Language-Parametric Formal Methods in the Field

Thesis Defense

Daejun Park

December 14, 2018

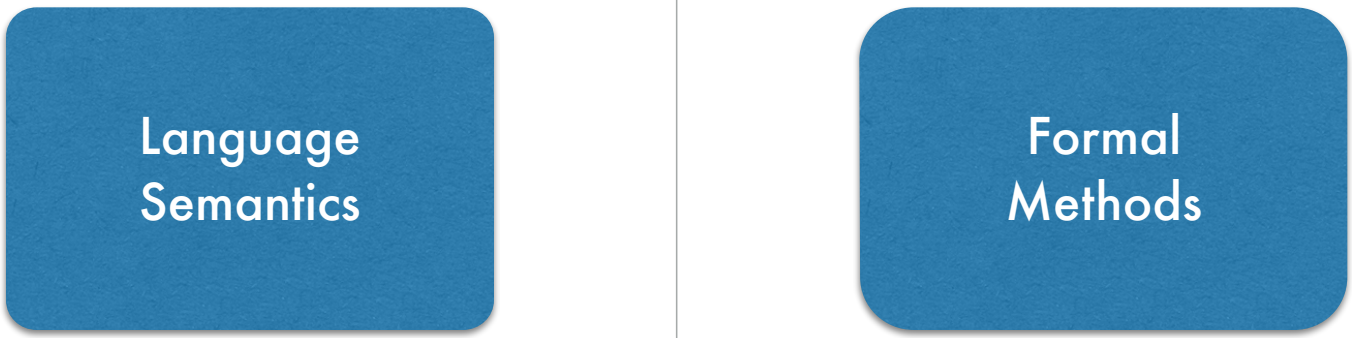
Reusability problem of formal tools

- Fragmentation: crafted for a fixed language
  - Implemented similar heuristics/optimizations: duplicating efforts
- Hard to re-target
  - Replace hardcoded semantics, or
  - Implement a translation to the original language

Language-parametric formal methods

- Mitigate reusability issue
- Develop universal formal methods
  - Parameterized by language semantics
- Instantiate formal tools
  - By plugging-in language semantics
- Admit operational semantics

Language-parametric formal methods



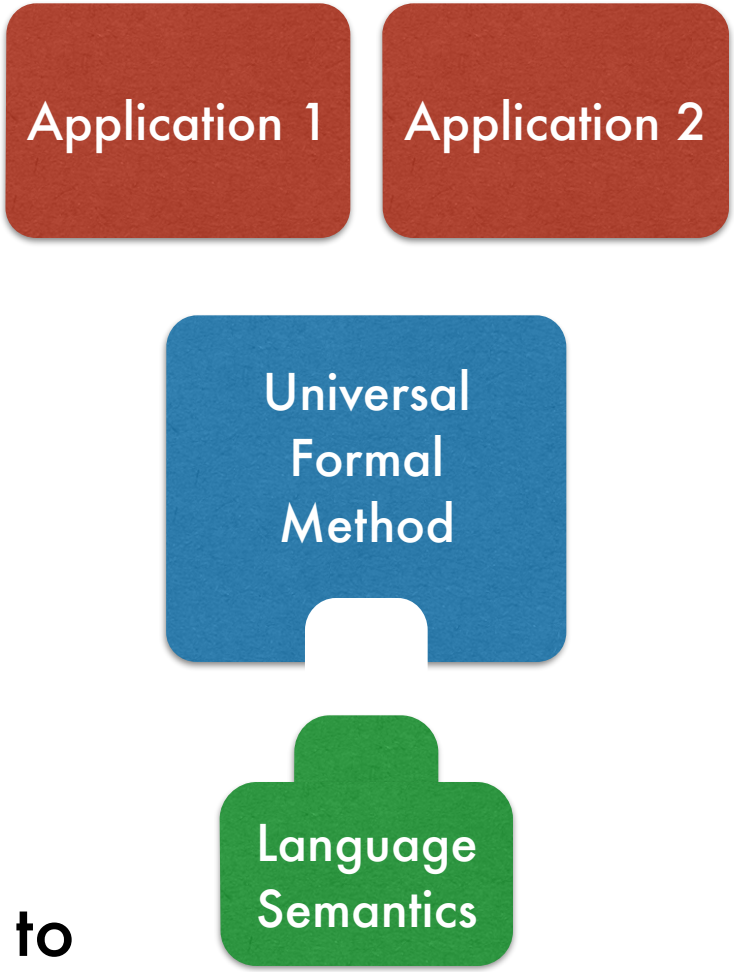
- C (c11, gcc, clang, ...)
- Java (6, 7, 8, ...)
- JavaScript (ES5, ES6, ...)
- ...

- Deductive verification
- Model checking
- Program equivalence checking
- ...

Defined/implemented once, and reused for all others

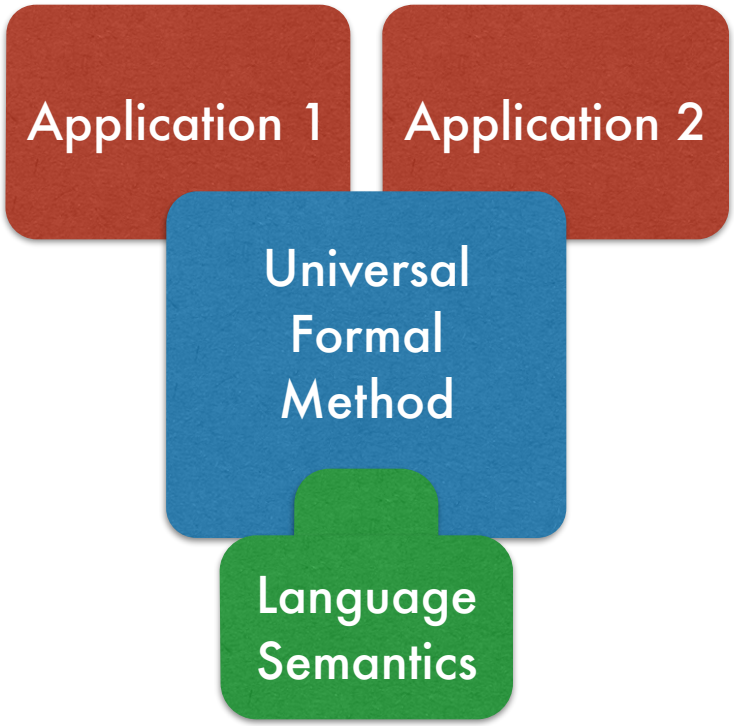
Thesis work

- Specifying real-world language semantics and measuring the specification effort
- Developing language-parametric formal methods
- Instantiating them by plugging-in various language semantics
- Applying the derived formal tools to real-world applications, demonstrating their practical feasibility

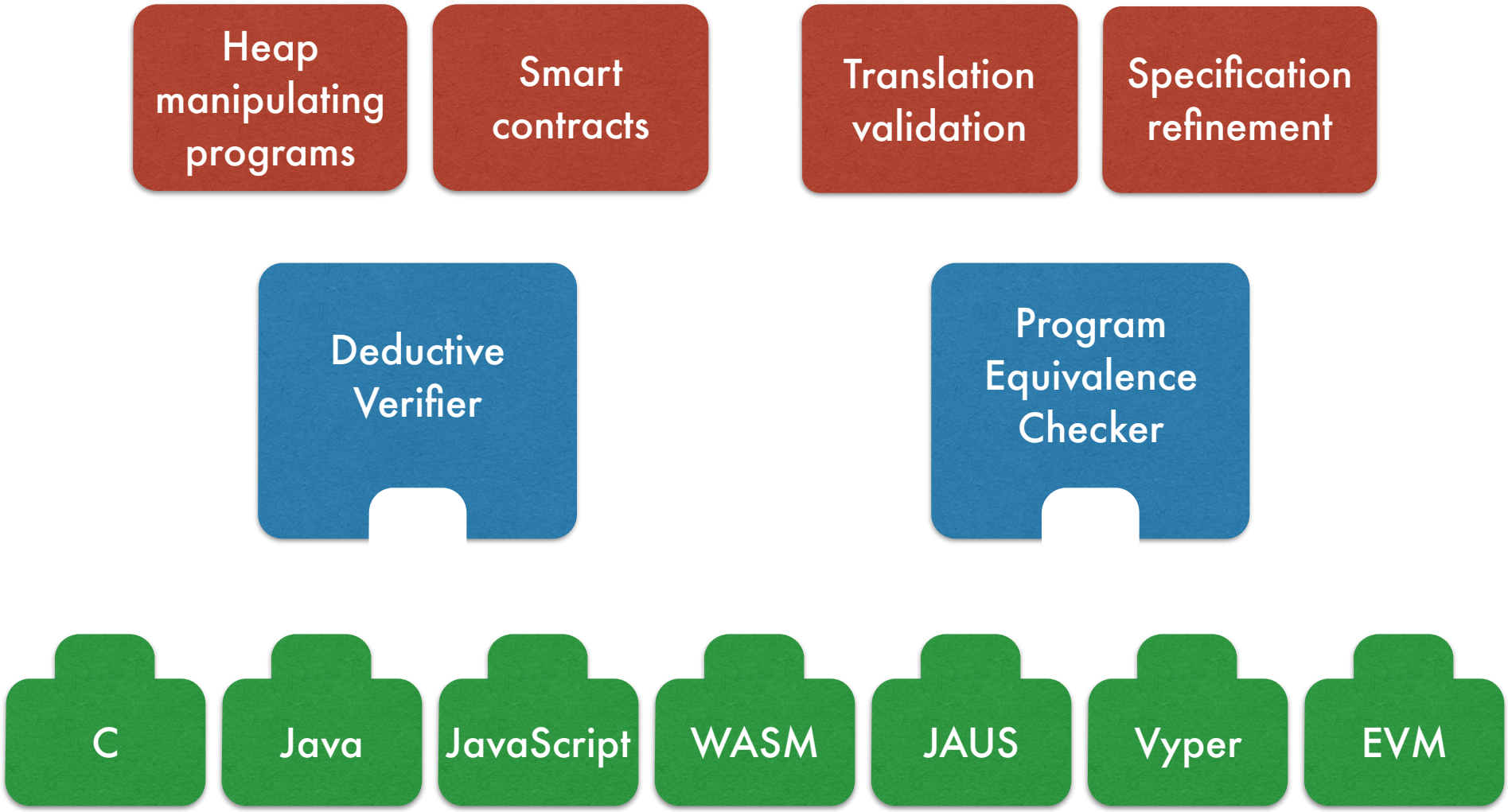


Thesis work

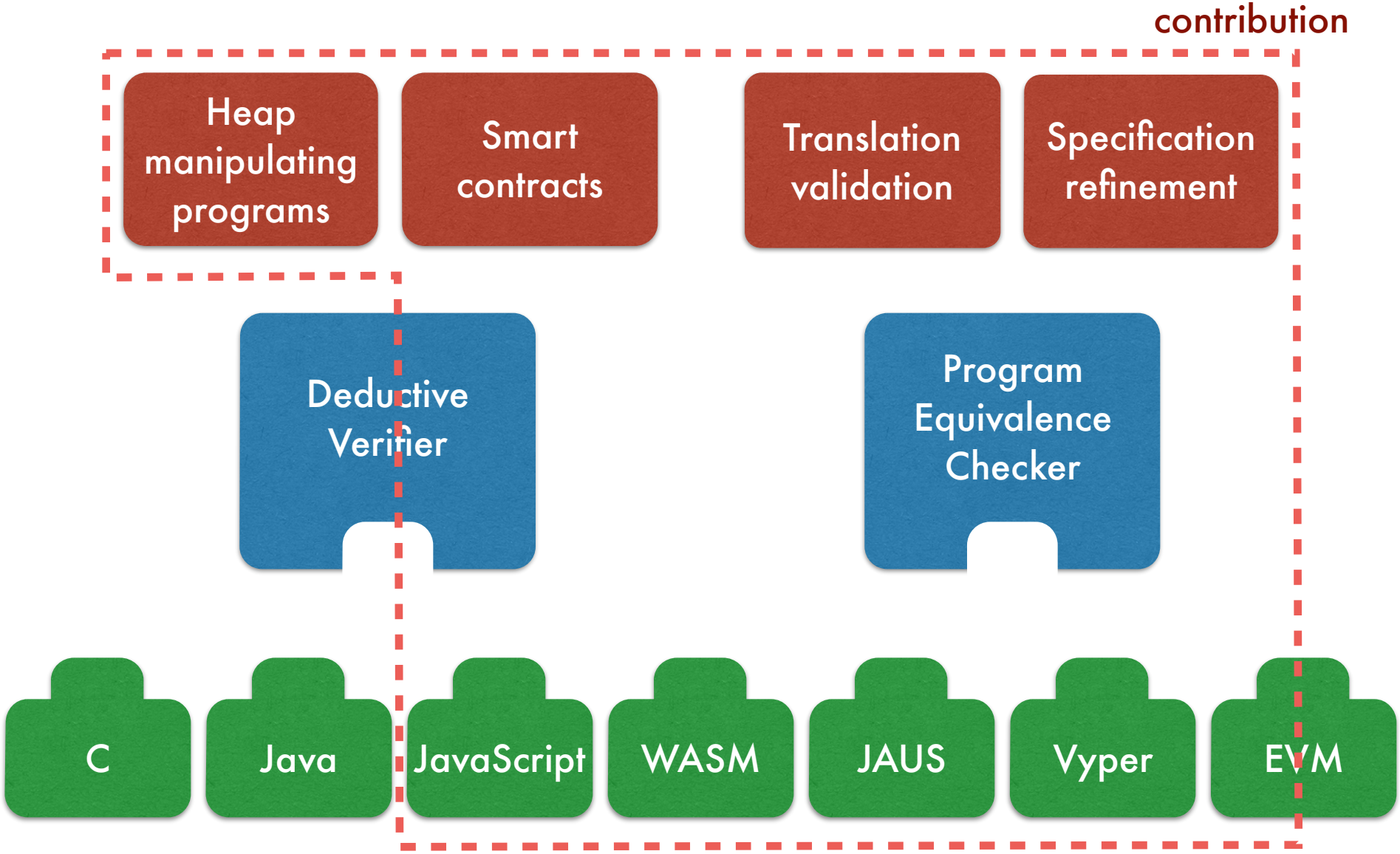
- Specifying real-world language semantics and measuring the specification effort
- Developing language-parametric formal methods
- Instantiating them by plugging-in various language semantics
- Applying the derived formal tools to real-world applications, demonstrating their practical feasibility



Thesis work

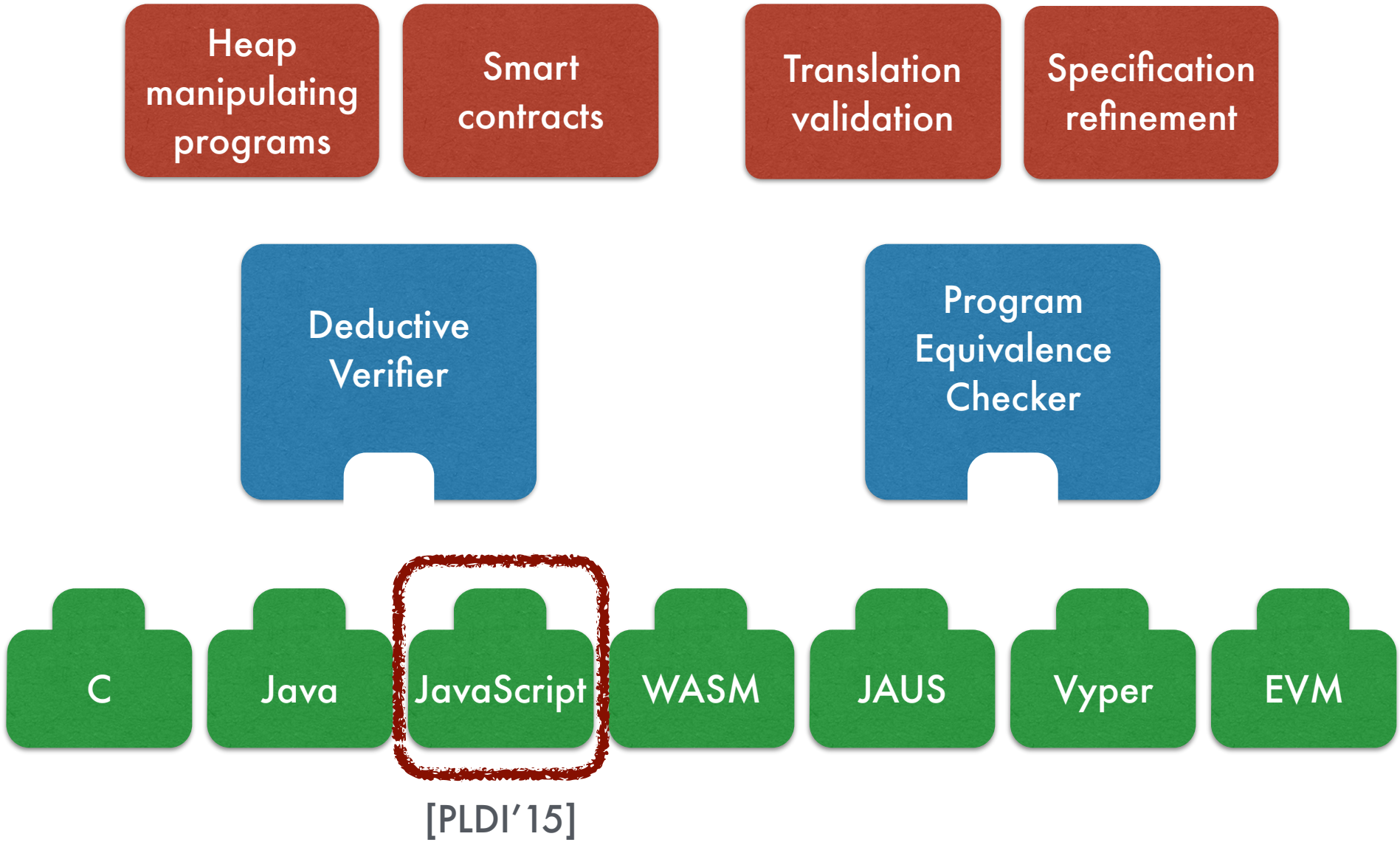

Heap manipulating programs
Smart contracts
Translation validation
Specification refinement
Deductive Verifier
Program Equivalence Checker
C
Java
JavaScript
WASM
JAUS
Vyper
EVM

Thesis work


contribution
Heap manipulating programs
Smart contracts
Translation validation
Specification refinement
Deductive Verifier
Program Equivalence Checker
C
Java
JavaScript
WASM
JAUS
Vyper
EVM

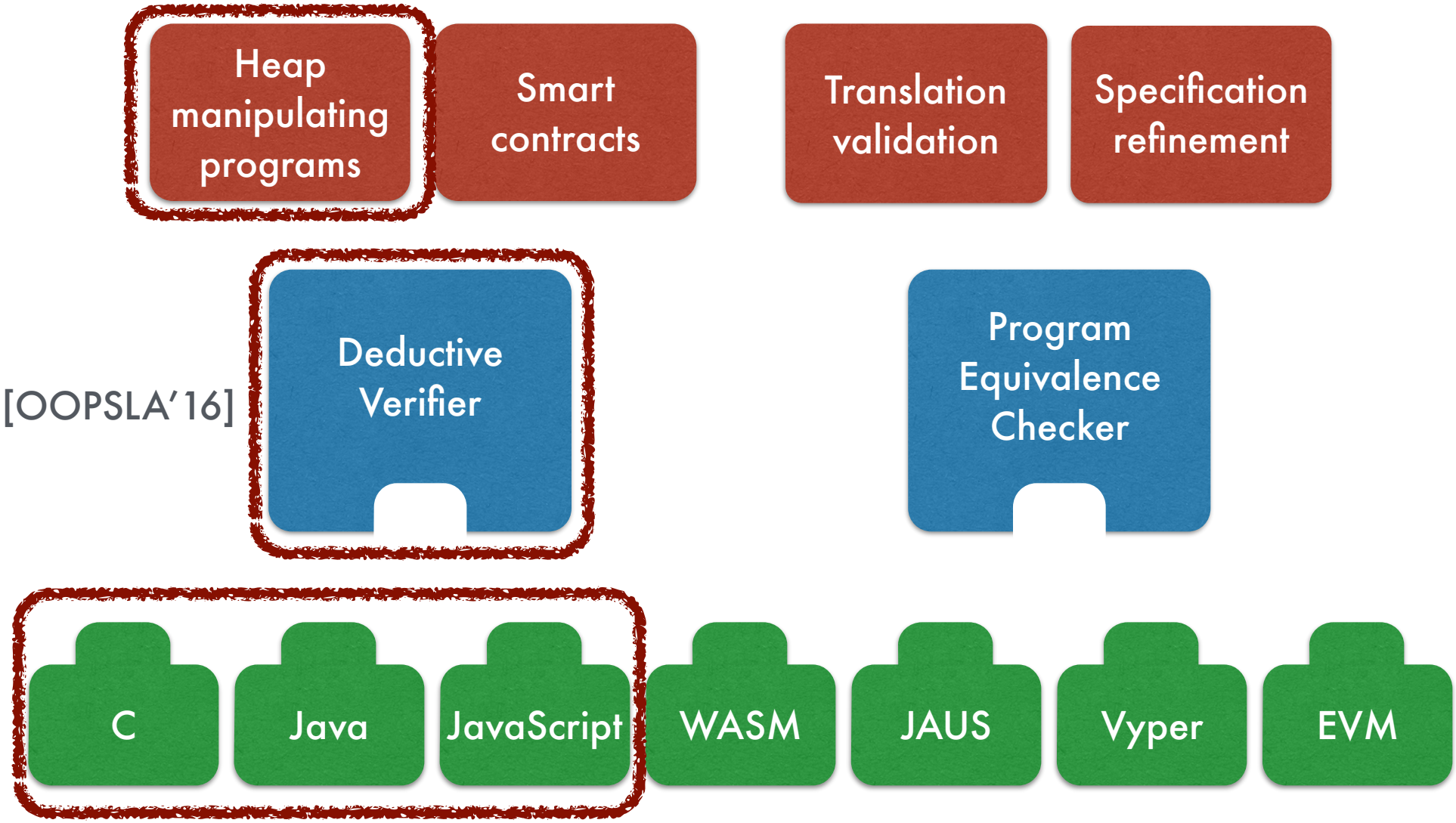
Main Results

Specifying language semantics



[PLDI'15]

Instantiating with various languages



Applying to real-world systems

[FSE'18]

