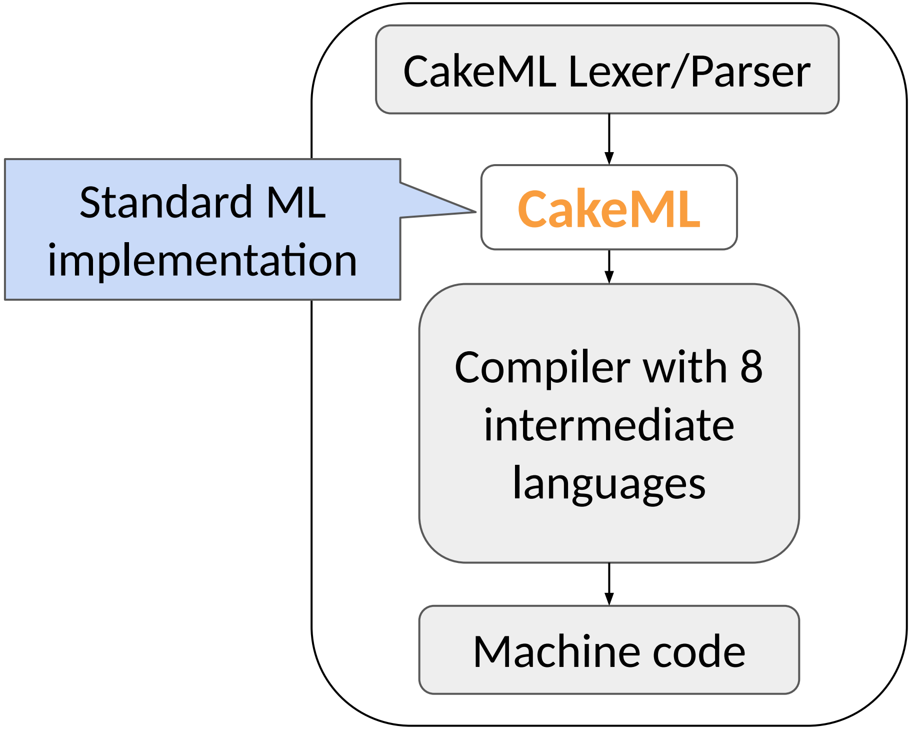
Grey arrows: existed before this project

Foundationally verified in HOL4

Verified Compilation

Dafny to CakeML

What is CakeML?