Heap manipulating programs

Smart contracts

Translation validation

Specification refinement

Deductive Verifier

Program Equivalence Checker

C

Java

JavaScript

WASM

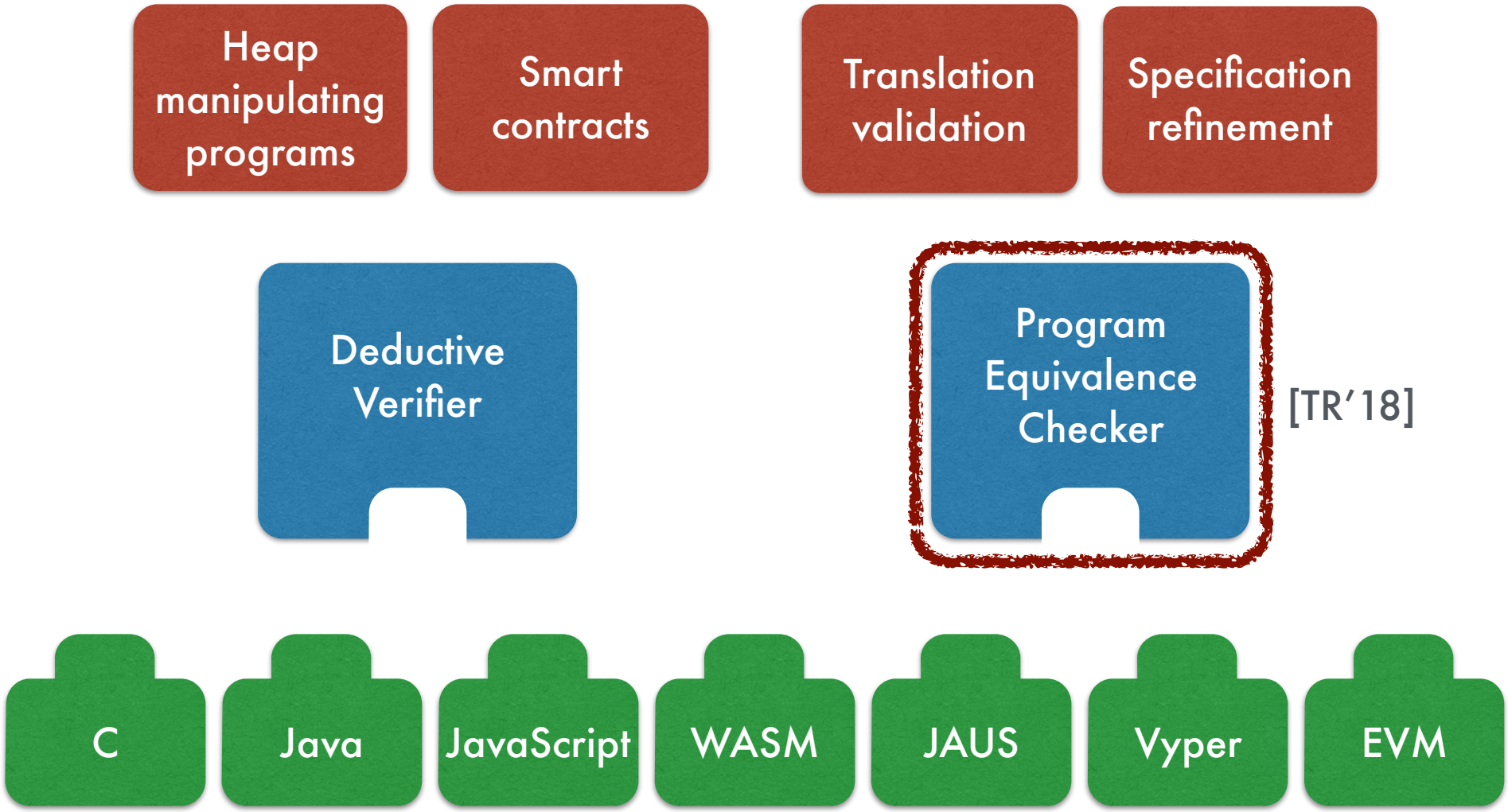
JAUS

Vyper

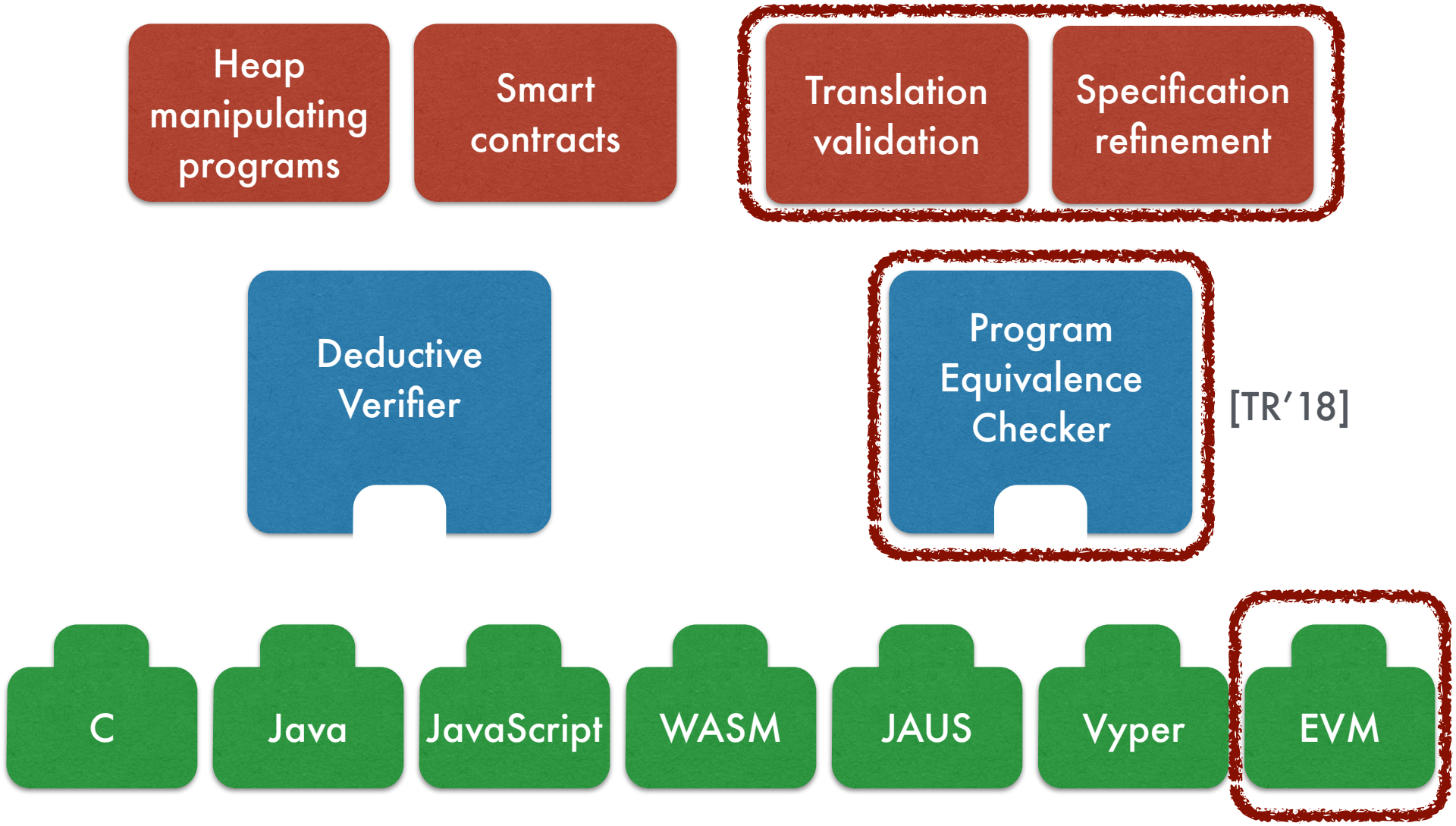
EVM

[CSF'18]

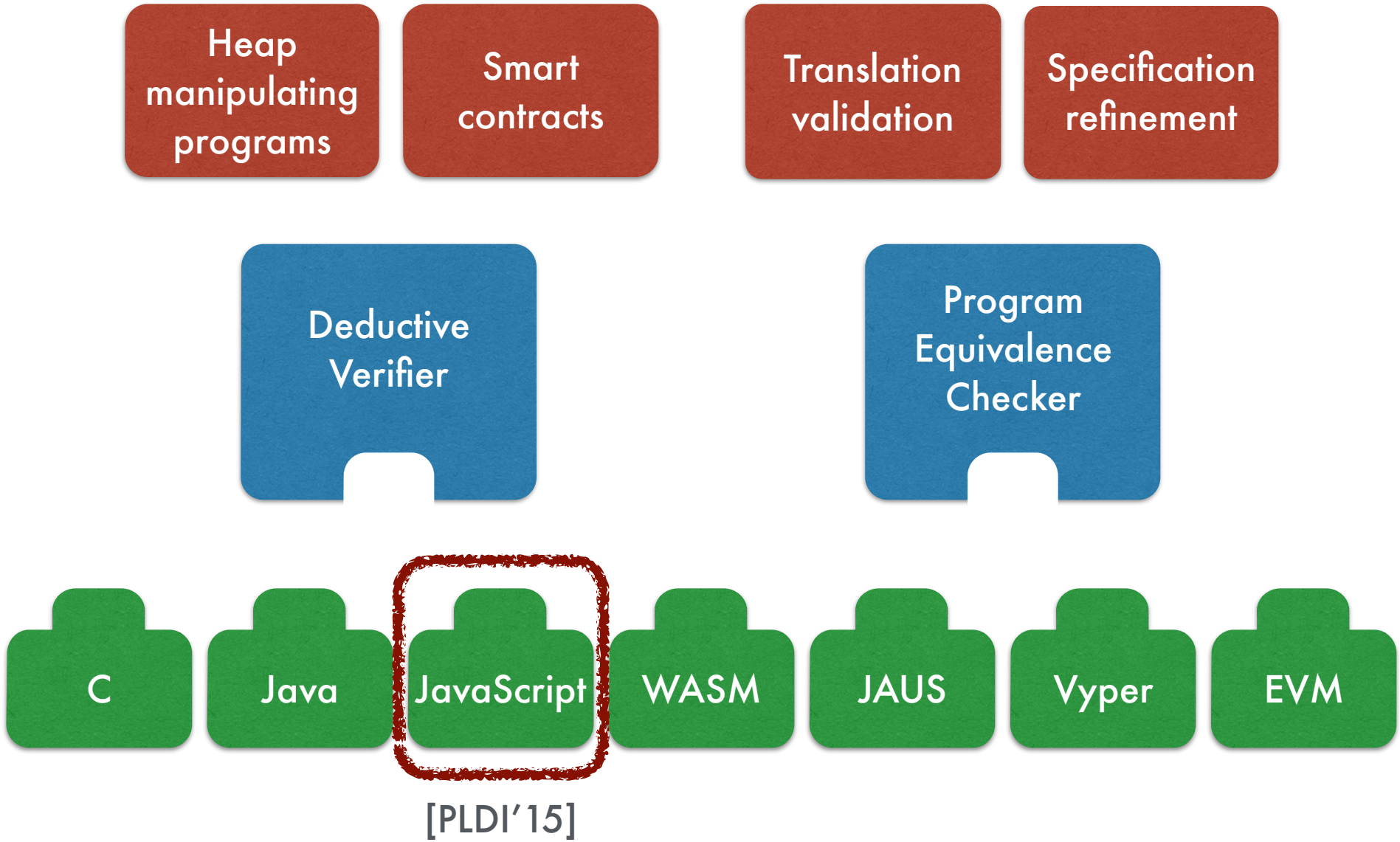
Developing new formal method



Evaluating new formal method

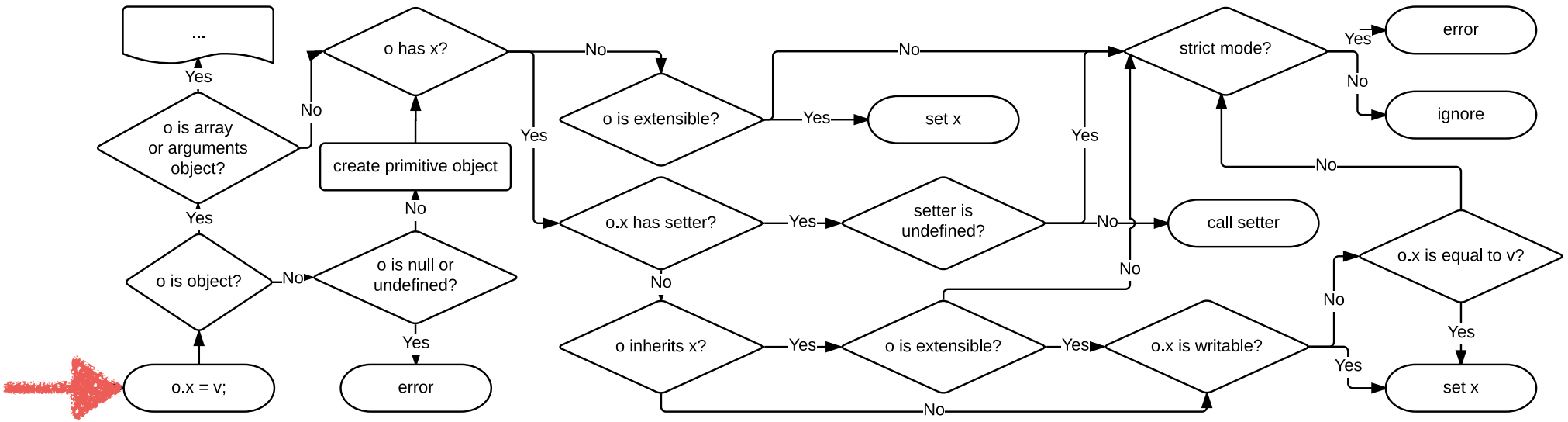


Specifying language semantics



[PLDI'15]

Object Property Update: o.x = v;



Different behaviors depending on whether:

- o is a normal object or not
- o is extensible or not
- x is inherited or not
- x is writable or not
- strict mode or not

KJS: complete formal semantics of JavaScript

KJS *faithfully* formalizes ECMAScript 5.1 (informal) standard.

The expression "++ *Expression*" is evaluated as follows:

1. Let *expr* be the result of evaluating *Expression*.
2. Let *oldValue* be ToNumber(GetValue(*expr*)).
3. Let *newValue* be the result of adding the value `1` to *oldValue*.
4. Call PutValue(*expr*, *newValue*).
5. Return *newValue*.

ECMAScript 5.1 standard

KJS: complete formal semantics of JavaScript

KJS *faithfully* formalizes ECMAScript 5.1 (informal) standard.

The expression "++ *Expression*" is evaluated as follows:

1. Let *expr* be the result of evaluating *Expression*.
2. Let *oldValue* be ToNumber(GetValue(*expr*)).
3. Let *newValue* be the result of adding the value 1 to *oldValue*.
4. Call PutValue(*expr*, *newValue*).
5. Return *newValue*.

ECMAScript 5.1 standard

```
rule ++ Expression =>
 Let $expr = @GetReference(Expression);
 Let $oldValue = ToNumber(GetValue($expr));
 Let $newValue = @Addition($oldValue,1);
 Call PutValue($expr,$newValue);
 Return $newValue;
```

KJS

KJS: complete formal semantics of JavaScript

KJS *faithfully* formalizes ECMAScript 5.1 (informal) standard.

each step of informal description → (systematic translation) → formal pseudo-code statement

The expression "++ *Expression*" is evaluated as follows:

1. Let *expr* be the result of evaluating *Expression*.
2. Let *oldValue* be ToNumber(GetValue(*expr*)).
3. Let *newValue* be the result of adding the value 1 to *oldValue*.
4. Call PutValue(*expr*, *newValue*).
5. Return *newValue*.

ECMAScript 5.1 standard

```
rule ++ Expression =>
 Let $expr = @GetReference(Expression);
 Let $oldValue = ToNumber(GetValue($expr));
 Let $newValue = @Addition($oldValue,1);
 Call PutValue($expr,$newValue);
 Return $newValue;
```

KJS

KJS: complete formal semantics of JavaScript

KJS *faithfully* formalizes ECMAScript 5.1 (informal) standard.

each step of informal description ⟷ formal pseudo-code statement

1-to-1 mapping

systematic translation

The expression "++ *Expression*" is evaluated as follows:

1. Let *expr* be the result of evaluating *Expression*.
2. Let *oldValue* be ToNumber(GetValue(*expr*)).
3. Let *newValue* be the result of adding the value 1 to *oldValue*.
4. Call PutValue(*expr*, *newValue*).
5. Return *newValue*.

```
rule ++ Expression =>
Let $expr = @GetReference(Expression);
Let $oldValue = ToNumber(GetValue($expr));
Let $newValue = @Addition($oldValue,1);
Call PutValue($expr,$newValue);
Return $newValue;
```

ECMAScript 5.1 standard

KJS

manual inspection

Completeness [PLDI'15]

Tested against ECMAScript conformance test suite.

Formal Semantics	Passed	Failed	% passed
KJS	2,782	0	100.0%
[Politz et al. 2012][5]	2,470	345	87.7%
[Bodin et al. 2014]	1,796	986	64.6%

Took me only four months.

Semantic coverage measurement

How many semantic rules are covered by 2,782 core tests?

Semantic coverage measurement

How many semantic rules are covered by 2,782 core tests?

17 rules *never* covered:

Page #	Section # - Step #	KJS	Po	Bo	CR	FF	SF
p35	8.7.1 GetValue (V) - [[Get]], Step 6	○	×	⊗	○	○	○
p36	8.7.2 PutValue (V, W) - [[Put]], Step 2.a	○	○	⊗	○	○	○
p36	8.7.2 PutValue (V, W) - [[Put]], Step 2.b	○	⊗	⊗	○	○	○
p36	8.7.2 PutValue (V, W) - [[Put]], Step 4.a	-	-	-	-	-	-
p36	8.7.2 PutValue (V, W) - [[Put]], Step 4.b	-	-	-	-	-	-
p36	8.7.2 PutValue (V, W) - [[Put]], Step 6.a & 6.b	○	○	⊗	○	○	○
p36	8.7.2 PutValue (V, W) - [[Put]], Step 7.a	○	×	○	○	×	○
p40	8.12.4 [[CanPut]] (P) - Step 8.a	○	⊗	⊗	○	○	○
p53	10.2.1.1.3 SetMutableBinding (N,V,S) - Step 4	○	×	○	×	○	×
p53	10.2.1.1.4 GetBindingValue(N,S) - Step 3.a	-	-	-	-	-	-
p53	10.2.1.1.5 DeleteBinding (N) - Step 2	-	-	-	-	-	-
p54	10.2.1.1.5 DeleteBinding (N) - Step 4 & 5	○	⊗	○	○	○	○
p55	10.2.1.2.4 GetBindingValue(N,S) - Step 4.a	-	-	-	-	-	-
p59	10.5 Declaration Binding Instantiation - Step 5.e.iii.1	○	○	○	○	○	○
p59	10.5 Declaration Binding Instantiation - Step 5.e.iv, 1st condition is true	○	⊗	⊗	○	○	×
p59	10.5 Declaration Binding Instantiation - Step 5.e.iv, 2nd condition is true	○	⊗	⊗	○	○	×
p62	10.6 Arguments Object - [[DefineOwnProperty]], Step 4.a, else-branch	-	-	-	-	-	-

○: Passed ×: Failed ⊗: Not applicable (failed due to unsupported semantics) -: Infeasible semantic behaviors

Po: [Politz et al. 2012] *Bo*: [Bodin et al. 2014] *CR*: Chrome 38.0 (V8 3.28.71) *FF*: Firefox 32.0 (SpiderMonkey 32) *SF*: Safari 7.0.4 (WebKit 537.76.4)

Semantic coverage measurement

How many semantic rules are covered by 2,782 core tests?

17 rules *never* covered:

- 6: shown infeasible → inconsistency of standard

Page #	Section # - Step #	KJS	Po	Bo	CR	FF	SF
p35	8.7.1 GetValue (V) - [[Get]], Step 6	○	×	⊗	○	○	○
p36	8.7.2 PutValue (V, W) - [[Put]], Step 2.a	○	○	⊗	○	○	○
p36	8.7.2 PutValue (V, W) - [[Put]], Step 2.b	○	⊗	⊗	○	○	○
p36	8.7.2 PutValue (V, W) - [[Put]], Step 4.a	-	-	-	-	-	-
p36	8.7.2 PutValue (V, W) - [[Put]], Step 4.b	-	-	-	-	-	-
p36	8.7.2 PutValue (V, W) - [[Put]], Step 6.a & 6.b	○	○	⊗	○	○	○
p36	8.7.2 PutValue (V, W) - [[Put]], Step 7.a	○	×	○	○	×	○
p40	8.12.4 [[CanPut]] (P) - Step 8.a	○	⊗	⊗	○	○	○
p53	10.2.1.1.3 SetMutableBinding (N,V,S) - Step 4	○	×	○	×	○	×
p53	10.2.1.1.4 GetBindingValue(N,S) - Step 3.a	-	-	-	-	-	-
p53	10.2.1.1.5 DeleteBinding (N) - Step 2	-	-	-	-	-	-
p54	10.2.1.1.5 DeleteBinding (N) - Step 4 & 5	○	⊗	○	○	○	○
p55	10.2.1.2.4 GetBindingValue(N,S) - Step 4.a	-	-	-	-	-	-
p59	10.5 Declaration Binding Instantiation - Step 5.e.iii.1	○	○	○	○	○	○
p59	10.5 Declaration Binding Instantiation - Step 5.e.iv, 1st condition is true	○	⊗	⊗	○	○	×
p59	10.5 Declaration Binding Instantiation - Step 5.e.iv, 2nd condition is true	○	⊗	⊗	○	○	×
p62	10.6 Arguments Object - [[DefineOwnProperty]], Step 4.a, else-branch	-	-	-	-	-	-

○: Passed ×: Failed ⊗: Not applicable (failed due to unsupported semantics) -: Infeasible semantic behaviors

Po: [Politz et al. 2012] *Bo*: [Bodin et al. 2014] *CR*: Chrome 38.0 (V8 3.28.71) *FF*: Firefox 32.0 (SpiderMonkey 32) *SF*: Safari 7.0.4 (WebKit 537.76.4)

Semantic coverage measurement

How many semantic rules are covered by 2,782 core tests?

17 rules *never* covered:

- 6: shown infeasible → inconsistency of standard
- 11: wrote new tests → found bugs in JS engines

Page #	Section # - Step #	KJS	Po	Bo	CR	FF	SF
p35	8.7.1 GetValue (V) - [[Get]], Step 6	○	×	⊗	○	○	○
p36	8.7.2 PutValue (V, W) - [[Put]], Step 2.a	○	○	⊗	○	○	○
p36	8.7.2 PutValue (V, W) - [[Put]], Step 2.b	○	⊗	⊗	○	○	○
p36	8.7.2 PutValue (V, W) - [[Put]], Step 4.a	-	-	-	-	-	-
p36	8.7.2 PutValue (V, W) - [[Put]], Step 4.b	-	-	-	-	-	-
p36	8.7.2 PutValue (V, W) - [[Put]], Step 6.a & 6.b	○	○	⊗	○	○	○
p36	8.7.2 PutValue (V, W) - [[Put]], Step 7.a	○	×	○	○	×	○
p40	8.12.4 [[CanPut]] (P) - Step 8.a	○	⊗	⊗	○	○	○
p53	10.2.1.1.3 SetMutableBinding (N,V,S) - Step 4	○	×	○	×	○	×
p53	10.2.1.1.4 GetBindingValue(N,S) - Step 3.a	-	-	-	-	-	-
p53	10.2.1.1.5 DeleteBinding (N) - Step 2	-	-	-	-	-	-
p54	10.2.1.1.5 DeleteBinding (N) - Step 4 & 5	○	⊗	○	○	○	○
p55	10.2.1.2.4 GetBindingValue(N,S) - Step 4.a	-	-	-	-	-	-
p59	10.5 Declaration Binding Instantiation - Step 5.e.iii.1	○	○	○	○	○	○
p59	10.5 Declaration Binding Instantiation - Step 5.e.iv, 1st condition is true	○	⊗	⊗	○	○	×
p59	10.5 Declaration Binding Instantiation - Step 5.e.iv, 2nd condition is true	○	⊗	⊗	○	○	×
p62	10.6 Arguments Object - [[DefineOwnProperty]], Step 4.a, else-branch	-	-	-	-	-	-

○: Passed ×: Failed ⊗: Not applicable (failed due to unsupported semantics) -: Infeasible semantic behaviors

Po: [Politz et al. 2012] *Bo*: [Bodin et al. 2014] *CR*: Chrome 38.0 (V8 3.28.71) *FF*: Firefox 32.0 (SpiderMonkey 32) *SF*: Safari 7.0.4 (WebKit 537.76.4)

Semantic coverage measurement

How many semantic rules are covered by 2,782 core tests?