Standard ML implementation

CakeML Lexer/Parser

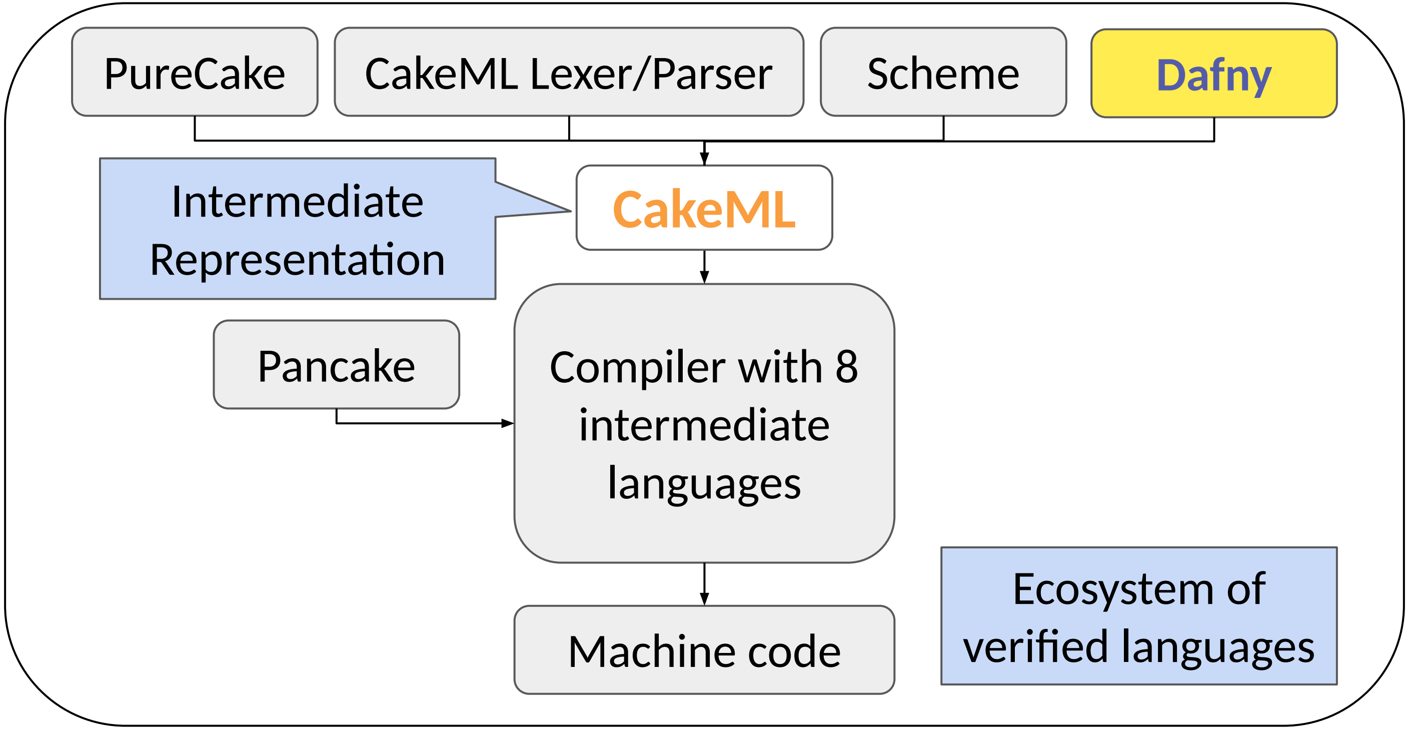
CakeML

Compiler with 8 intermediate languages

Machine code

Verified in HOL4

What is CakeML?


PureCake
CakeML Lexer/Parser
Scheme
Dafny
Intermediate Representation
CakeML
Pancake
Compiler with 8 intermediate languages
Machine code
Ecosystem of verified languages

Verified in HOL4

MiniDafny to CakeML: Variables

```
var b: bool;
b := true;
```

MiniDafny

Zero-initialized reference (instead of option)

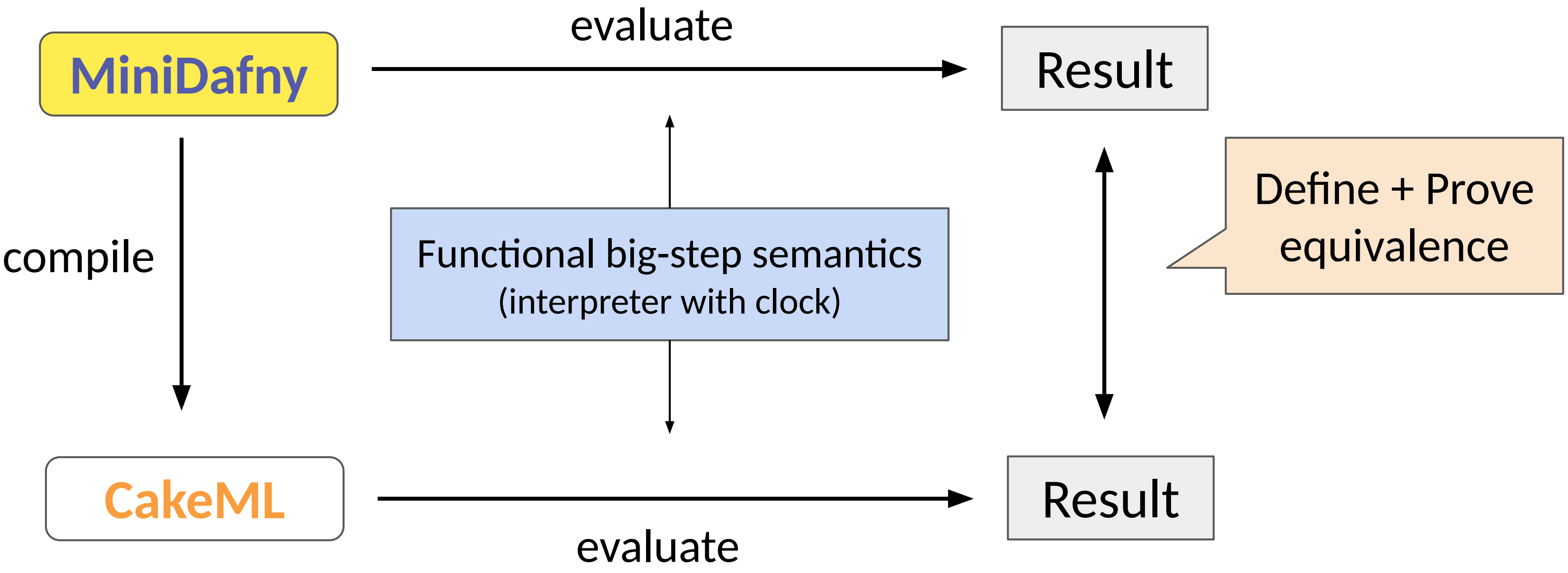
```
let val b = ref 0 in
  b := True;
```

CakeML

Valid VCs ⇒ No undefined behavior

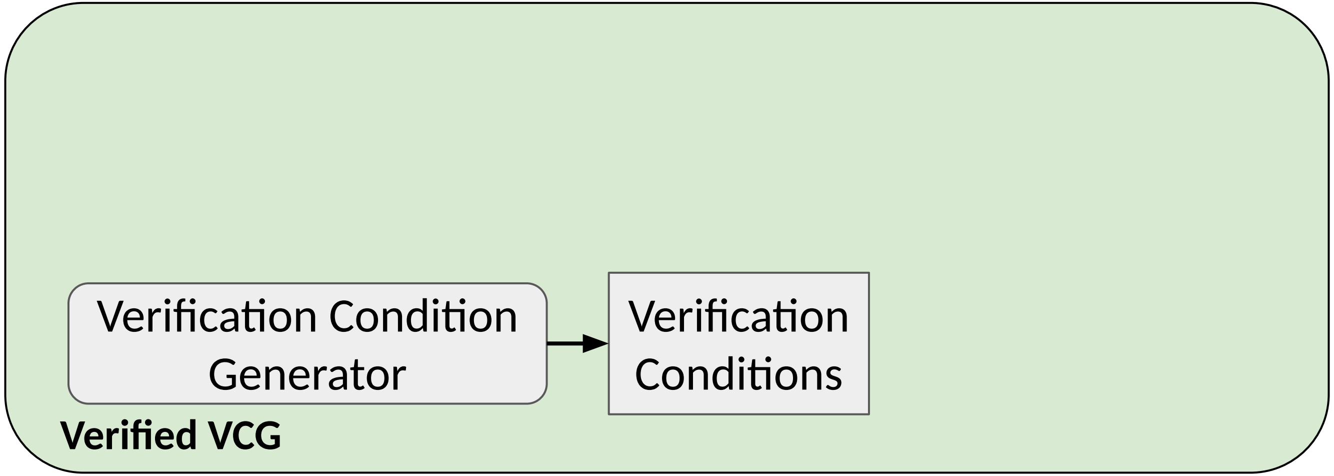
Paper: Section 3.1 (Compiler Implementation: Variables, Methods)

MiniDafny to CakeML: Proof Sketch



Paper: Section 3.2 (Compiler Correctness)