17 rules *never* covered:

- 6: shown infeasible → inconsistency of standard
- 11: wrote new tests → found bugs in JS engines

Page #	Section # - Step #	KJS	Po	Bo	CR	FF	SF
p35	8.7.1 GetValue (V) - [[Get]], Step 6	○		⊗	○	○	○
p36	8.7.2 PutValue (V, W) - [[Put]], Step 2.a	○	○	⊗	○	○	○
p36	8.7.2 PutValue (V, W) - [[Put]], Step 2.b	○	⊗	⊗	○	○	○
p36	8.7.2 PutValue (V, W) - [[Put]], Step 4.a	-	-	-	-	-	-
p36	8.7.2 PutValue (V, W) - [[Put]], Step 4.b	-	-	-	-	-	-
p36	8.7.2 PutValue (V, W) - [[Put]], Step 6.a & 6.b	○	○	⊗	○	○	○
p36	8.7.2 PutValue (V, W) - [[Put]], Step 7.a	○		○	○		○
p40	8.12.4 [[CanPut]] (P) - Step 8.a	○	⊗	⊗	○	○	○
p53	10.2.1.1.3 SetMutableBinding (N,V,S) - Step 4	○		○		○	
p53	10.2.1.1.4 GetBindingValue(N,S) - Step 3.a	-	-	-	-	-	-
p53	10.2.1.1.5 DeleteBinding (N) - Step 2	-	-	-	-	-	-
p54	10.2.1.1.5 DeleteBinding (N) - Step 4 & 5	○	⊗	○	○	○	○
p55	10.2.1.2.4 GetBindingValue(N,S) - Step 4.a	-	-	-	-	-	-
p59	10.5 Declaration Binding Instantiation - Step 5.e.iii.1	○	○	○	○	○	
p59	10.5 Declaration Binding Instantiation - Step 5.e.iv, 1st condition is true	○	⊗	⊗	○	○	
p59	10.5 Declaration Binding Instantiation - Step 5.e.iv, 2nd condition is true	○	⊗	⊗	○	○	
p62	10.6 Arguments Object - [[DefineOwnProperty]], Step 4.a, else branch	-	-	-	-	-	-

○: Passed ×: Failed ⊗: Not applicable (failed due to unsupported semantics) -: Infeasible semantic behaviors

Po: [Politz et al. 2012] *Bo*: [Bodin et al. 2014] *CR*: Chrome 38.0 (V8 3.28.71) *FF*: Firefox 32.0 (SpiderMonkey 32) *SF*: Safari 7.0.4 (WebKit 537.76.4)

Semantic coverage measurement

How many semantic rules are covered by 2,782 core tests?

17 rules *never* covered:

- 6: shown infeasible → inconsistency of standard
- 11: wrote new tests → found bugs in JS engines

Page #	Section # - Step #	KJS	Po	Bo	CR	FF	SF
p35	8.7.1 GetValue (V) - [[Get]], Step 6	○		⊗	○	○	○
p36	8.7.2 PutValue (V, W) - [[Put]], Step 2.a	○	○	⊗	○	○	○
p36	8.7.2 PutValue (V, W) - [[Put]], Step 2.b	○	⊗	⊗	○	○	○
p36	8.7.2 PutValue (V, W) - [[Put]], Step 4.a	-	-	-	-	-	-
p36	8.7.2 PutValue (V, W) - [[Put]], Step 4.b	-	-	-	-	-	-
p36	8.7.2 PutValue (V, W) - [[Put]], Step 6.a & 6.b	○	○	⊗	○	○	○
p36	8.7.2 PutValue (V, W) - [[Put]], Step 7.a	○		○	○		○
p40	8.12.4 [[CanPut]] (P) - Step 8.a	○	⊗	⊗	○	○	○
p53	10.2.1.1.3 SetMutableBinding (N,V,S) - Step 4	○		○		○	
p53	10.2.1.1.4 GetBindingValue(N,S) - Step 3.a	-	-	-	-	-	-
p53	10.2.1.1.5 DeleteBinding (N) - Step 2	-	-	-	-	-	-
p54	10.2.1.1.5 DeleteBinding (N) - Step 4 & 5	○	⊗	○	○	○	○
p55	10.2.1.2.4 GetBindingValue(N,S) - Step 4.a	-	-	-	-	-	-
p59	10.5 Declaration Binding Instantiation - Step 5.e.iii.1	○	○	○	○	○	○
p59	10.5 Declaration Binding Instantiation - Step 5.e.iv, 1st condition is true	○	⊗	⊗	○	○	
p59	10.5 Declaration Binding Instantiation - Step 5.e.iv, 2nd condition is true	○	⊗	⊗	○	○	
p62	10.6 Arguments Object - [[DefineOwnProperty]], Step 4.a, else branch	-	-	-	-	-	-

○: Passed ×: Failed ⊗: Not applicable (failed due to unsupported semantics) -: Infeasible semantic behaviors

Po: [Politz et al. 2012] *Bo*: [Bodin et al. 2014] *CR*: Chrome 38.0 (V8 3.28.71) *FF*: Firefox 32.0 (SpiderMonkey 32) *SF*: Safari 7.0.4 (WebKit 537.76.4)

Bugs found in JavaScript engines

Chrome 38.0 and Safari 7.0.4 failed to conform to standard.
Fixed in Chrome 41.0 and Safari 8.0.6

```
"use strict";
var _ = function f() {
          f = 0;
        };
```

← runtime error in Firefox, but silently ignored in Chrome and Safari.

Bugs found in JavaScript engines

Chrome 38.0 and Safari 7.0.4 failed to conform to standard.
Fixed in Chrome 41.0 and Safari 8.0.6

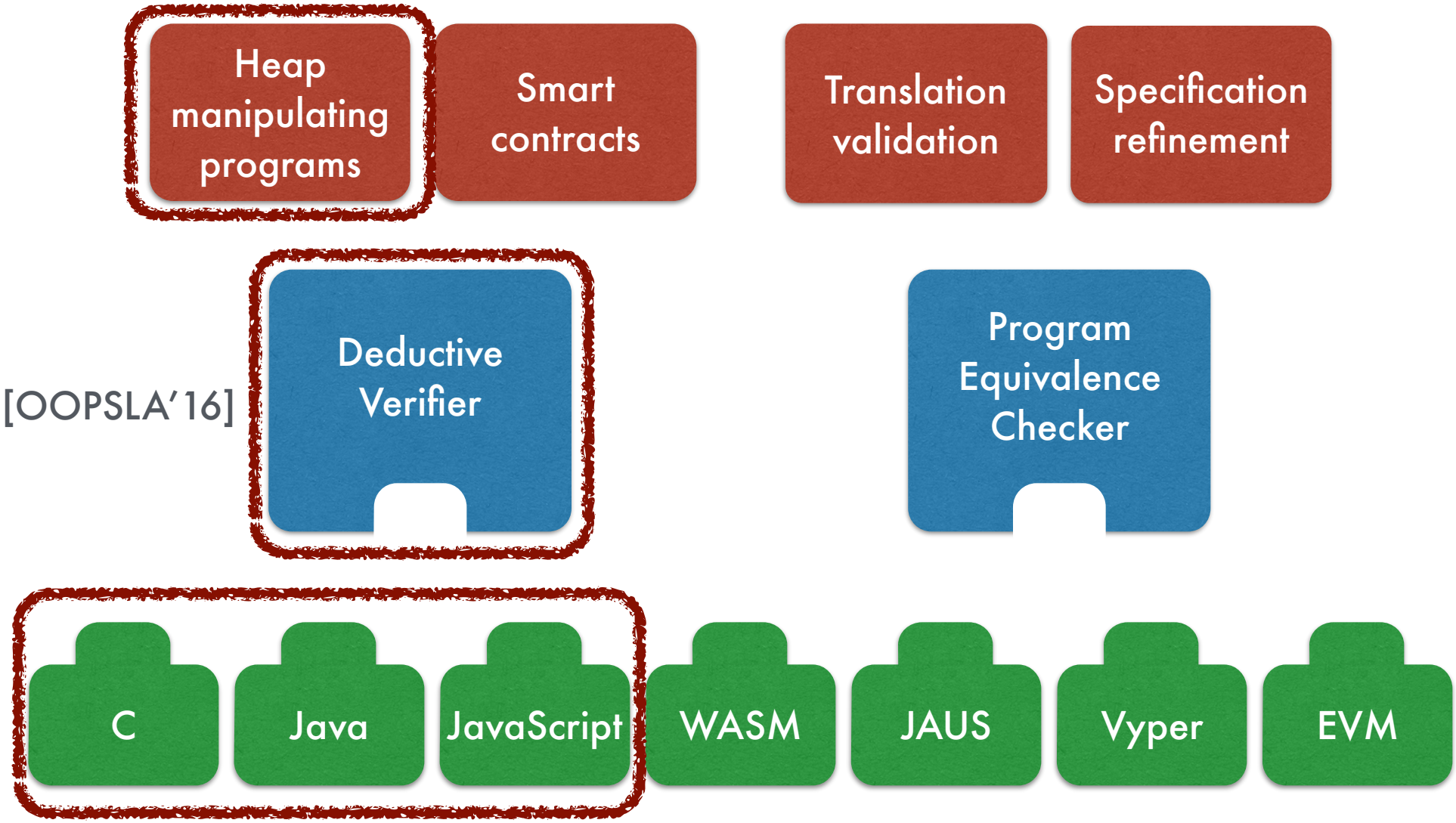
```
"use strict";
var _ = function f() {
          f = 0;
        };
```

← runtime error in Firefox, but silently ignored in Chrome and Safari.

Compare:

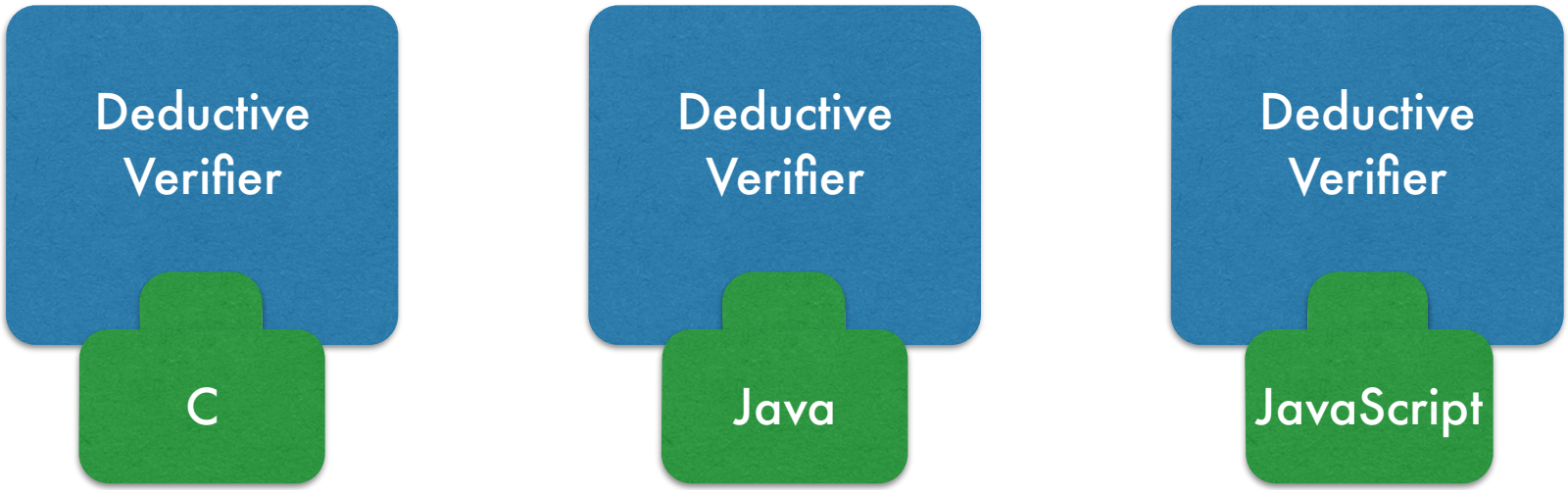
```
"use strict";
        function f() {
                  ← here, f is this function
          f = 0;  ← no error
                  ← here, f is 0
        }
```

Instantiating with various languages



Deductive program verifiers

- Instantiated framework by plugging-in three language semantics.



- Verified challenging heap-manipulating programs implementing the same algorithms in all three languages.

Experiments [OOPSLA'16]

Time (secs)

Programs	C	Java	JS
BST find	14.0	4.7	6.3
BST insert	30.2	8.6	8.2
BST delete	71.7	24.9	21.2
AVL find	13.0	4.9	6.4
AVL insert	281.3	105.2	135.0
AVL delete	633.7	271.6	239.6
RBT find	14.5	5.0	6.8
RBT insert	903.8	115.6	114.5
RBT delete	1,902.1	171.2	183.6

Programs	C	Java	JS
Treap find	14.4	4.9	6.5
Treap insert	67.7	23.1	18.9
Treap delete	90.4	28.4	33.2
List reverse	11.4	4.1	5.5
List append	14.8	7.3	5.3
Bubble sort	66.4	38.8	31.3
Insertion sort	61.9	31.1	44.8
Quick sort	79.2	47.1	48.1
Merge sort	170.6	87.0	66.0
Total	4,441.1	983.5	981.2
Average	246.7	54.6	54.5

Experiments [OOPSLA'16]

Time (secs)

Programs	C	Java	JS
BST find	14.0	4.7	6.3
BST insert	30.2	8.6	8.2

Programs	C	Java	JS
Treap find	14.4	4.9	6.5
Treap insert	67.7	23.1	18.9

Full functional correctness:

```
/*
 * @pre  bst(t)
 * @post bst(t')
 * @post keys(t') == keys(t) \union { v }
 */
function insert(t, v) {
  ...
}
```

Total	4,441.1	983.5	981.2
Average	246.7	54.6	54.5

Experiments [OOPSLA'16]

Time (secs)

Programs	C	Java	JS
BST find	14.0	4.7	6.3
BST insert	30.2	8.6	8.2
BST delete	71.7	24.9	21.2
AVL find	13.0	4.9	6.4
AVL insert	281.3	105.2	135.0

Programs	C	Java	JS
Treap find	14.4	4.9	6.5
Treap insert	67.7	23.1	18.9
Treap delete	90.4	28.4	33.2
List reverse	11.4	4.1	5.5
List append	14.8	7.3	5.3

Performance is comparable to a state-of-the-art verifier for C, VCDryad, based on a separation logic extension of VCC:
e.g., AVL insert : 260s vs 280s (ours)

Total	4,441.1	983.5	981.2
Average	246.7	54.6	54.5

Applying to real-world systems

[FSE'18]

Heap manipulating programs

Smart contracts

Translation validation

Specification refinement

Deductive Verifier

Program Equivalence Checker

C

Java

JavaScript

WASM

JAUS

Vyper

EVM

[CSF'18]

Smart contracts

- Programs that run on blockchain
- Usually written in a high-level language
  - Solidity (JavaScript-like), Vyper (Python-like), …
- Compiled down to VM bytecode
  - EVM (Ethereum VM), IELE (LLVM-like VM), …

  - Runs on VM of blockchain nodes

Smart contract example

```
function transfer(address from,
                  address to,
                  uint256 value) returns (bool) {

    if ( balances[from] >= value ) {
        balances[to] =+ value;
        balances[from] -= value;

        return true;
    } else {
        return false;
    }
}
```

Smart contract example

```
function transfer(address from,
                  address to,
                  uint256 value) returns (bool) {

    if ( balances[from] >= value ) {
        balances[to] =+ value;
        balances[from] -= value;

        return true;
    } else {
        return false;
    }
}
```

Smart contract example

```
function transfer(address from,
                  address to,
                  uint256 value) returns (bool) {

    if ( balances[from] >= value ) {
        balances[to] =+ value;
        balances[from] -= value;

        return true;
    } else {
        return false;
    }
}
```

‘=+’ vs ‘+=’

* ETHNews.com, “Ether.Camp’s HKG Token Has A Bug And Needs To Be Reissued”

Smart contract example

```
function transfer(address from,
                  address to,
                  uint256 value) returns (bool) {

    if ( balances[from] >= value ) {
        balances[to] = (+value);
        balances[from] -= value;

        return true;
    } else {
        return false;
    }
}
```

* ETHNews.com, “Ether.Camp’s HKG Token Has A Bug And Needs To Be Reissued”

Smart contract example

```
function transfer(address from,
                  address to,
                  uint256 value) returns (bool) {

    if ( balances[from] >= value ) {
        balances[to] = value;
        balances[from] -= value;

        return true;
    } else {
        return false;
    }
}
```

* ETHNews.com, “Ether.Camp’s HKG Token Has A Bug And Needs To Be Reissued”

Smart contract example

```
function transfer(address from,
                  address to,
                  uint256 value) returns (bool) {

    if ( balances[from] >= value ) {
        balances[to] += value;
        balances[from] -= value;

        return true;
    } else {
        return false;
    }
}
```

Smart contract example

```
function transfer(address from,
                  address to,
                  uint256 value) returns (bool) {

    if ( balances[from] >= value ) {
        balances[to] += value;
        balances[from] -= value;

        return true;
    } else {
        return false;
    }
}
```

Smart contract example

```
function transfer(address from,
                  address to,
                  uint256 value) returns (bool) {

    if ( balances[from] >= value ) {
        balances[to] += value;
        balances[from] -= value;

        return true;
    } else {
        return false;
    }
}
```

arithmetic overflow

Smart contract example

```
function transfer(address from,
                  address to,
                  uint256 value) returns (bool) {

    if ( balances[from] >= value ) {
        balances[to] = SafeMath.add(balances[to], value);
        balances[from] -= value;

        return true;
    } else {
        return false;
    }
}
```

will throw if overflow

Smart contract example

```
function transfer(address from,
                  address to,
                  uint256 value) returns (bool) {

    if ( balances[from] >= value ) {
        balances[to] = SafeMath.add(balances[to], value);
        balances[from] -= value;

        return true;
    } else {
        return false;
    }
}
```

Smart contract example

```
function transfer(address from,
                  address to,
                  uint256 value) returns (bool) {

    if ( balances[from] >= value ) {
        balances[to] = SafeMath.add(balances[to], value);
        balances[from] -= value;

        return true;
    } else {
        return false;
    }
}
```

self-transfer may fail

Smart contract example

```
function transfer(address from,
                  address to,
                  uint256 value) returns (bool) {

    if ( balances[from] >= value ) {

        balances[from] -= value;
        balances[to] = SafeMath.add(balances[to], value);
        return true;
    } else {
        return false;
    }
}
```

more robust

Why bytecode?

address: 0x01

```
contract GoodToken {
  function transfer() {
    return true;
  }
}
```

address: 0x02

```
contract BadToken {
  function transfer() { }
}
```

Why bytecode?

```
interface Token {
  function transfer() returns (bool);
}

contract Wallet {
  function transfer(address token) {
    return Token(token).transfer();
  }
}
```

address: 0x01

```
contract GoodToken {
  function transfer() {
    return true;
  }
}
```

address: 0x02

```
contract BadToken {
  function transfer() { }
}
```

Why bytecode?

```
interface Token {
  function transfer() returns (bool);
}

contract Wallet {
  function transfer(address token) {
    return Token(token).transfer();
  }
}
```

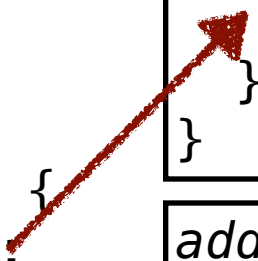
if token = *0x01*

address: 0x01

```
contract GoodToken {
  function transfer() {
    return true;
  }
}
```

address: 0x02

```
contract BadToken {
  function transfer() { }
}
```



Why bytecode?

```
interface Token {
  function transfer() returns (bool);
}

contract Wallet {
  function transfer(address token) {
    return Token(token).transfer();
  }
}
```

if token = *0x02*

address: 0x01

```
contract GoodToken {
  function transfer() {
    return true;
  }
}
```

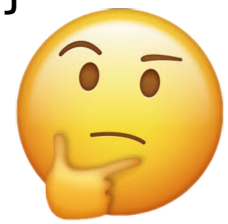
address: 0x02

```
contract BadToken {
  function transfer() { }
}
```

Why bytecode?

```
interface Token {
  function transfer() returns (bool);
}

contract Wallet {
  function transfer(address token) {
    return Token(token).transfer();
  }
}
```



if token = *0x02*

address: 0x01

```
contract GoodToken {
  function transfer() {
    return true;
  }
}
```

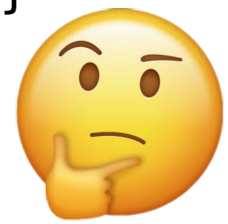
address: 0x02

```
contract BadToken {
  function transfer() { }
}
```

Why bytecode?

```
interface Token {
  function transfer() returns (bool);
}

contract Wallet {
  function transfer(address token) {
    return Token(token).transfer();
  }
}
```



if token = *0x02*

address: 0x01

```
contract GoodToken {
  function transfer() {
    return true;
  }
}
```