Foundationally verified in HOL4

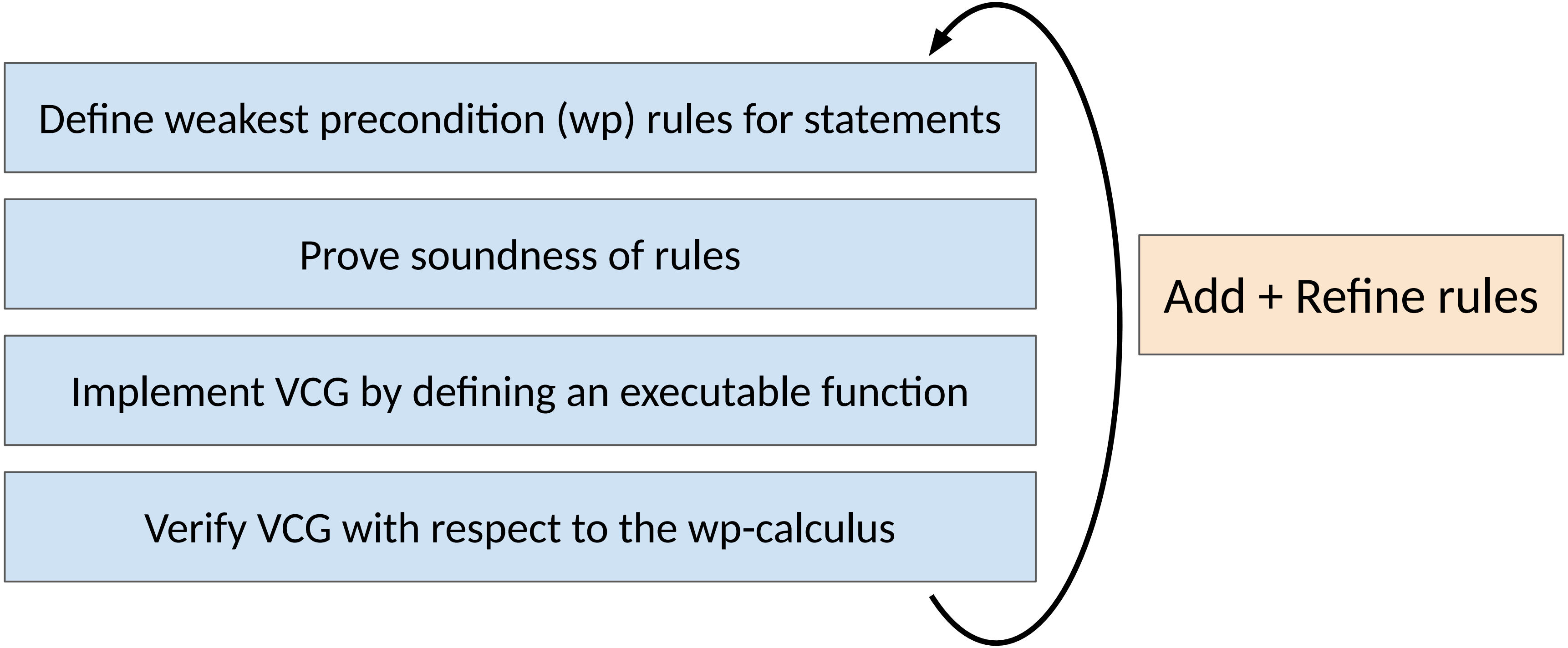


Verification Condition Generator

Verification Conditions

Verified VCG

Verified Verification Condition Generation



Define weakest precondition (wp) rules for statements

Prove soundness of rules

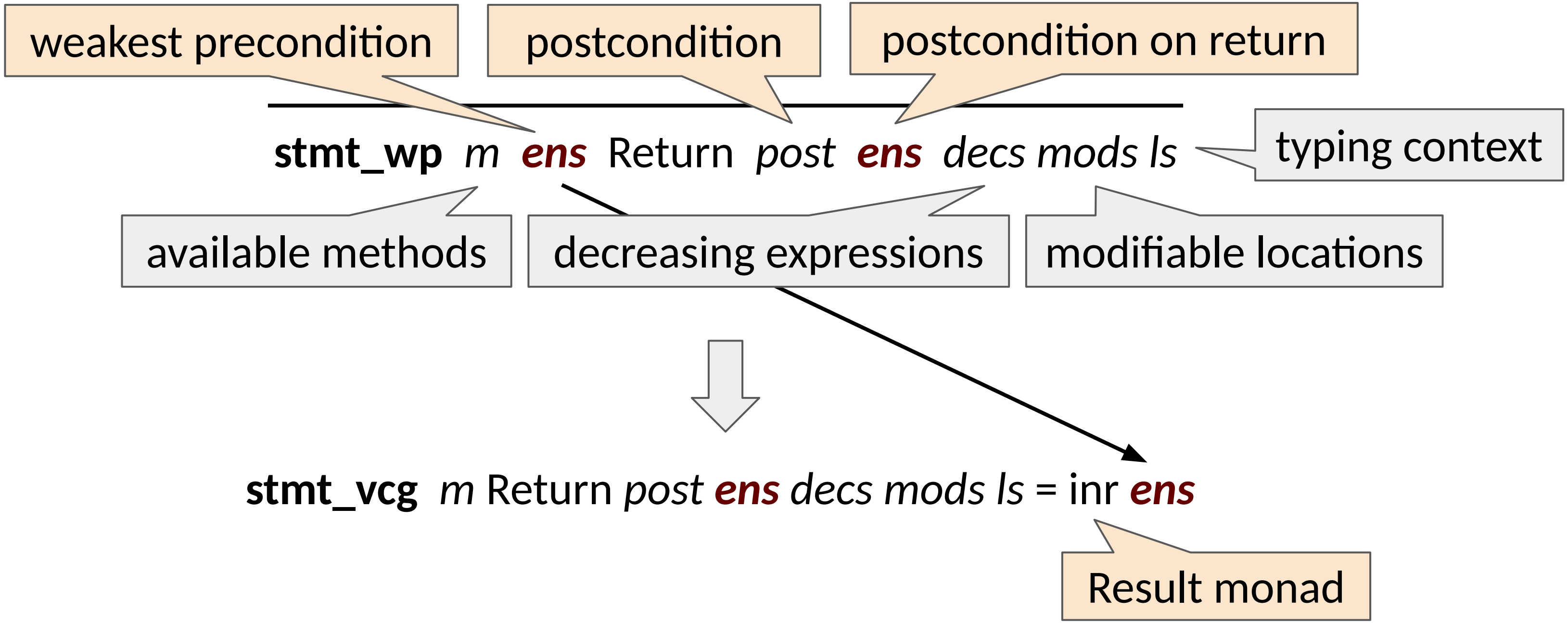
Implement VCG by defining an executable function