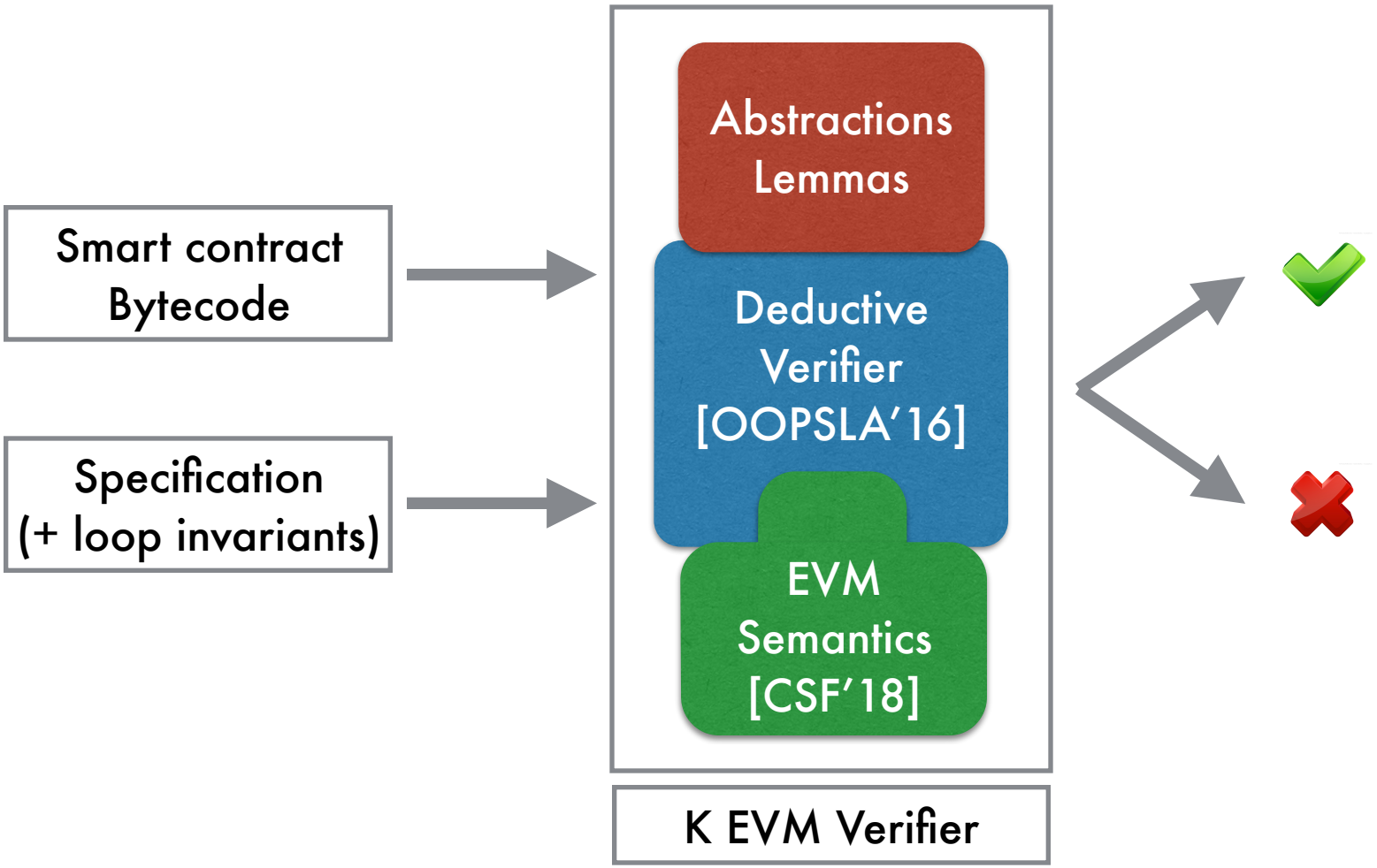
address: 0x02

```
contract BadToken {
  function transfer() { }
}
```

- *Return true* in Solidity 0.4.21 or earlier
- *Revert* in Solidity 0.4.22 or later (latest: 0.5.1)

* Lukas Cremer, "Missing return value bug—At least 130 tokens affected"

K EVM Verifier [FSE'18]



Specification example

[transfer-success]

callData:
  #abiCallData("**transfer**",
      #address(**FROM**), #address(**TO**), #uint256(**VALUE**))

storage:
  #(**BALANCES[FROM]**) ⟼ (BAL_FROM ⟹ **BAL_FROM - VALUE**)
  #(**BALANCES[TO]**) ⟼ (BAL_TO ⟹ **BAL_TO + VALUE**)

requires:
  FROM ≠ TO
  VALUE ≤ BAL_FROM
  BAL_TO + VALUE < (2 ^ 256)

true

output:
  _ ⟹ #asByteArray(**1**, 32)

statusCode:
  _ ⟹ EVMC_SUCCESS

```
function transfer(address from,
                  address to,
                  uint256 value) returns (bool) {

    if ( balances[from] >= value ) {

        balances[from] -= value;
        balances[to] = SafeMath.add(balances[to], value);
        return true;
    } else {
        return false;
    }
}
```

Verified smart contracts* [FSE'18]

- High-profile ERC20 token contracts
- Ethereum Casper FFG (Hybrid PoW/PoS)
- Gnosis MultiSigWallet (ongoing)
- DappHub MakerDAO (by DappHub)
- Uniswap (decentralized exchange)
- Bihu (KEY token operation)

* https://github.com/runtimeverification/verified-smart-contracts

Challenges for EVM bytecode verification

- Byte-twiddling operations
  - Non-linear integer arithmetic (e.g., modulo reduction)
- Arithmetic overflow detection
- Gas limit
  - Variable gas cost depending on contexts
- Hash collision

Byte-twiddling operations

Given:

$$x[n] \stackrel{\texttt{def}}{=} (x/256^n) \bmod 256$$

$$merge(x[i..j]) \stackrel{\texttt{def}}{=} merge(x[i..j+1]) \mathbin{\underline{*}} 256 \mathbin{\underline{+}} x[j] \quad \text{when } i > j$$

$$merge(x[i..i]) \stackrel{\texttt{def}}{=} x[i]$$

Prove:

$$\text{“}x = merge(x[31..0])\text{”}.$$

Abstractions

```
syntax Int ::= nthByte(Int, Int, Int) [function]
```

```
rule merge(nthByte(V, 0, N) ... nthByte(V, N-1, N))
  => V
  requires 0 <= V <  2 ^ (N * 8)
       and 1 <= N <= 32
```

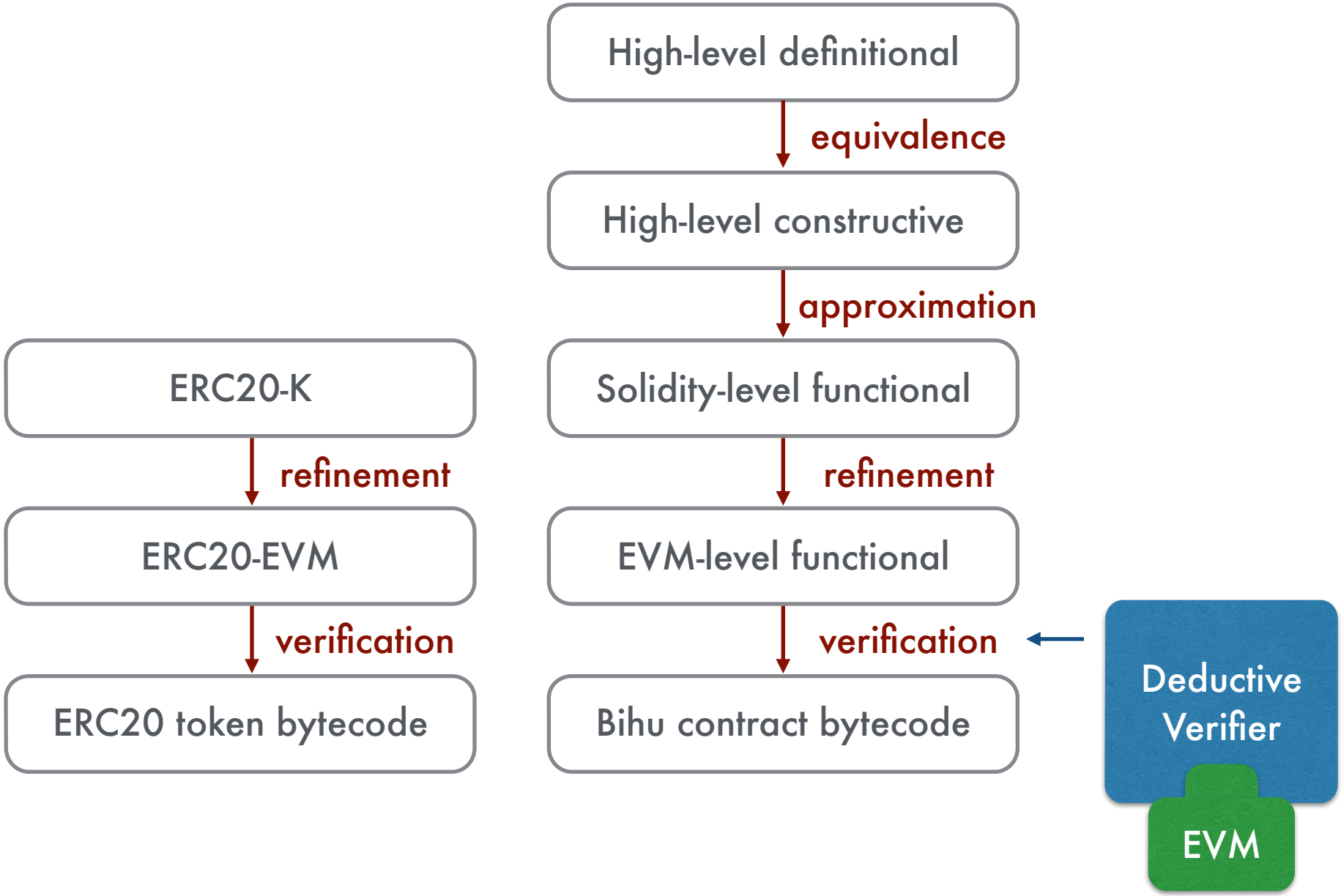
Challenges for EVM bytecode verification

- Byte-twiddling operations
  - Non-linear integer arithmetic (e.g., modulo reduction)
- Arithmetic overflow detection
- Gas limit
  - Variable gas cost depending on contexts
- Hash collision

End-to-end verification

- Formalize high-level business logic
  - Confirmed by contract developers
- Refine it to EVM level
  - Capturing EVM-specific details
- Verify EVM bytecode using derived EVM verifier
  - No need to trust Solidity/Vyper compilers

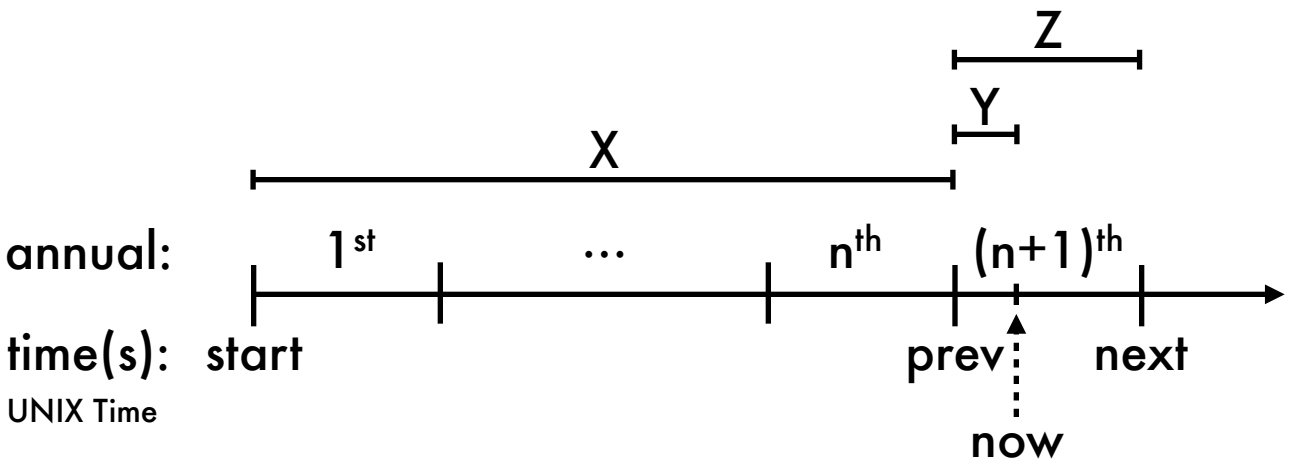
End-to-end verification



Informal specification

"KeyRewardPool contract is responsible for releasing key tokens in the reward pool (initially about 45 billion tokens). The reward pool has a start time. From the start time, every 365 days, we define it as an annual. In each year, a total of 10% of the remaining amount is released. In each year, the token is released linearly."

High-level definitional specification



$$Y = Z \times b$$

$$Z = T \times 0.9^n \times 0.1$$

$$X = T(1 - 0.9^n)$$

$$X + Z = T(1 - 0.9^{n+1})$$

$$b = \frac{\text{now} - \text{prev}}{\text{next} - \text{prev}}$$

High-level constructive specification

```
/* @input:   balance                                                    // T − C
 *           collectedTokens                                            // C
 *           rewardStartTime                                            // start
 *           now                                                        // now
 *
 * @output: balance'                                                   // T − C′
 *           collectedTokens'                                           // C′
 *
 * @pre−condition: now > rewardStartTime
 */
procedure collectToken() {
  total := collectedTokens + balance                                    // T
  yearCount := floor(days(now − rewardStartTime) / 365)                 // n
  fractionOfThisYear := (days(now − rewardStartTime) % 365) / 365       // b

  remainingTokens := total * (0.9 ^ yearCount)
  totalRewardThisYear := remainingTokens * 0.1                          // Z
  canExtractThisYear := totalRewardThisYear * fractionOfThisYear        // Y
  canExtract := canExtractThisYear + (total − remainingTokens)
                                   − collectedTokens                    // Y + X − C

  collectedTokens' := collectedTokens + canExtract                      // C′
  balance' := balance − canExtract                                      // T − C′
}
```

High-level constructive specification

```
/* @input:   balance                                   // T − C
 *           collectedTokens                           // C
 *           rewardStartTime                           // start
 *           now                                       // now
 *
 * @output: balance'                                   // T − C'
 *           collectedTokens'                          // C'
```

Lemma 1. *If the inputs of* `collectToken` *in Figure 2 satisfies the following equations:*

$$\begin{aligned} \mathit{balance} &= T - C \\ \mathit{collectedTokens} &= C \\ \mathit{rewardStartTime} &= \mathit{start} \\ \mathit{now} &= \mathit{now} \end{aligned}$$

then the following holds for the outputs of the function:

$$\begin{aligned} \mathit{balance'} &= T - C' &&= T - (X + Y) \\ \mathit{collectedTokens'} &= C' &&= X + Y \end{aligned}$$

```
    collectedTokens' := collectedTokens + canExtract   // C'
    balance' := balance − canExtract                   // T − C'
}
```

Solidity-level functional specification

```
/* @input:  uint balance
 *          unit collectedTokens
 *          unit rewardStartTime
 *          unit now
 *
 * @output: unit balance'
 *          unit collectedTokens'
 *
 * @pre-condition: now > rewardStartTime
 */
procedure collectToken() {
  unit total := collectedTokens + balance
  unit yearCount := days(now - rewardStartTime) / 365
  unit fractionOfThisYear365 := days(now - rewardStartTime) % 365

  unit remainingTokens := power(total, 90, 100, yearCount)
  unit totalRewardThisYear := remainingTokens * 10 / 100
  unit canExtractThisYear := totalRewardThisYear * fractionOfThisYear365 / 365
  unit canExtract := canExtractThisYear + (total - remainingTokens)
                                        - collectedTokens

  collectedTokens' := collectedTokens + canExtract
  balance' := balance - canExtract
}

// return (conceptually): acc * ((base_n / base_d) ^ exp)
function power(acc, base_n, base_d, exp) {
  if exp == 0 {
    return acc
  } else {
    return power(acc * base_n / base_d, base_n, base_d, exp - 1)
  }
}
```

Solidity-level functional specification

```
/* @input:  uint balance
 *          unit collectedTokens
 *          unit rewardStartTime
 *          unit now
```

Lemma 2. *If the inputs of* `collectToken` *in Figure 3 satisfies the following equations:*

$$\begin{aligned} \texttt{balance} &= T - C \\ \texttt{collectedTokens} &= C \\ \texttt{rewardStartTime} &= \textit{start} \\ \texttt{now} &= \textit{now} \end{aligned}$$

then the following holds for the outputs of the function:

$$\begin{array}{rcccl} (T - C') - 10 & < & \texttt{balance'} & < & (T - C') + 3 \\ C' - 3 & < & \texttt{collectedTokens'} & < & C' + 10 \end{array}$$

Lemma 3. *Suppose two* `collectToken` *function (as shown in Figure 3) calls are made at times* t *and* t'*, respectively, where* $t < t'$*. Let* r *and* r' *be the output values of* $\texttt{canExtractThisYear} + (\texttt{total} - \texttt{remainingTokens})$ *for each function call at* t *and* t'*, respectively. Assume that* `total` *does* not *descrese between* t *and* t'*. Then,* $r \leq r'$*.*

Corollary 1. *If* `balance` *does not decrease since the last* `collectToken` *function call, the following always hold:*

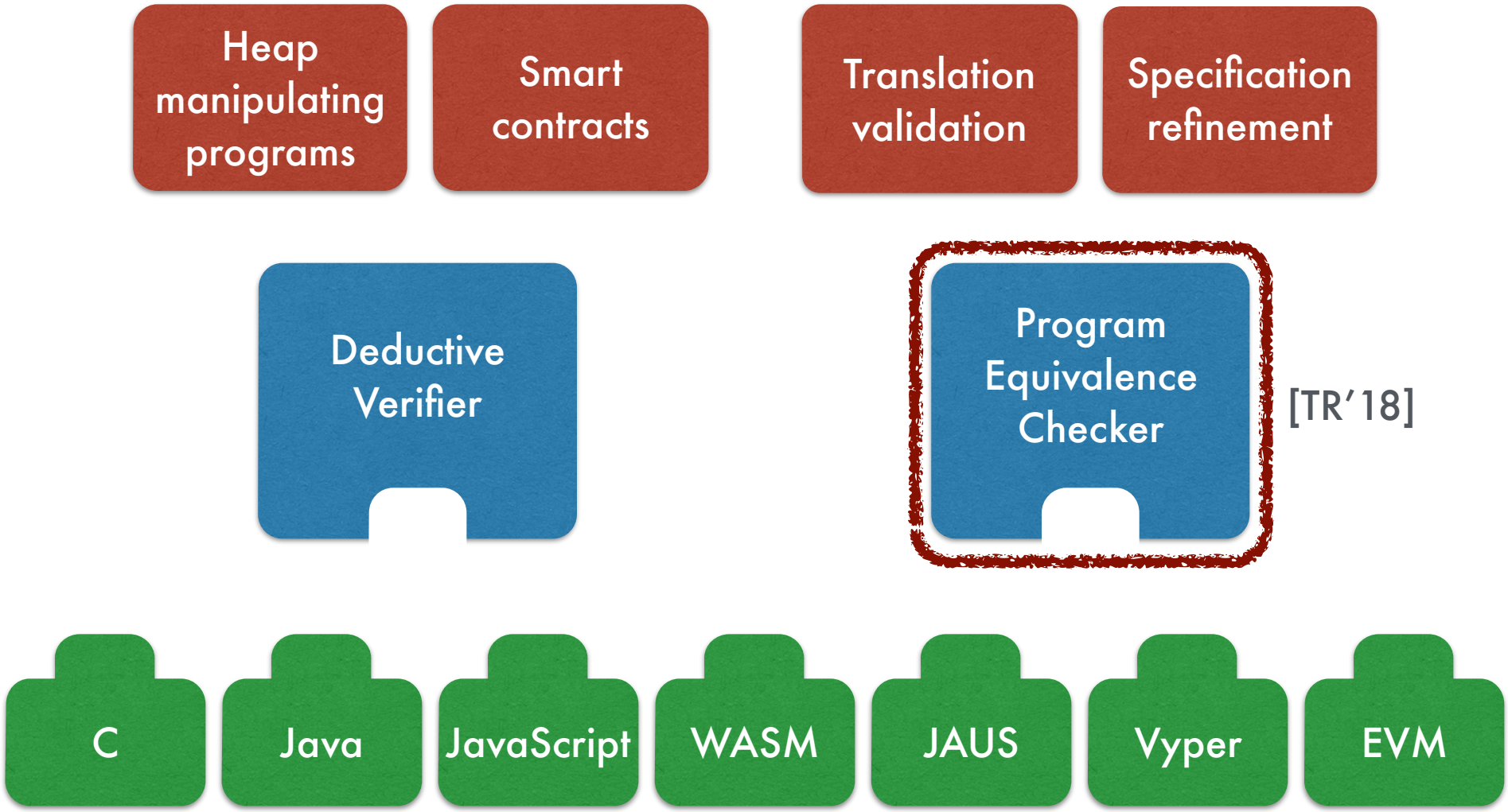
$$\texttt{canExtractThisYear} + (\texttt{total} - \texttt{remainingTokens}) \geq \texttt{collectedTokens}$$

```
  } else {
    return power(acc * base_n / base_d, base_n, base_d, exp - 1)
  }
}
```

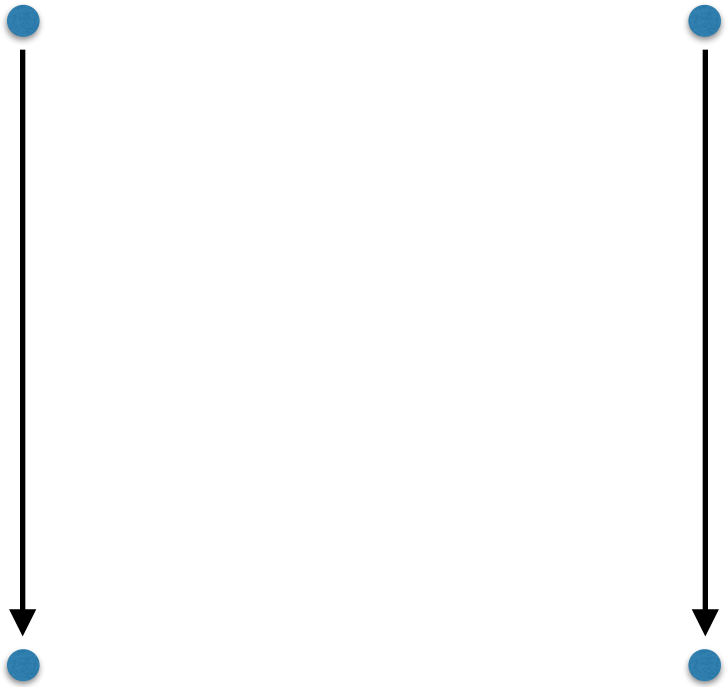
EVM-level functional specification

```
k: #execute => (RETURN RET_ADDR:Int 32 ~> _)
output: _
memoryUsed: 0 => _
callData: #abiCallData("collectToken", #uint256(NOW), #uint256(START))
wordStack: .WordStack => _
localMem: .Map => .Map[ RET_ADDR := #asByteStackInWidth(1, 32) ] _:Map
pc: 0 => _
gas: GASCAP => _
log: _
refund: _ => _
storage:
    #hashedLocation("Solidity", {_COLLECTEDTOKENS}, .IntList) |-> (COLLECTED => COLLECTED +Int VALUE)
    #hashedLocation("Solidity", {_BALANCE},         .IntList) |-> (BAL       => BAL       -Int VALUE)
    _:Map
requires:
    andBool 0 <=Int COLLECTED  andBool COLLECTED <Int (2 ^Int 256)
    andBool 0 <=Int BAL        andBool BAL       <Int (2 ^Int 256)
    andBool 0 <=Int START      andBool START     <Int (2 ^Int 256)
    andBool 0 <=Int (COLLECTED +Int BAL) andBool (COLLECTED +Int BAL) *Int 3153600 <Int (2 ^Int 256)
    andBool 0 <Int (NOW -Int START) andBool (NOW -Int START) <Int (2 ^Int 256)
    andBool #accumulatedReleasedTokens(BAL, COLLECTED, START, NOW) >Int COLLECTED +Int 3
    andBool #accumulatedReleasedTokens(BAL, COLLECTED, START, NOW) <Int (BAL +Int COLLECTED) -Int 10
    andBool GASCAP >=Int (293 *Int ((NOW -Int START) /Int 31536000)) +Int 43000
ensures: VALUE ==Int @canExtractThisYear(COLLECTED +Int BAL, NOW, START)
                     +Int BAL -Int @remainingTokens(COLLECTED +Int BAL, NOW, START)
```