Verify VCG with respect to the wp-calculus

Add + Refine rules

Paper: Section 4 (Verified VCG)

wp-calculus and VCG



weakest precondition

postcondition

postcondition on return

stmt_wp *m* ***ens*** Return *post* ***ens*** *decs mods ls*

typing context

available methods

decreasing expressions

modifiable locations

stmt_vcg *m* Return *post* ***ens*** *decs mods ls* = inr ***ens***

Result monad

Paper: Section 4.1 (Weakest Precondition Calculus), Section 4.3 (Verified VCG Implementation)

wp-calculus: Dealing with the Heap

```
a := new int[2];
a[0] := 67;
…
```

MiniDafny

Quantifies over heaps with new allocations

```
ForallHeap [] (forall a: array<int> ::
 (a.Length = 2 ∧ …) ⇒
  SetPrev (ForallHeap [a]
   (a[0] = 67 ∧ a[1] = PrevHeap (a[1])
    ⇒ …)))
```

a is “havoced”

wp (Sketch)

Also support old-expressions, general arrays, and `modifies` on variables

Paper: Section 4.1 (Weakest Precondition Calculus: Array Update, Array Allocation, While, Method Call)

wp-calculus: Soundness

⊢ stmt_wp *m* ***reqs stmt post ens*** *decs ls* ⇒

conditions_hold *st env* ***reqs*** ∧ … ⇒

∃ *st′ ret*.

eval_stmt *st env* ***stmt*** *st′ ret* ∧

(case *ret* of

| Rcont ⇒ conditions_hold *st′ env* ***post***

| Rstop Sret ⇒ conditions_hold *st′ env* ***ens***

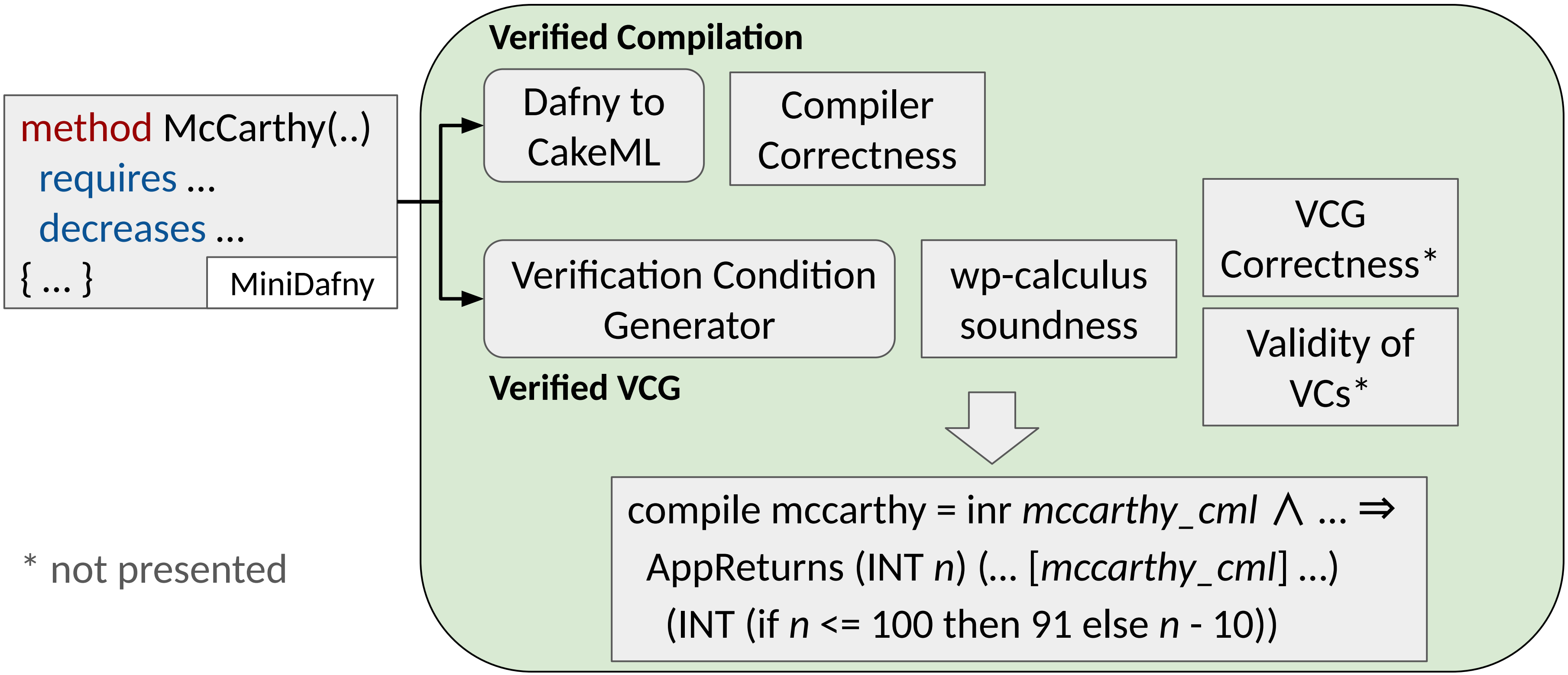
| Rstop (Serr _) ⇒ F) ∧ …

Termination comes from here

Paper: Section 4.2 (Weakest Precondition Soundness Statement)

Putting it All Together

Foundationally verified in HOL4



```
method McCarthy(..)
  requires ...
  decreases ...
{ ... }
```