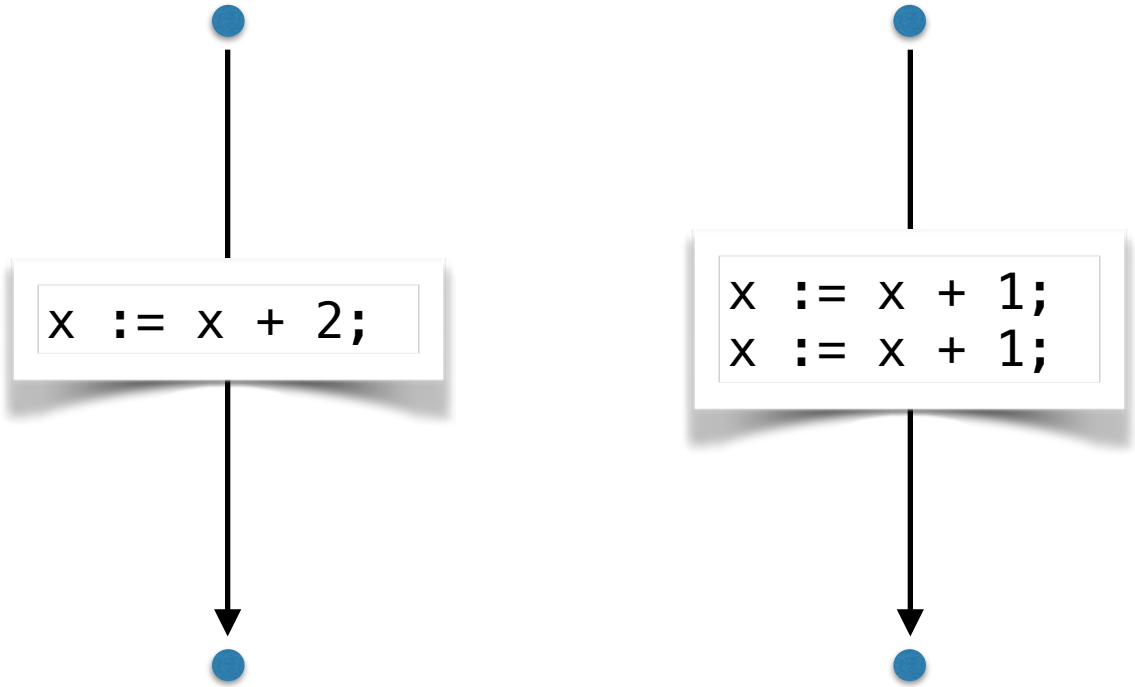
Developing new formal method



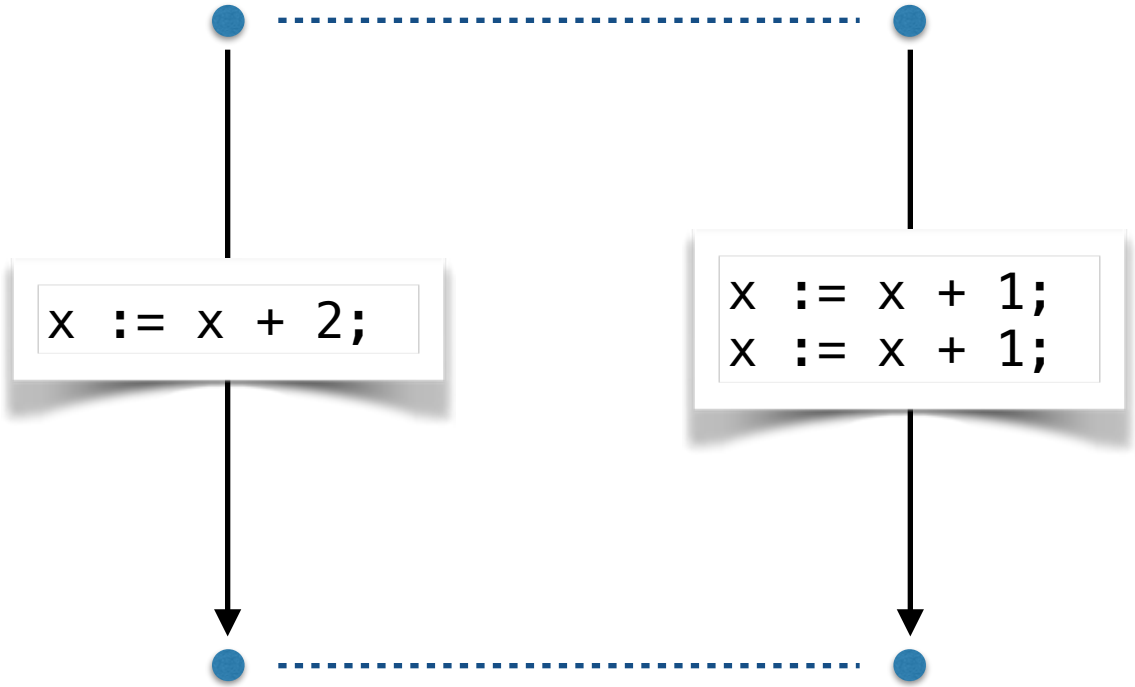
Deterministic, terminating programs



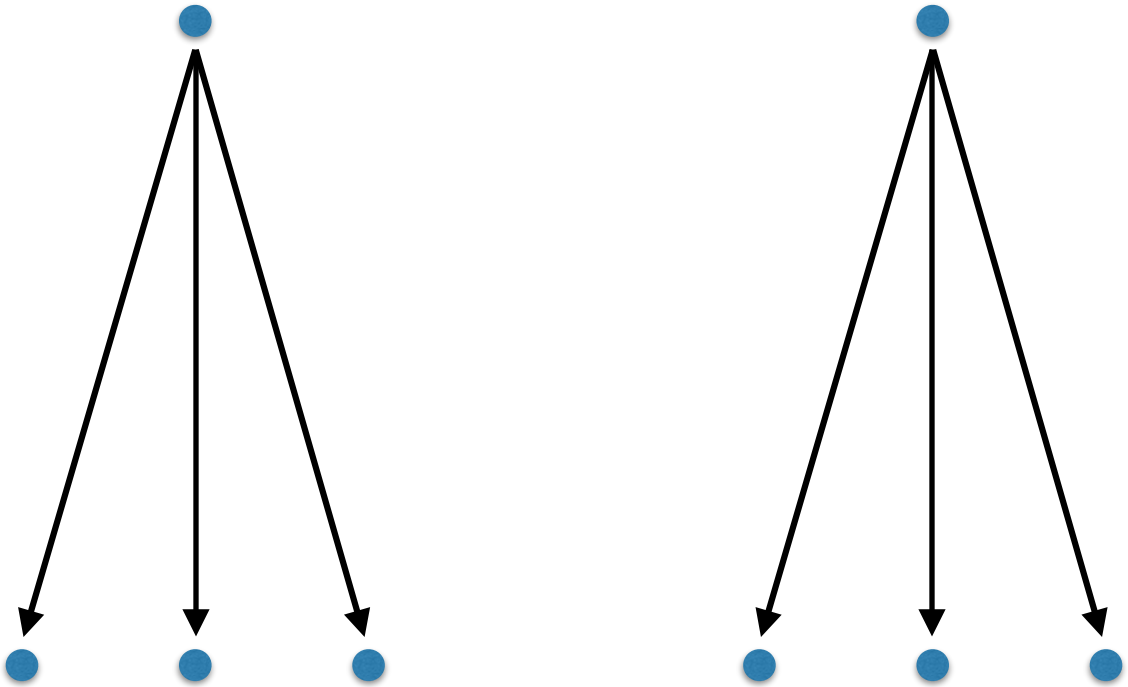
Deterministic, terminating programs



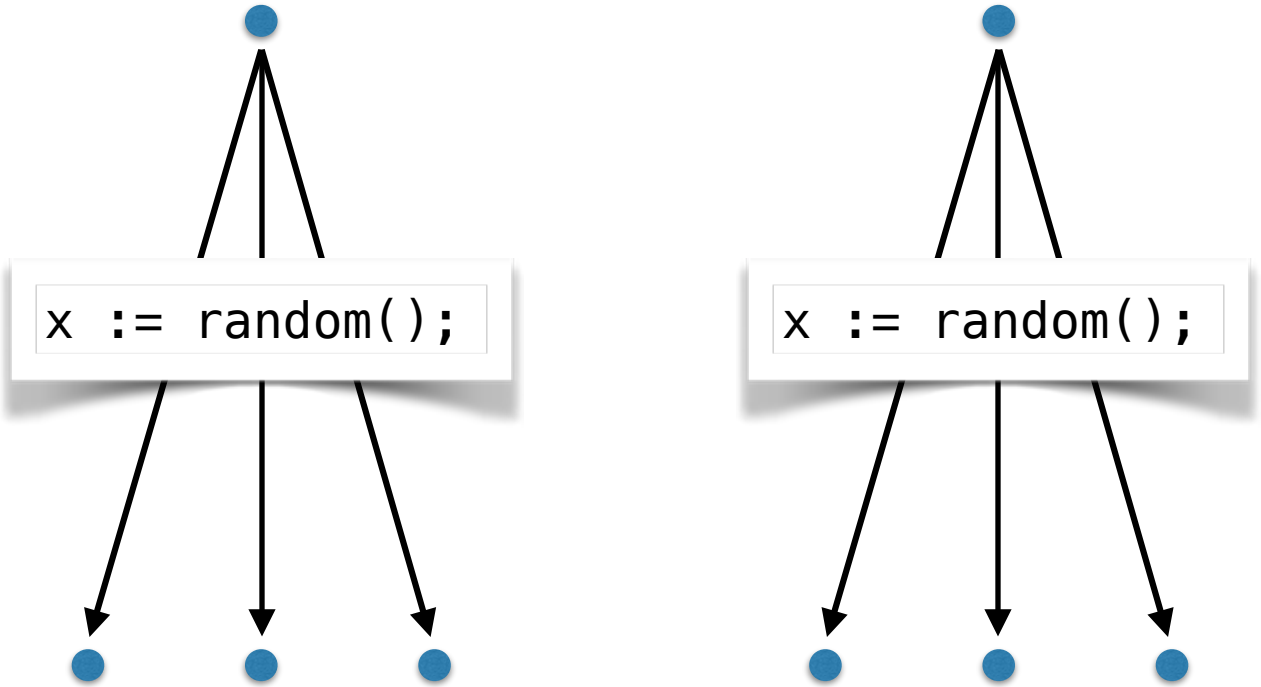
Deterministic, terminating programs



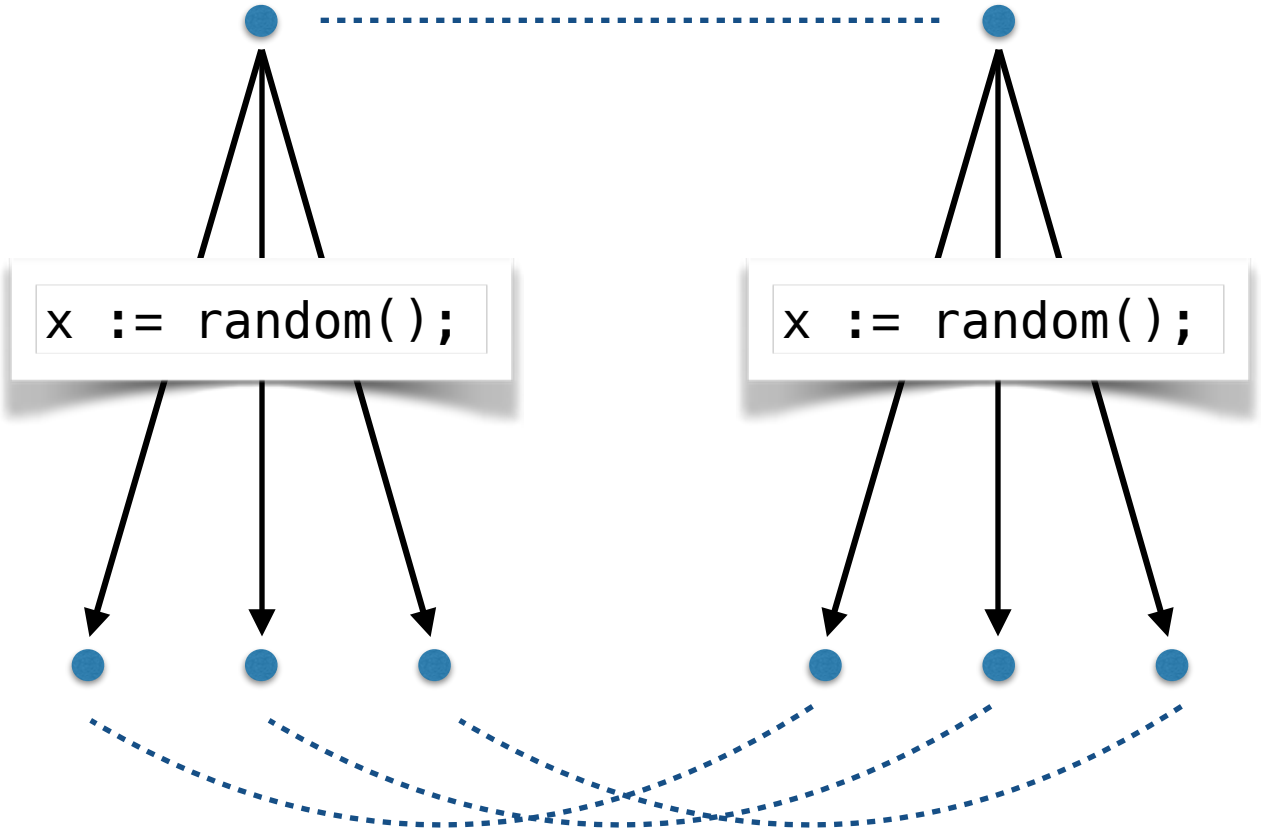
Nondeterministic programs



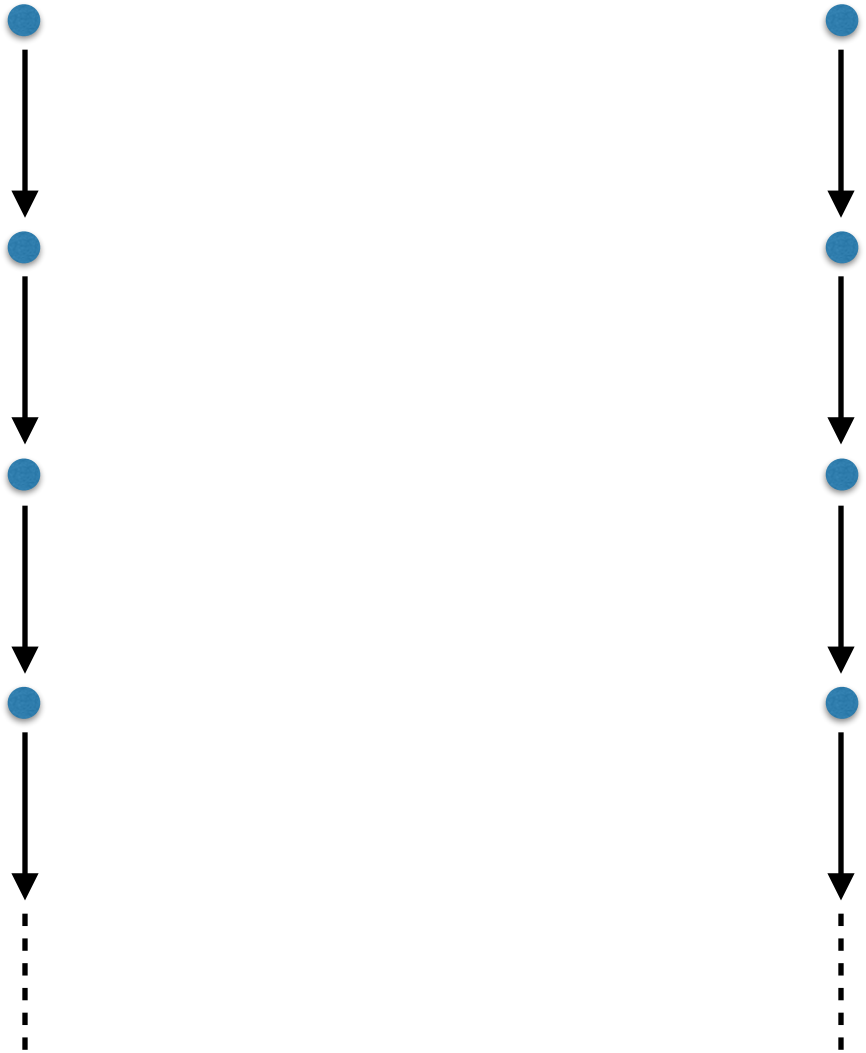
Nondeterministic programs



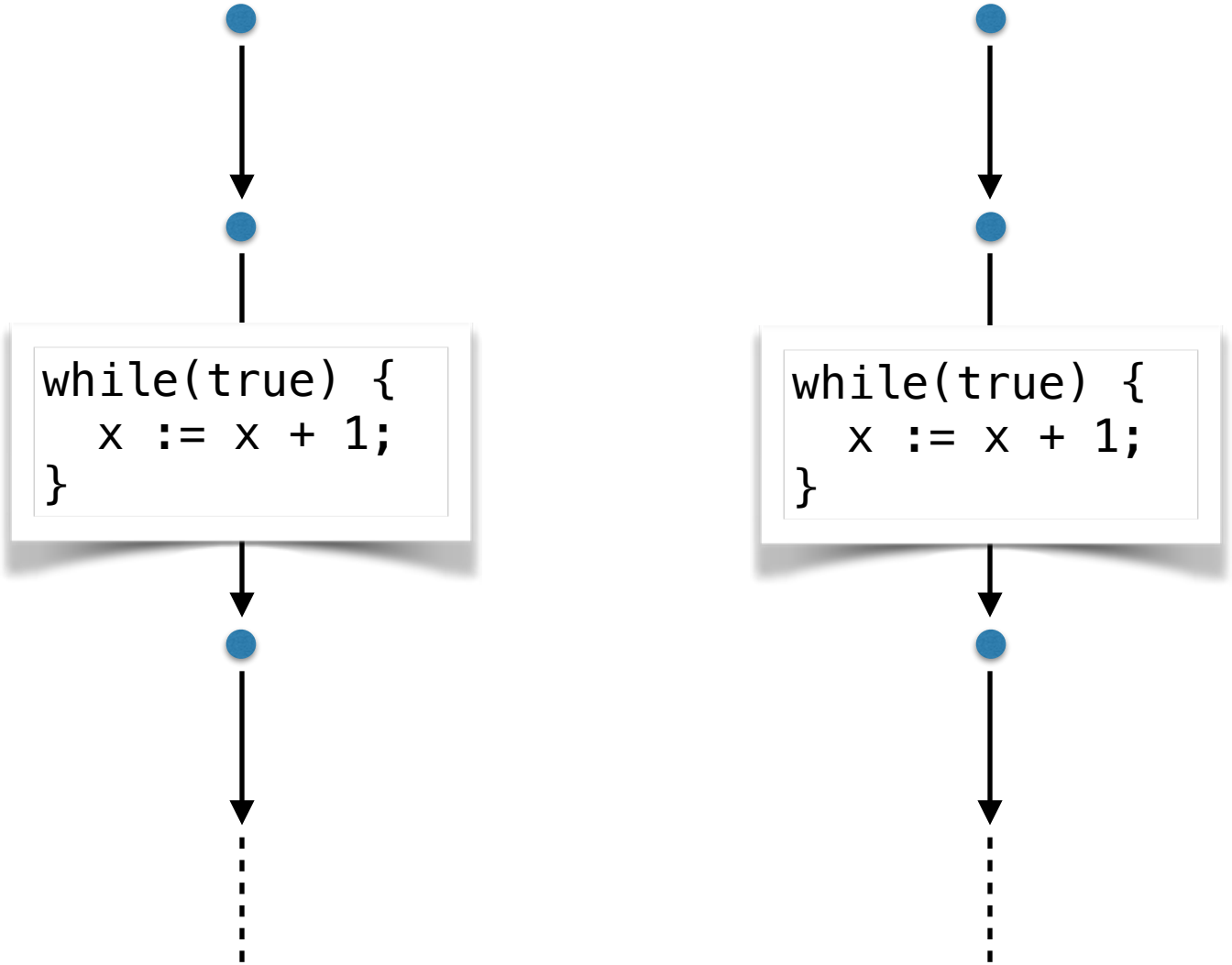
Nondeterministic programs