MiniDafny

Verified Compilation

Dafny to CakeML

Compiler Correctness

Verification Condition Generator

wp-calculus soundness

VCG Correctness*

Validity of VCs*

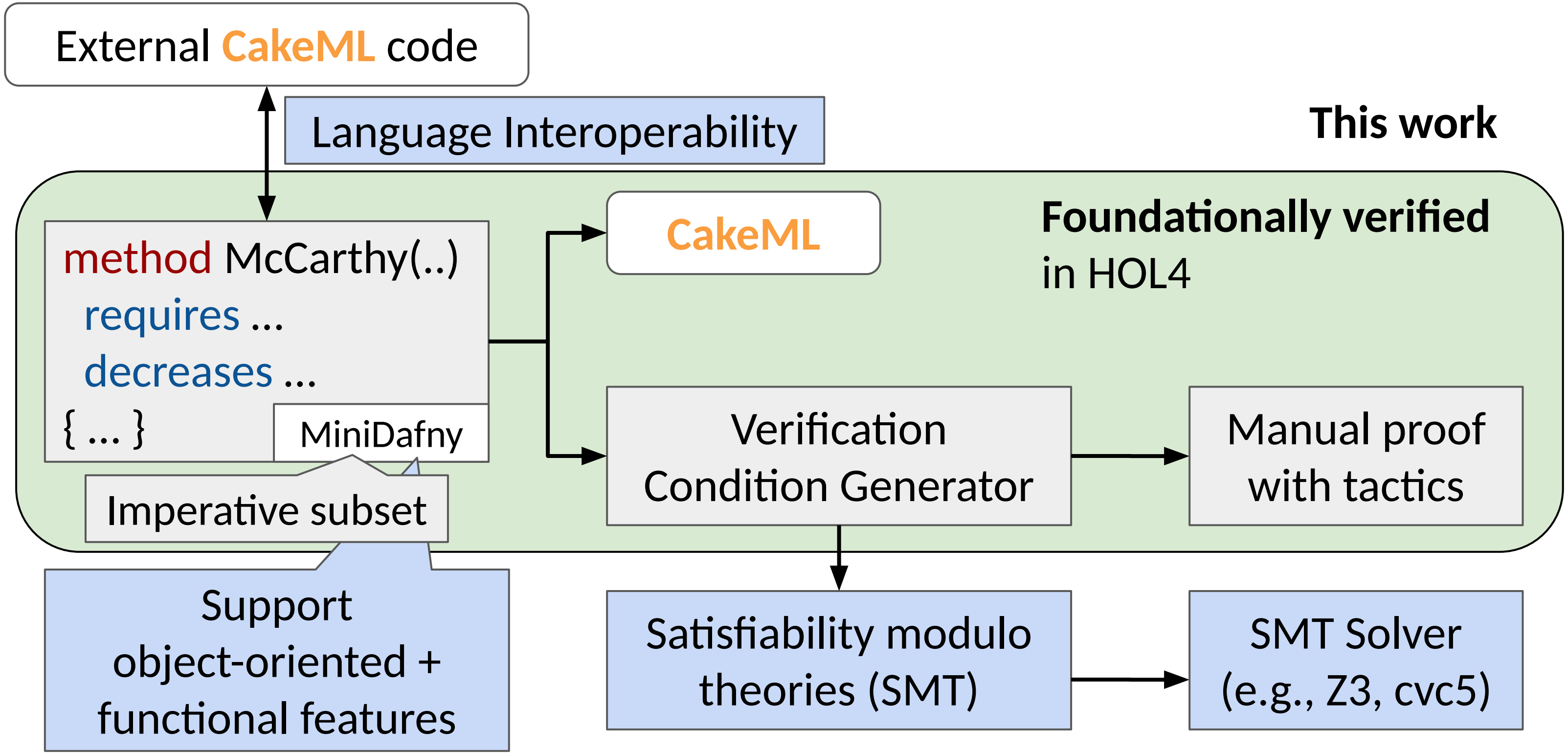
Verified VCG

compile mccarthy = inr *mccarthy_cml* ∧ ... ⇒
 AppReturns (INT *n*) (... [*mccarthy_cml*] ...)
  (INT (if *n* <= 100 then 91 else *n* - 10))

* not presented

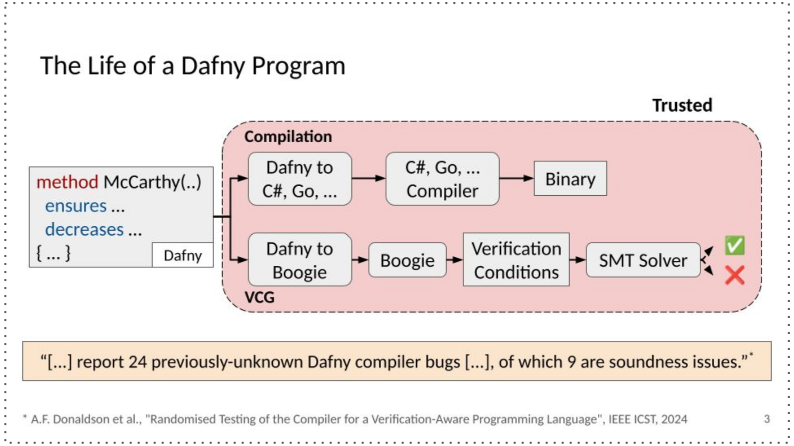
Paper: Section 5 (Putting It All Together)

Conclusion + Future Work



blue = Future Work

Paper: Section 7 (Conclusion)



The Life of a Dafny Program

Trusted

Compilation

```
method McCarthy(..)
  ensures ...
  decreases ...
{ ... }
```

Dafny

Dafny to C#, Go, ... → C#, Go, ... Compiler → Binary

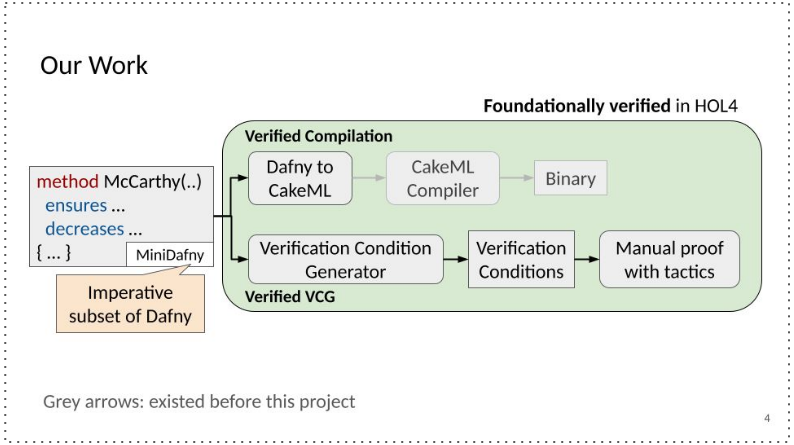
Dafny to Boogie → Boogie → Verification Conditions → SMT Solver → ✅ / ❌

VCG

"[...] report 24 previously-unknown Dafny compiler bugs [...], of which 9 are soundness issues."*

* A.F. Donaldson et al., "Randomised Testing of the Compiler for a Verification-Aware Programming Language", IEEE ICST, 2024

3



Our Work

Foundationally verified in HOL4

Verified Compilation

```
method McCarthy(..)
  ensures ...
  decreases ...
{ ... }
```

MiniDafny

Imperative subset of Dafny

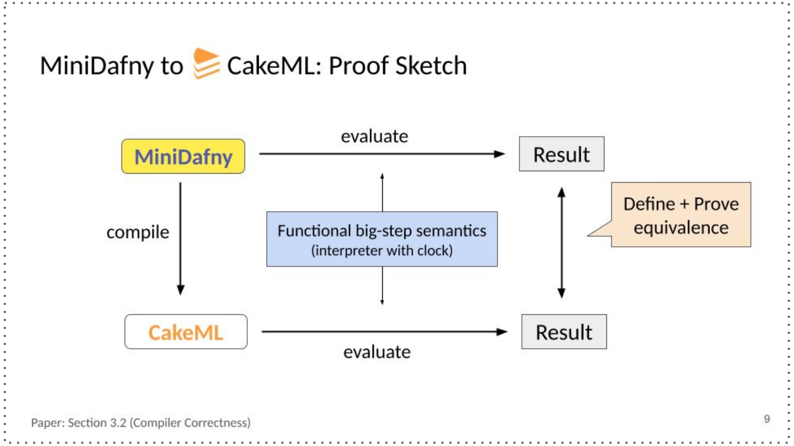
Dafny to CakeML → CakeML Compiler → Binary

Verification Condition Generator → Verification Conditions → Manual proof with tactics

Verified VCG

Grey arrows: existed before this project

4



MiniDafny to CakeML: Proof Sketch

MiniDafny —evaluate→ Result

compile

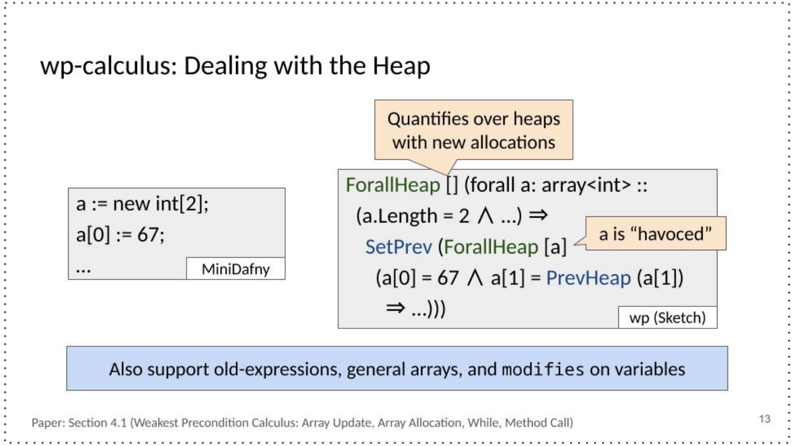
CakeML —evaluate→ Result

Functional big-step semantics
(interpreter with clock)

Define + Prove equivalence

Paper: Section 3.2 (Compiler Correctness)

9



wp-calculus: Dealing with the Heap

```
a := new int[2];
a[0] := 67;
...
```

MiniDafny

Quantifies over heaps with new allocations

```
ForallHeap [] (forall a: array<int> ::
 (a.Length = 2 ∧ ...) ⇒
  SetPrev (ForallHeap [a]
   (a[0] = 67 ∧ a[1] = PrevHeap (a[1])
    ⇒ ...)))
```

a is "havoced"

wp (Sketch)

Also support old-expressions, general arrays, and `modifies` on variables

Paper: Section 4.1 (Weakest Precondition Calculus: Array Update, Array Allocation, While, Method Call)

13

Code: https://github.com/CakeML/cakeml/tree/master/compiler/dafny