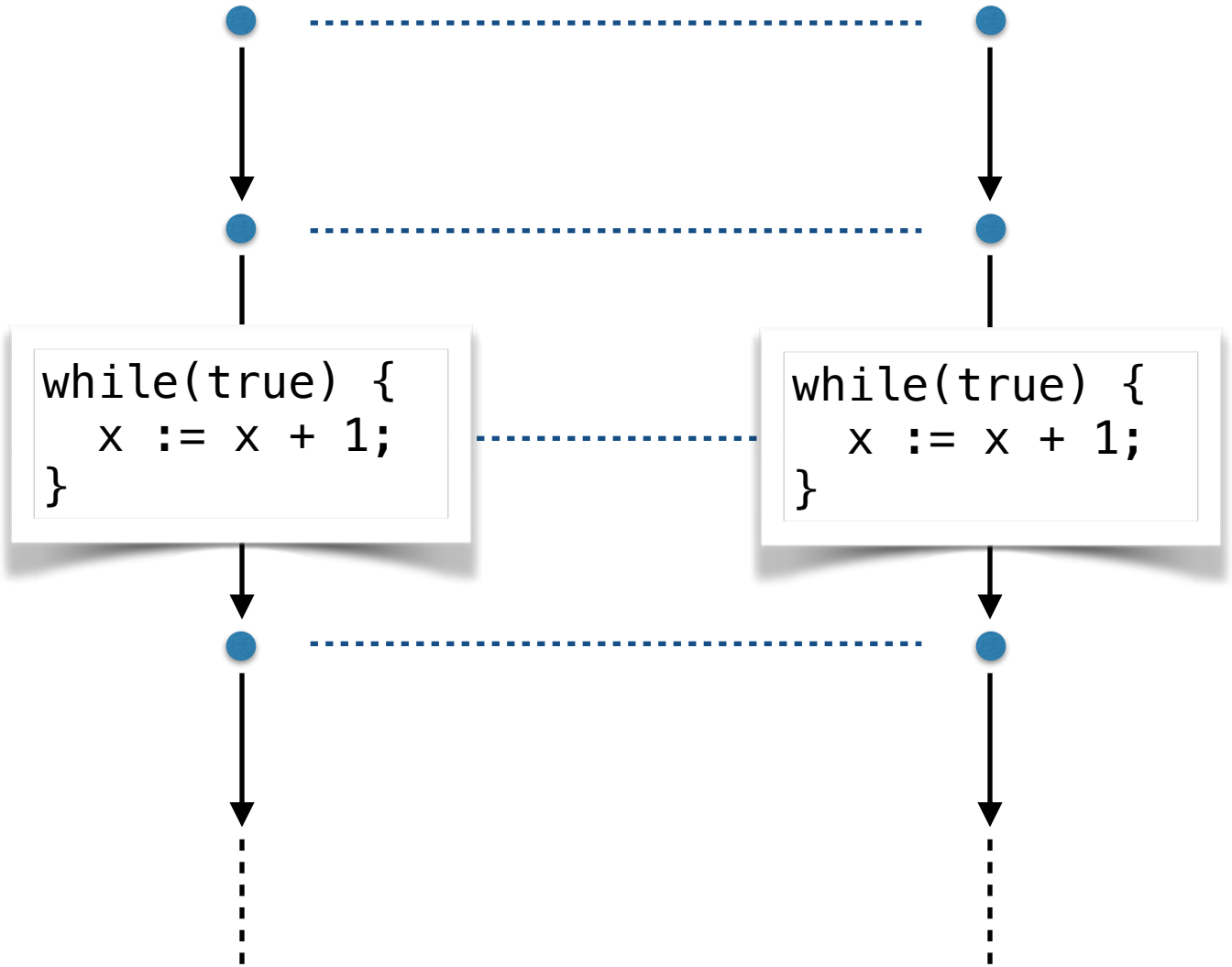
Non-terminating programs



Non-terminating programs



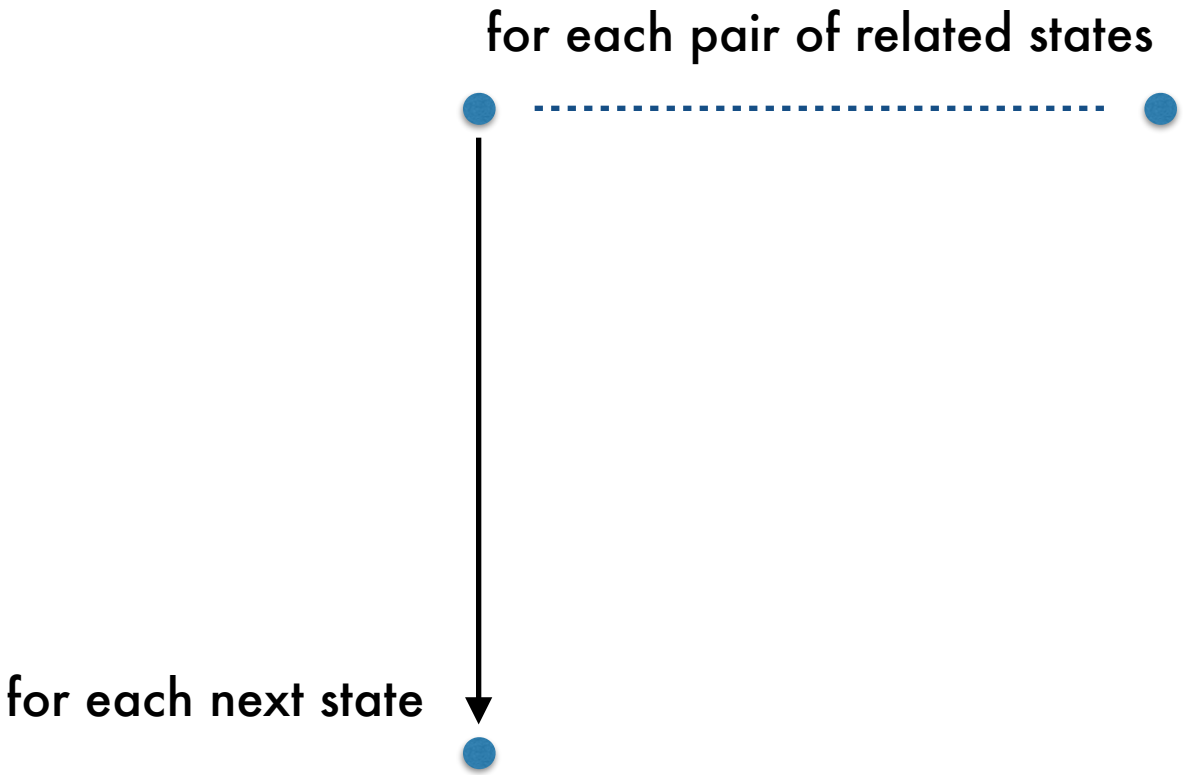
Non-terminating programs



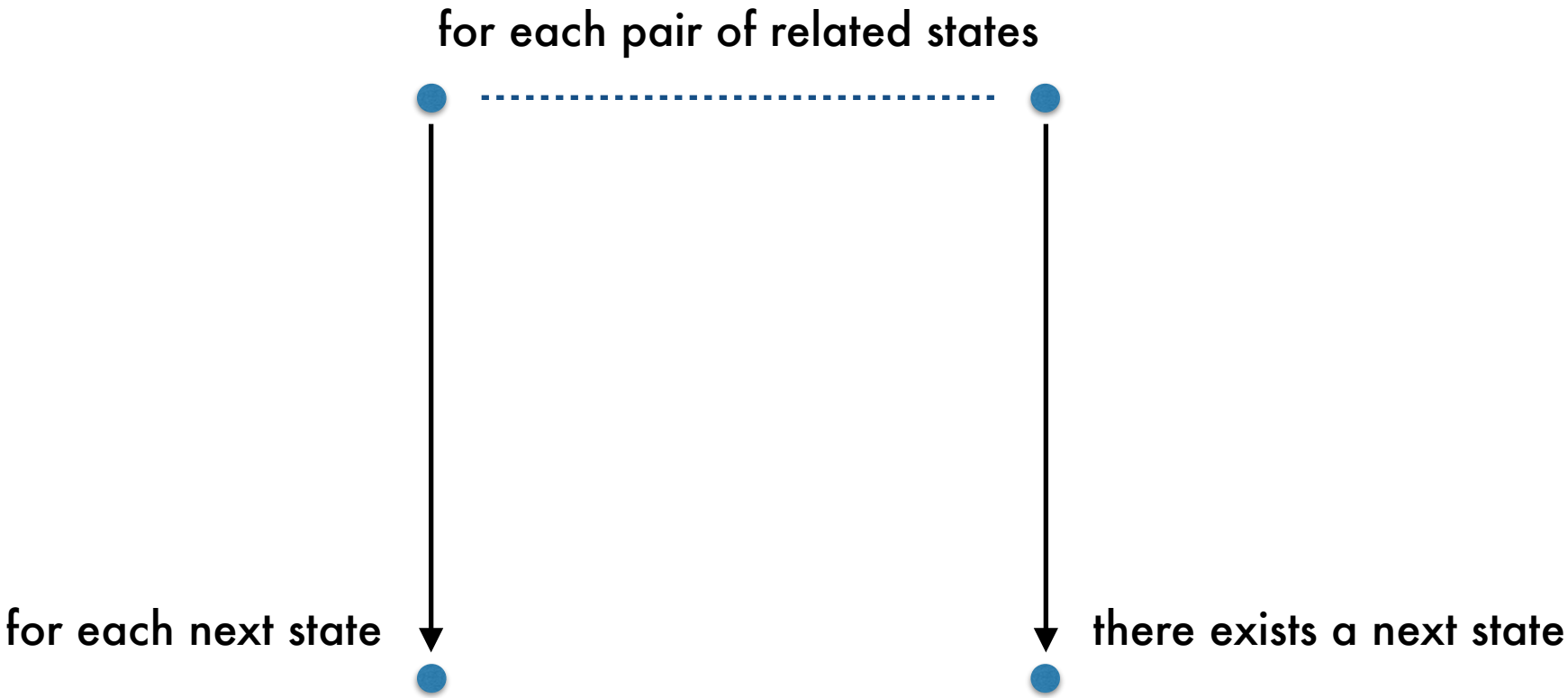
Bisimulation

for each pair of related states

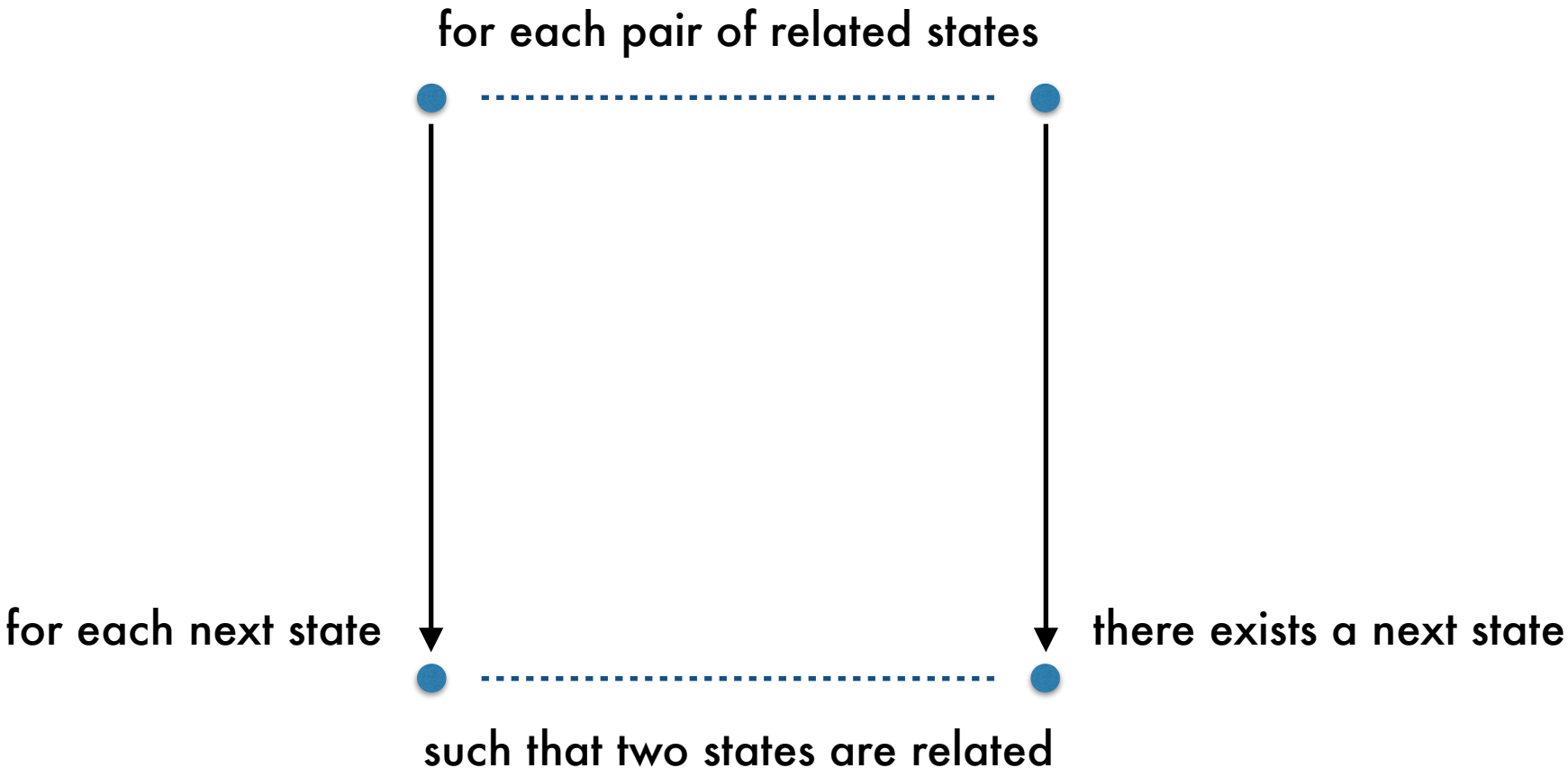
Bisimulation



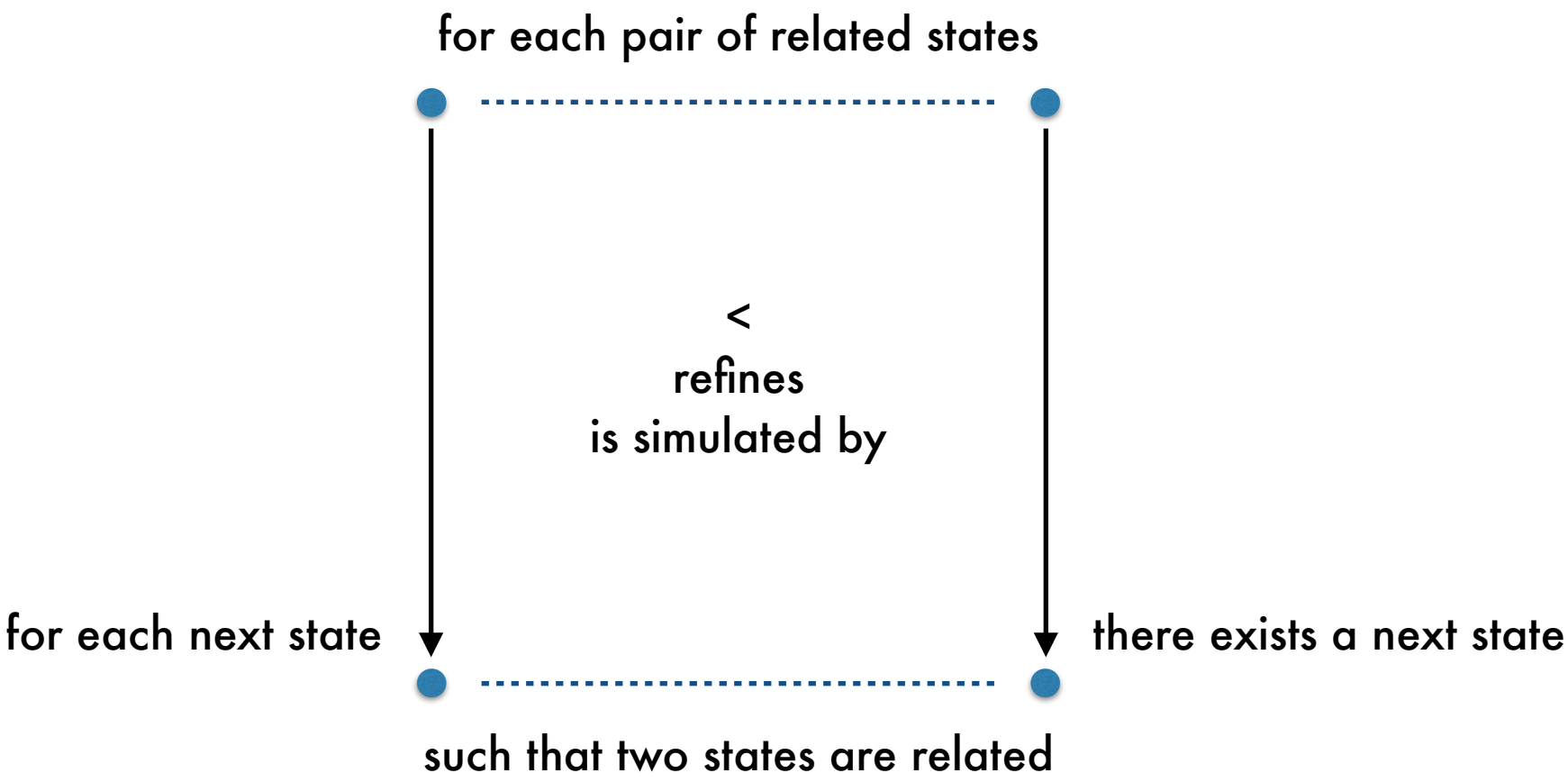
Bisimulation



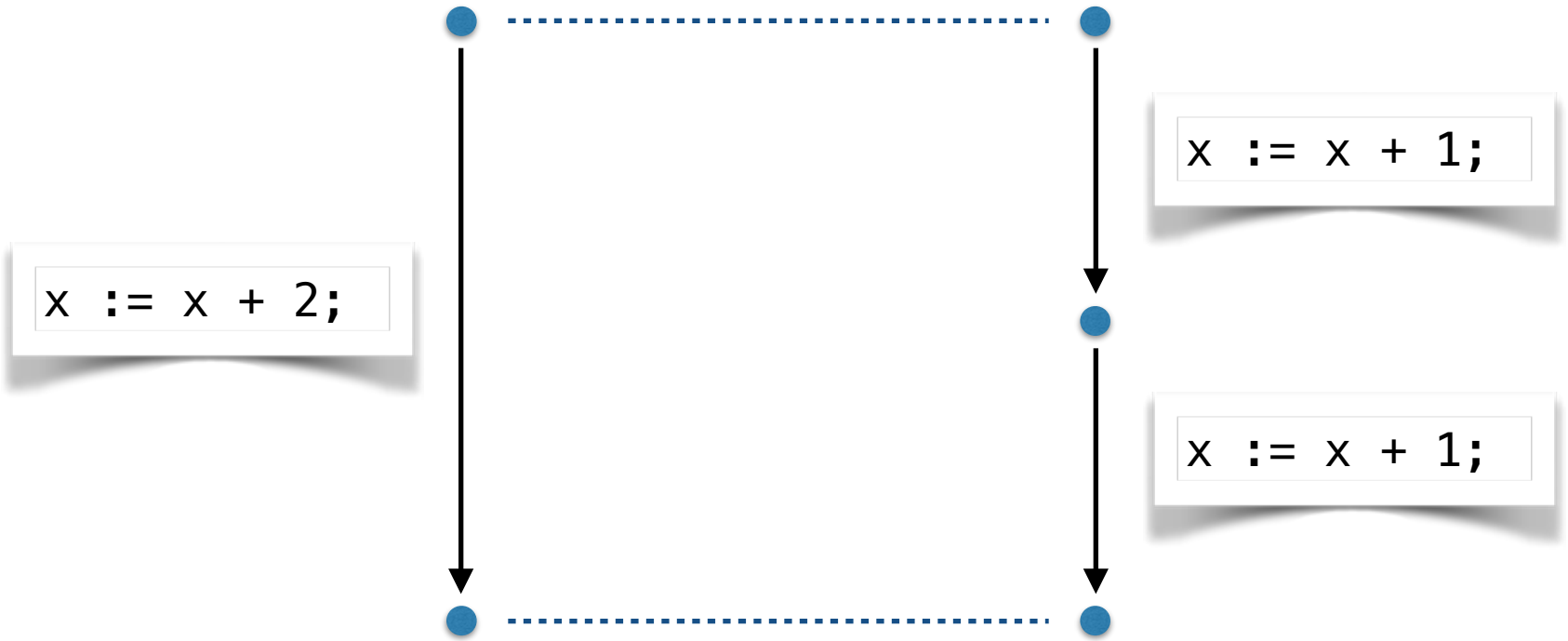
Bisimulation



Bisimulation

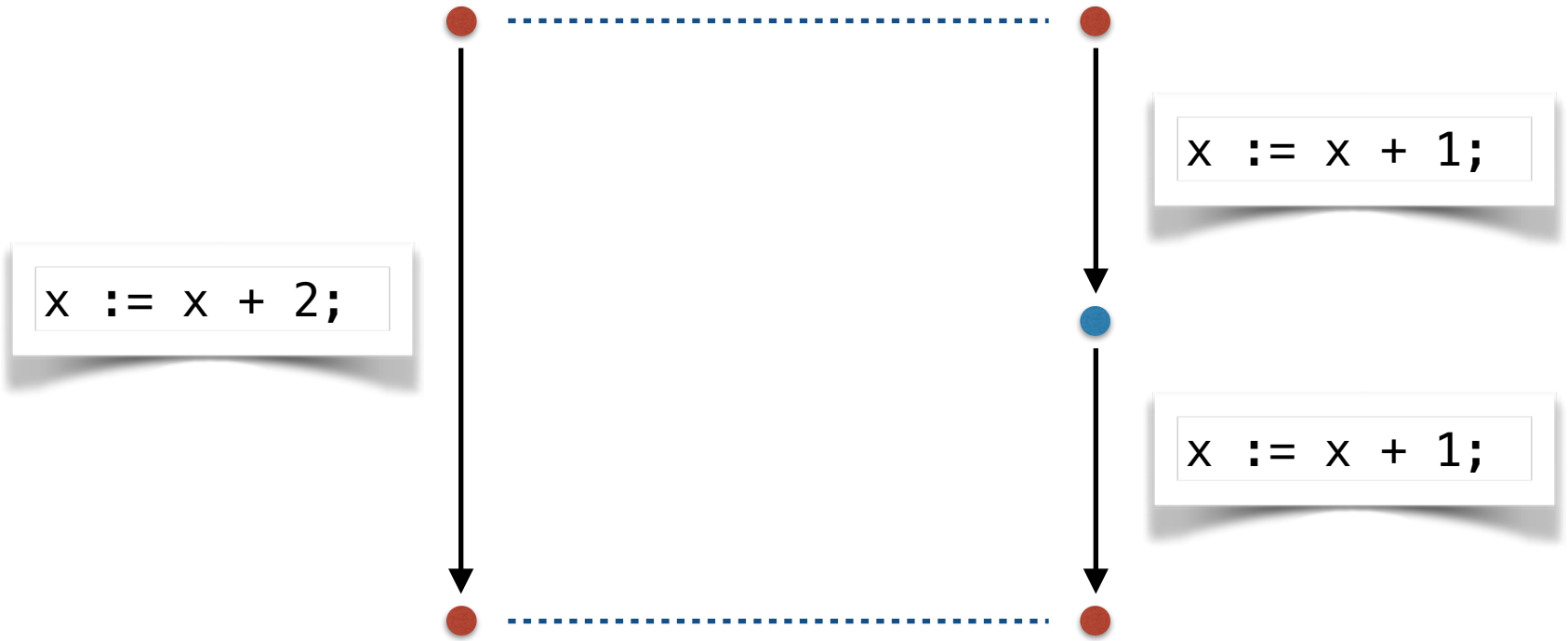


Bisimulation?

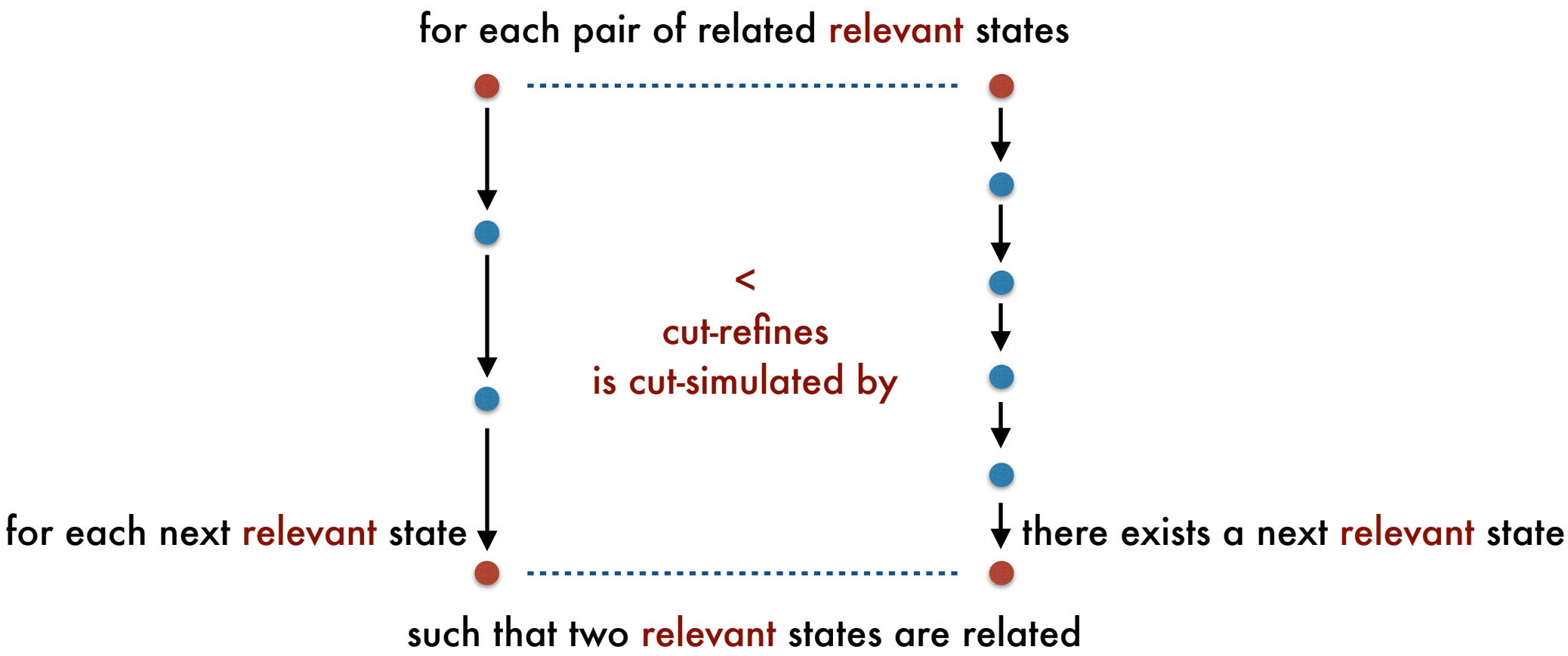


Cut-bisimulation

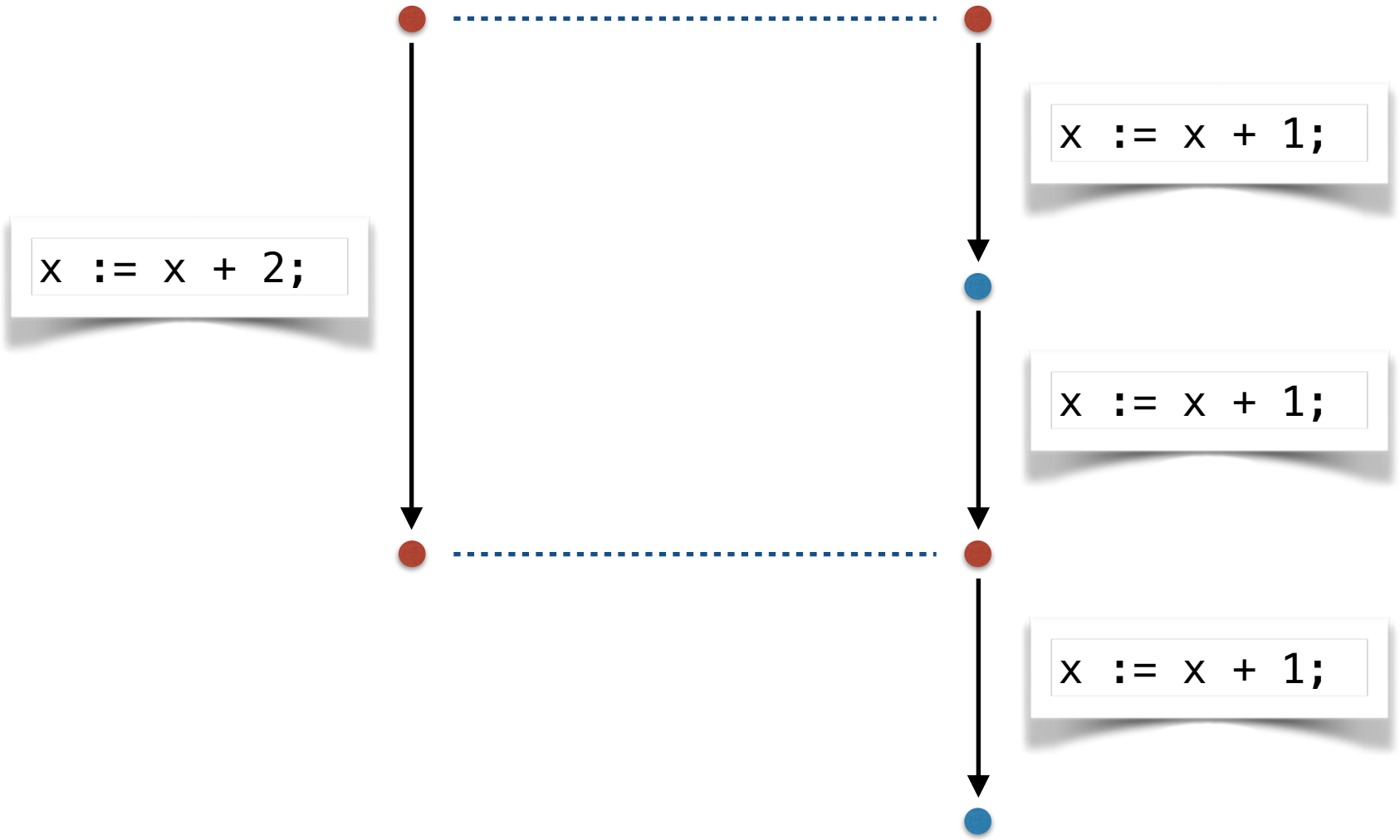
considering only relevant states



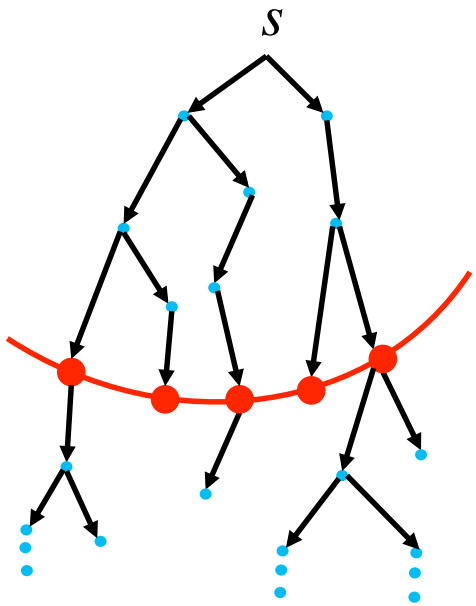
Cut-bisimulation



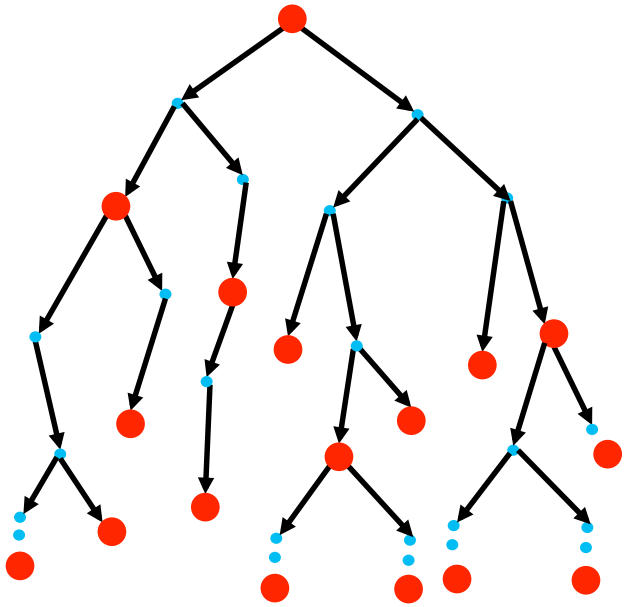
Well-formedness of relevant states



Cut



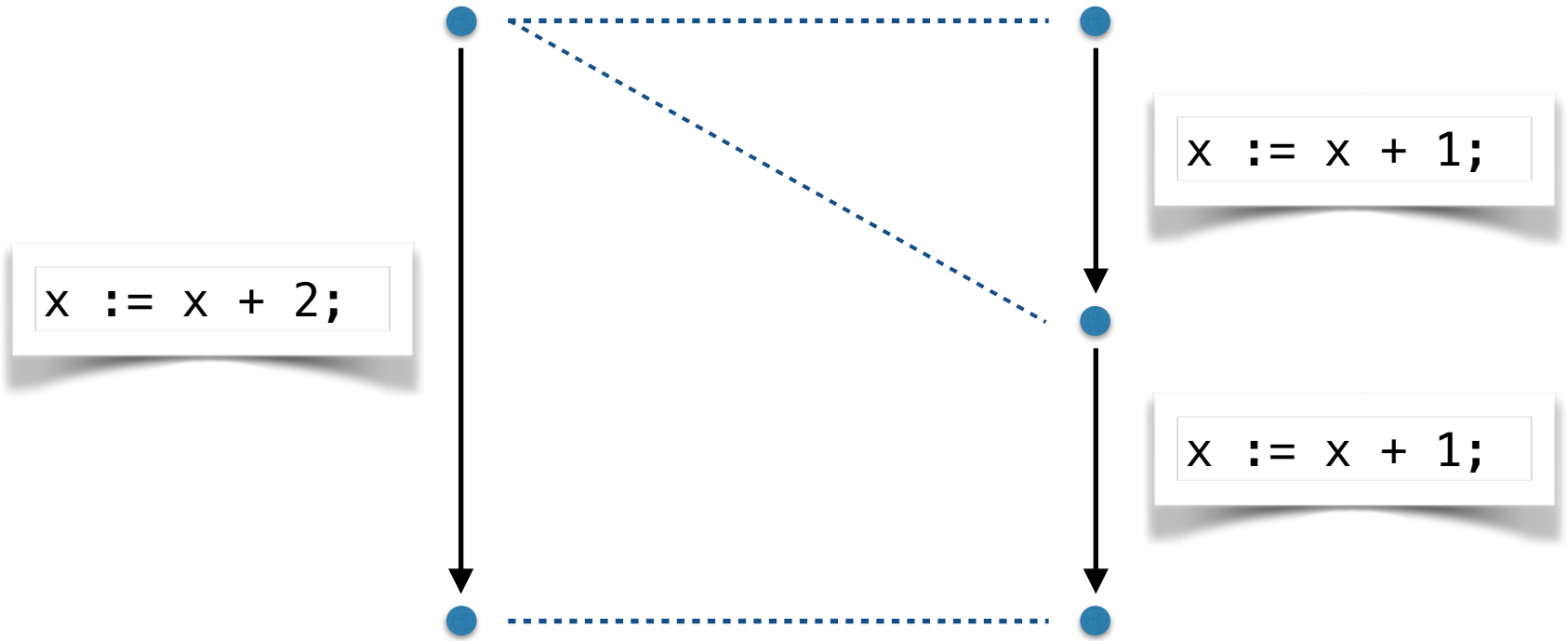
Cut for s



Cut for all

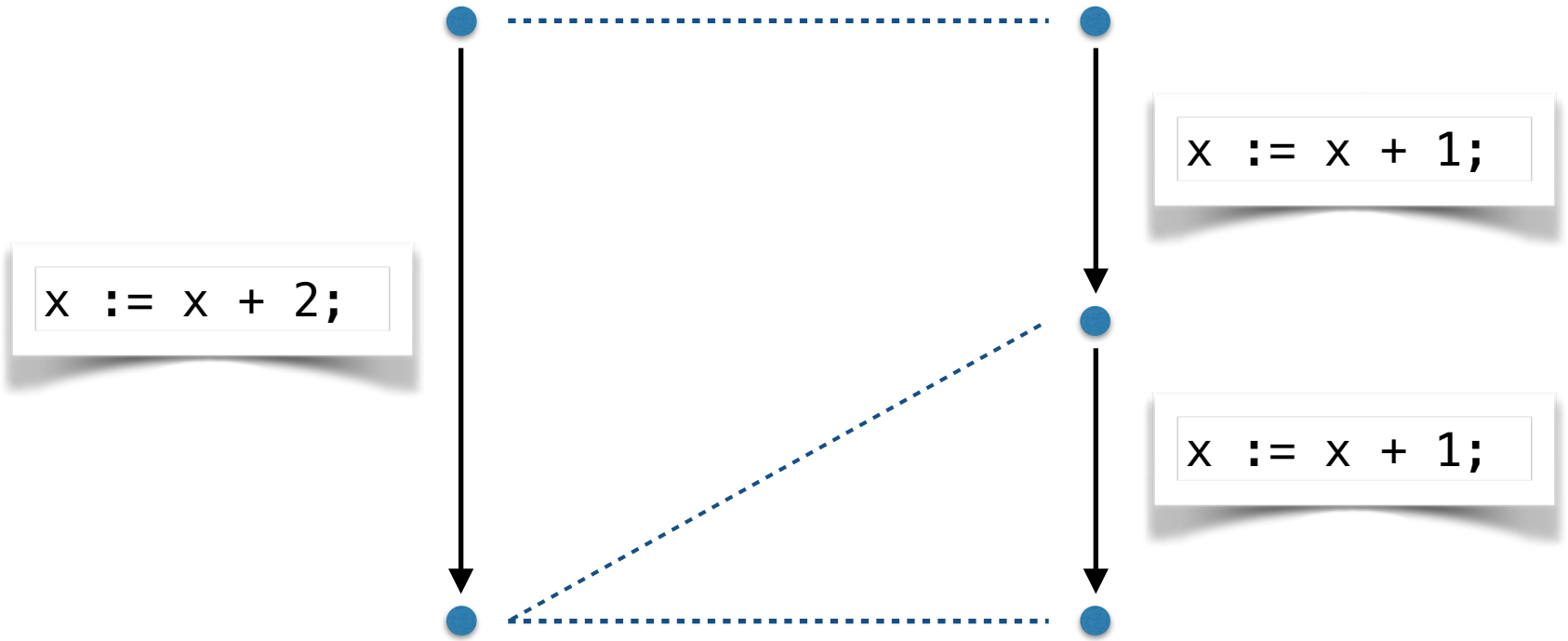
Stuttering bisimulation [BCG'88, dNV'90]

two possible stuttering bisimulations: 1 of 2

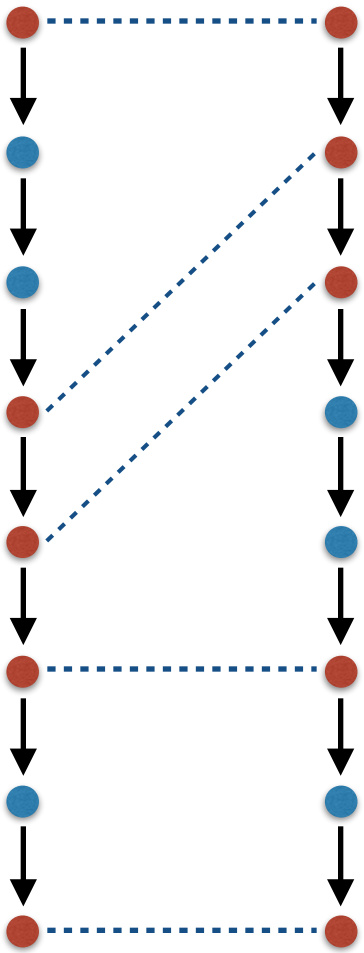


Stuttering bisimulation [BCG'88, dNV'90]

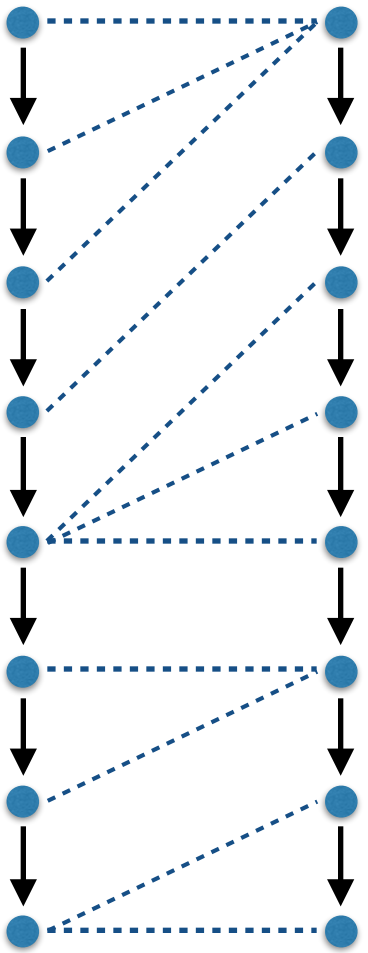
two possible stuttering bisimulations: 2 of 2



Cut-bisimulation vs stuttering bisimulation



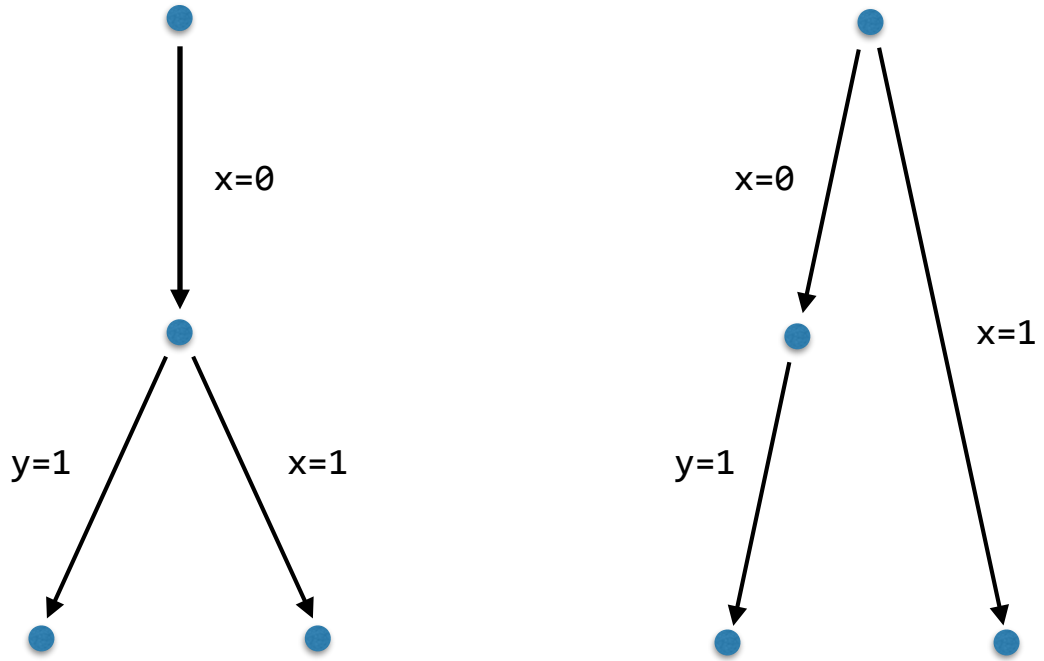
vs



[Namjoshi'97]

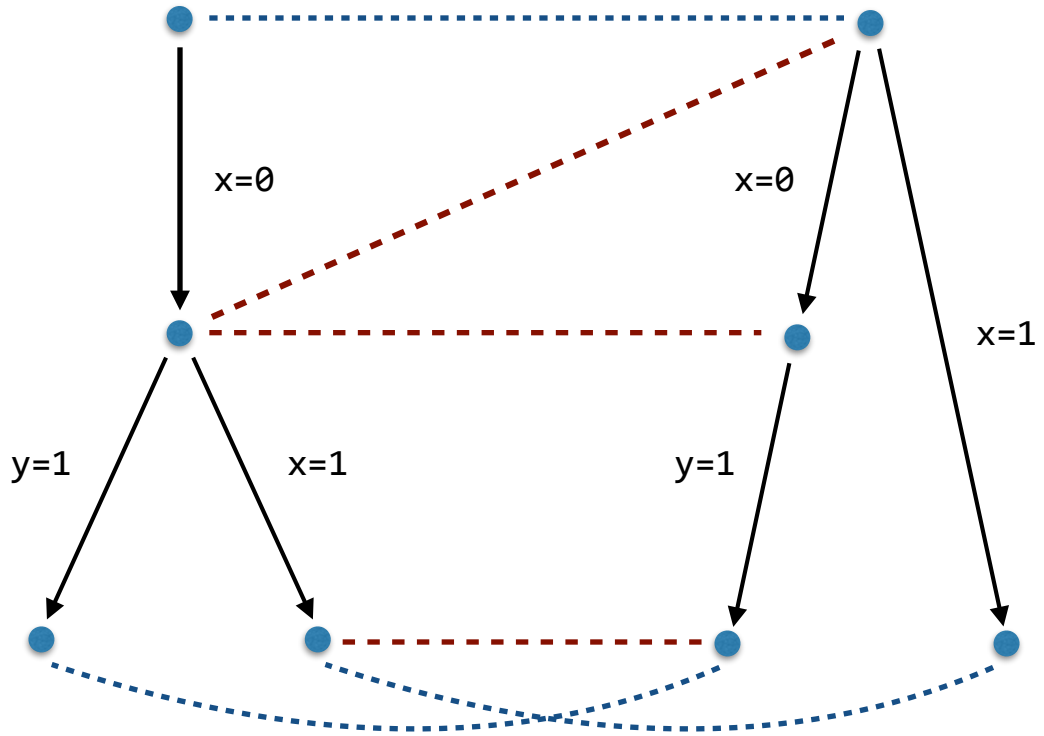
Non-intuitive stuttering bisimulation

Partial redundancy elimination



Non-intuitive stuttering bisimulation

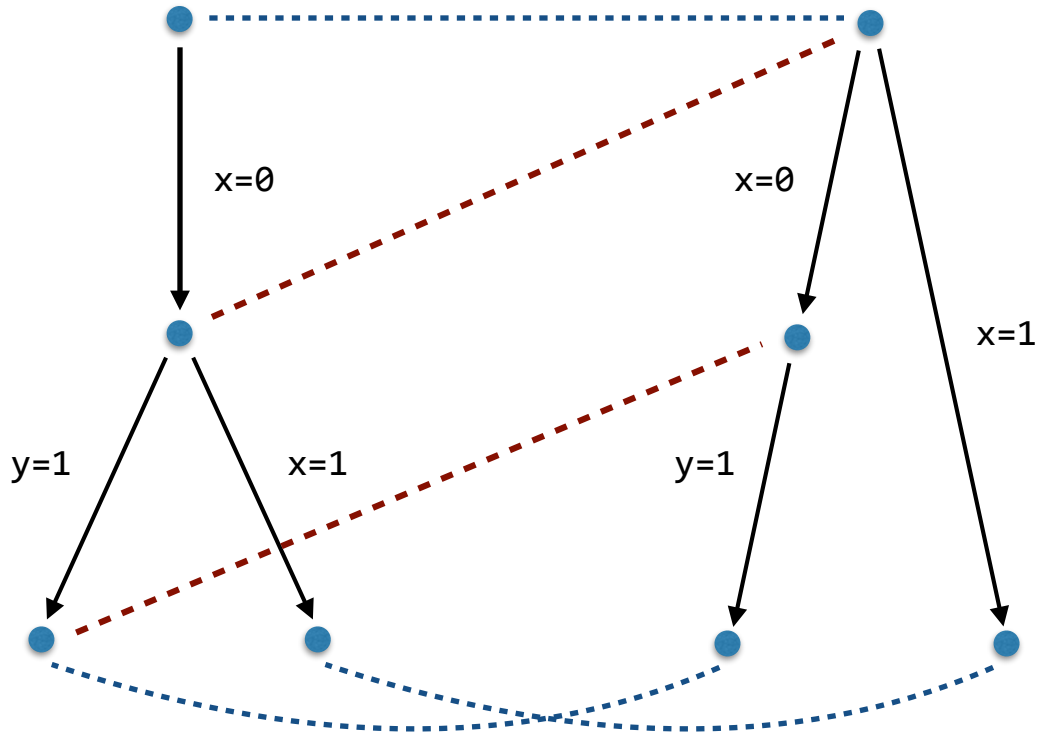
Partial redundancy elimination



two possible stuttering bisimulations: 1 of 2

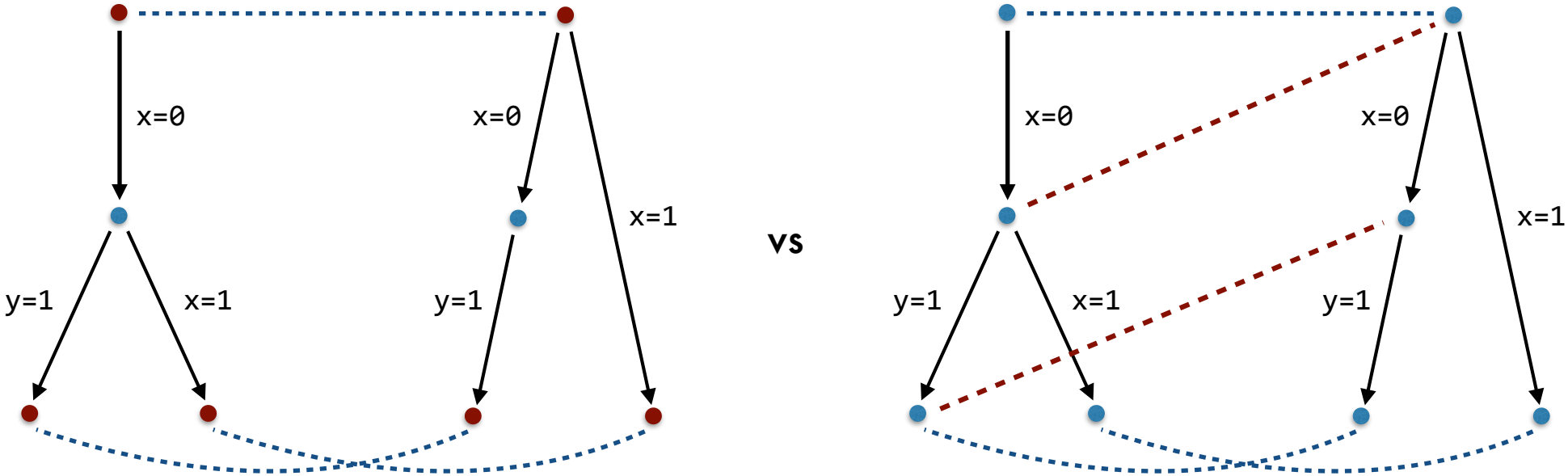
Non-intuitive stuttering bisimulation

Partial redundancy elimination



two possible stuttering bisimulations: 2 of 2

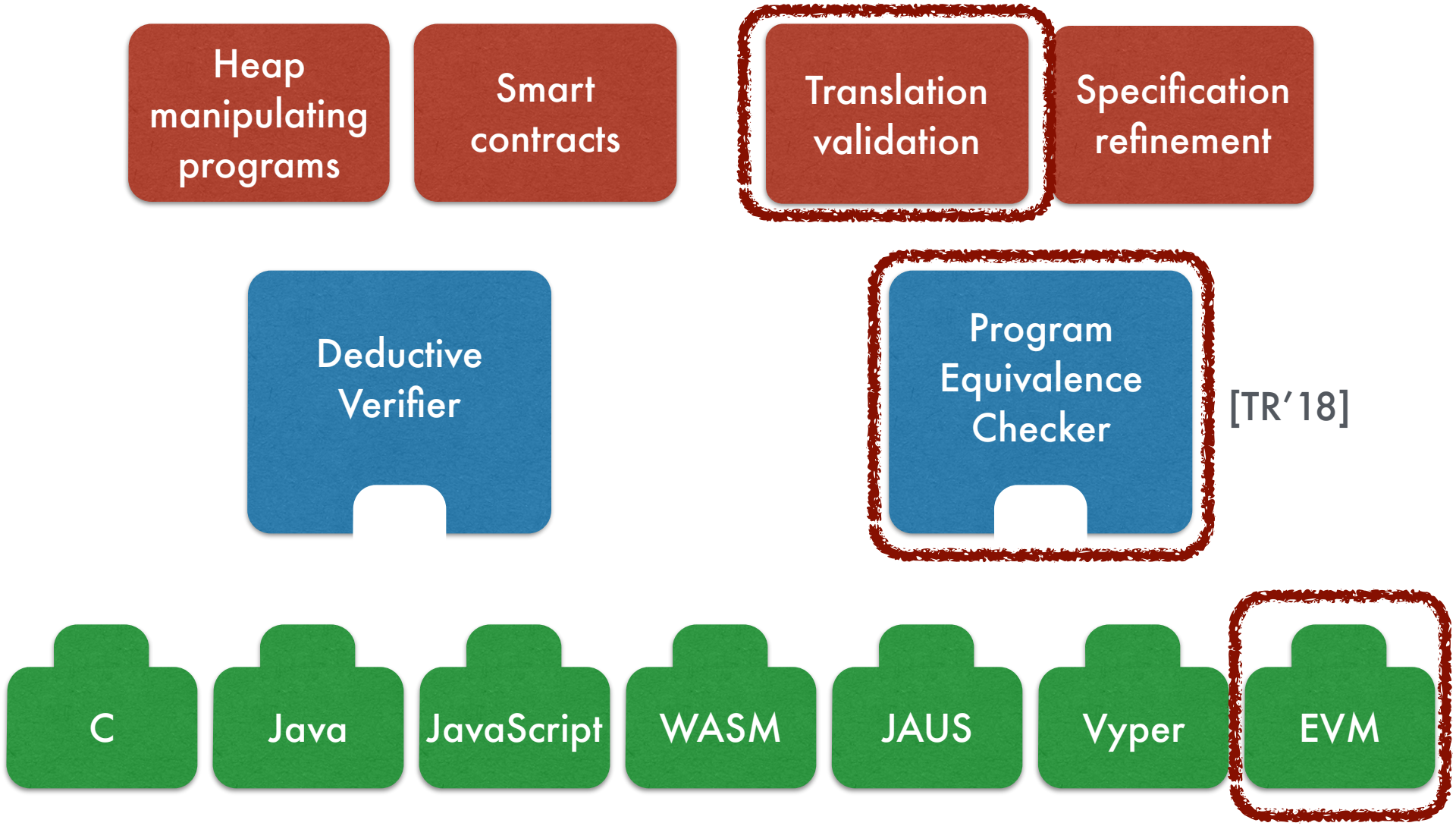
Cut-bisimulation vs stuttering bisimulation



KEQ: universal program equivalence checker

- Parameters
  - Two language semantics
- Inputs
  - Two programs
  - A relation (called synchronization points)
- Output
  - True if the relation is a cut-(bi)simulation
- Can be used for translation validation for compiler

Evaluating new formal method



Translation validation [Pnueli *et al.*'98]

- Validate each instance of compilation
- Verify equivalence of input/output programs
  - KEQ can be used for checking equivalence
- Practical advantages for compiler verification

Example

```
int foo(unsigned n) {
  int i = 0;
  while (i < n) {
    i = i + 1;
  }
  return i;
}
```

```
int foo(unsigned n) {
  int i = 0;
  while (i < n) {
    i = i + 2;
  }
  return i;
}
```

Example

```
int foo(unsigned n) {
  int i = 0;
  while (i < n) {
    i = i + 1;
  }
  return i;
}
```

```
int foo(unsigned n) {
  int i = 0;
  while (i < n) {
    i = i + 2;
  }
  return i;
}
```

Example

```
int foo(unsigned n) {
  int i = 0;
  while (i < n) {
    i = i + 1;
  }
  return i;
}
```

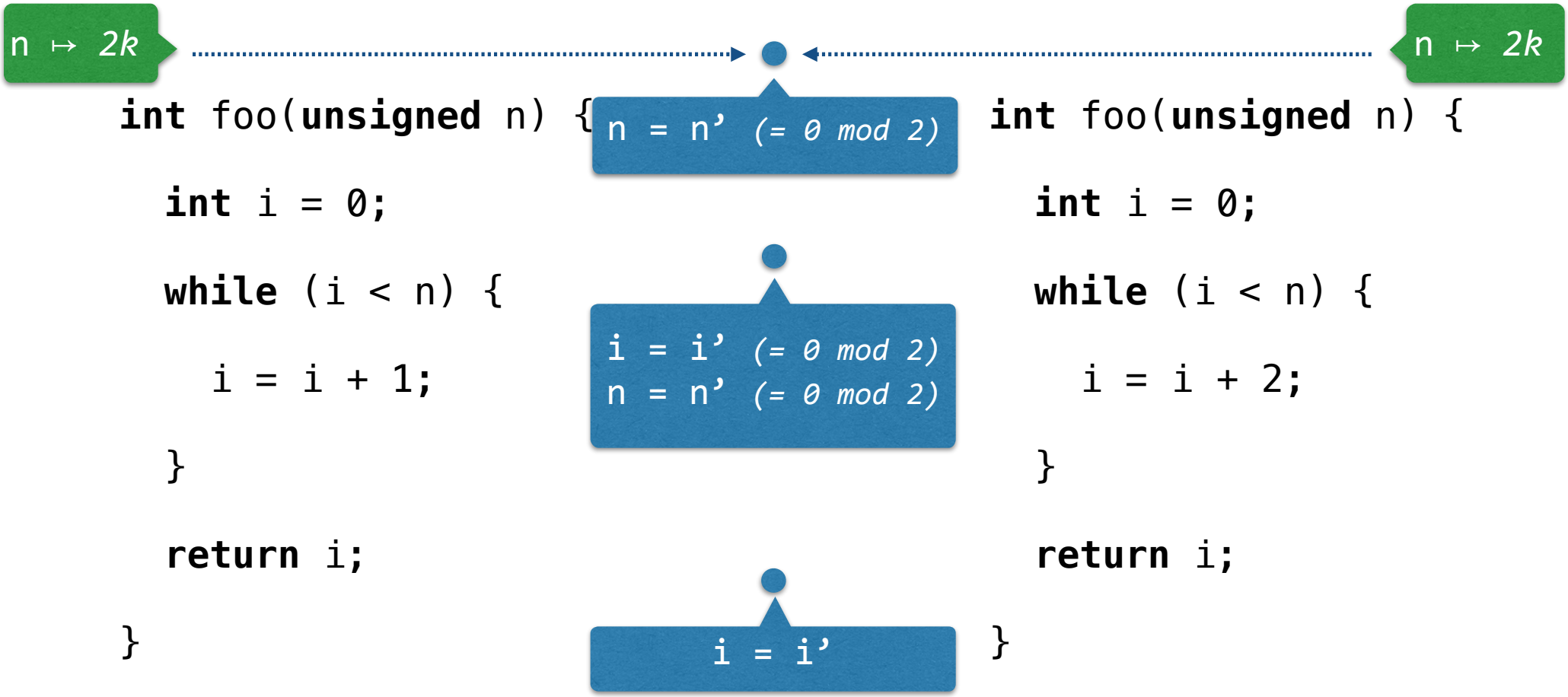
n = n' *(= 0 mod 2)*

i = i' *(= 0 mod 2)*
n = n' *(= 0 mod 2)*

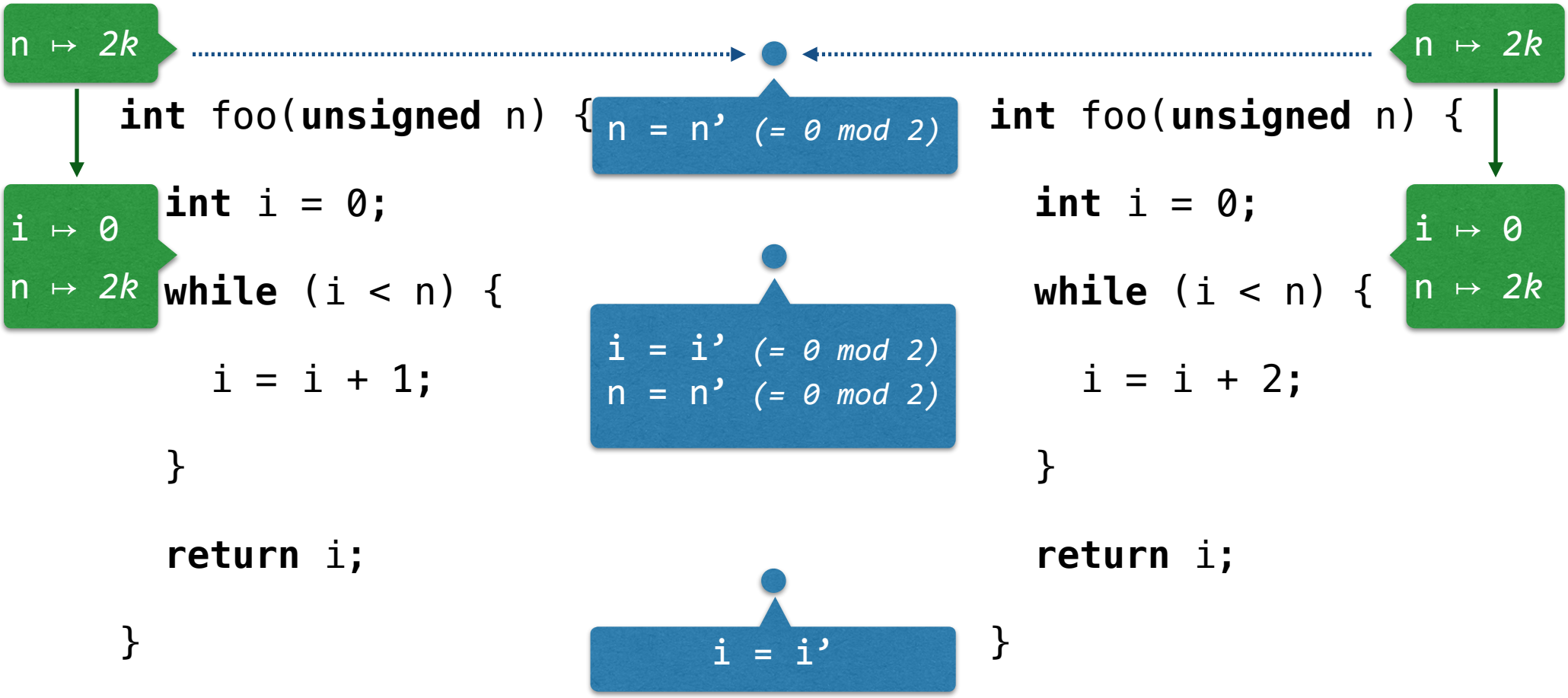
i = i'

```
int foo(unsigned n) {
  int i = 0;
  while (i < n) {
    i = i + 2;
  }
  return i;
}
```

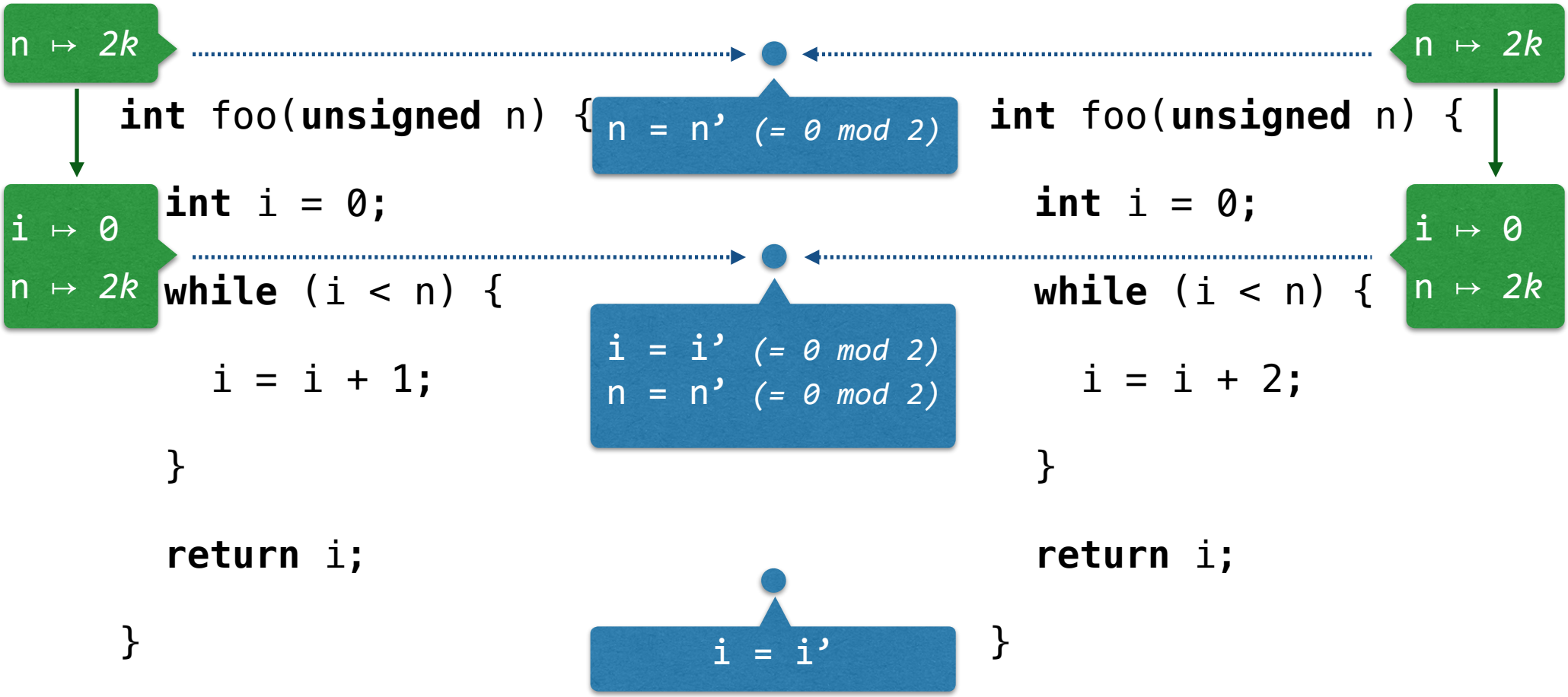
Example


n ↦ 2k
n ↦ 2k
int foo(unsigned n) {
int i = 0;
while (i < n) {
i = i + 1;
}
return i;
}
n = n' (= 0 mod 2)
i = i' (= 0 mod 2)
n = n' (= 0 mod 2)
i = i'
int foo(unsigned n) {
int i = 0;
while (i < n) {
i = i + 2;
}
return i;
}

Example


n ↦ 2k
int foo(unsigned n) {
int i = 0;
i ↦ 0
n ↦ 2k
while (i < n) {
i = i + 1;
}
return i;
}
n = n' (= 0 mod 2)
i = i' (= 0 mod 2)
n = n' (= 0 mod 2)
i = i'
n ↦ 2k
int foo(unsigned n) {
int i = 0;
while (i < n) {
i = i + 2;
}
return i;
}
i ↦ 0
n ↦ 2k

Example


n ↦ 2k
int foo(unsigned n) {
n = n' (= 0 mod 2)
n ↦ 2k
int foo(unsigned n) {
i ↦ 0
n ↦ 2k
int i = 0;
int i = 0;
i ↦ 0
n ↦ 2k
while (i < n) {
i = i' (= 0 mod 2)
n = n' (= 0 mod 2)
while (i < n) {
i = i + 1;
i = i + 2;
}
}
return i;
return i;
i = i'
}
}

Example

```
int foo(unsigned n) {
  int i = 0;
  while (i < n) {
    i = i + 1;
  }
  return i;
}
```

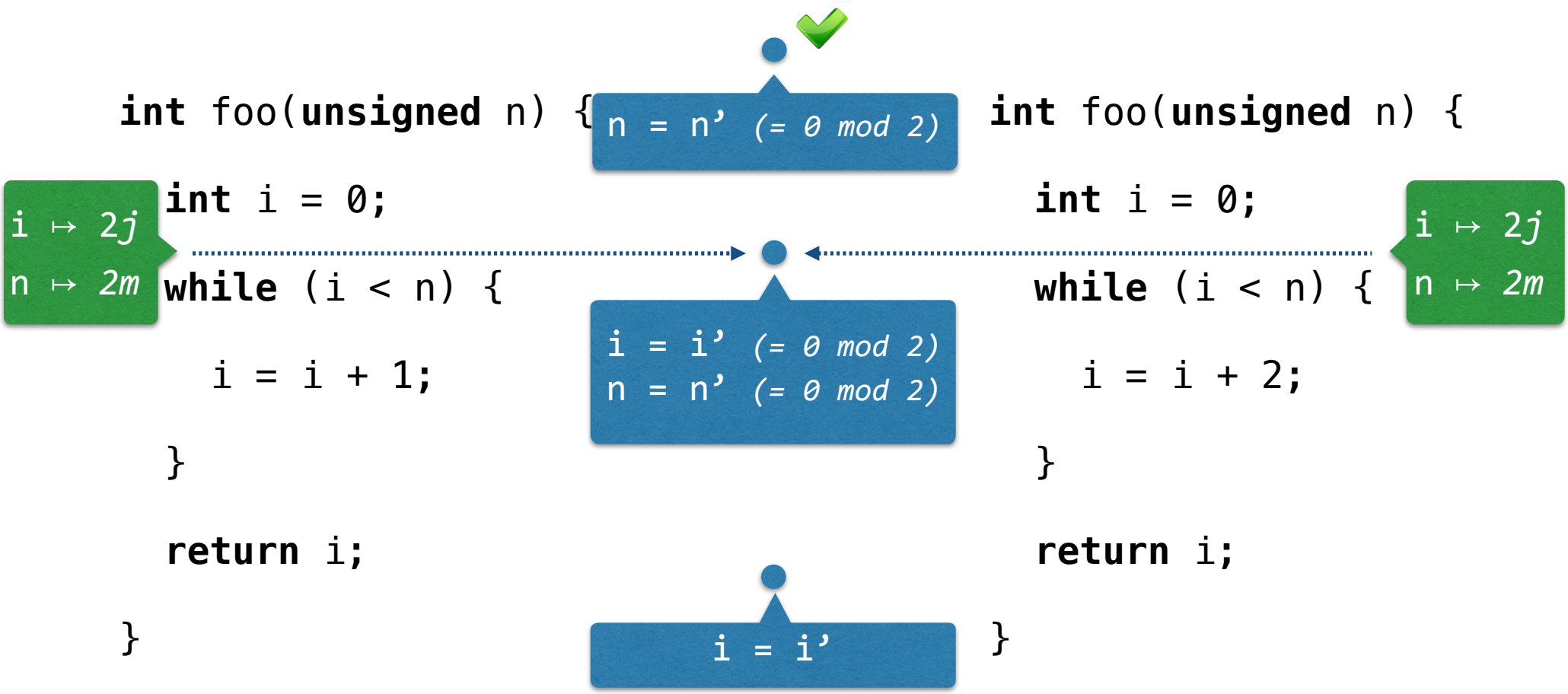
n = n' *(= 0 mod 2)*

i = i' *(= 0 mod 2)*
n = n' *(= 0 mod 2)*

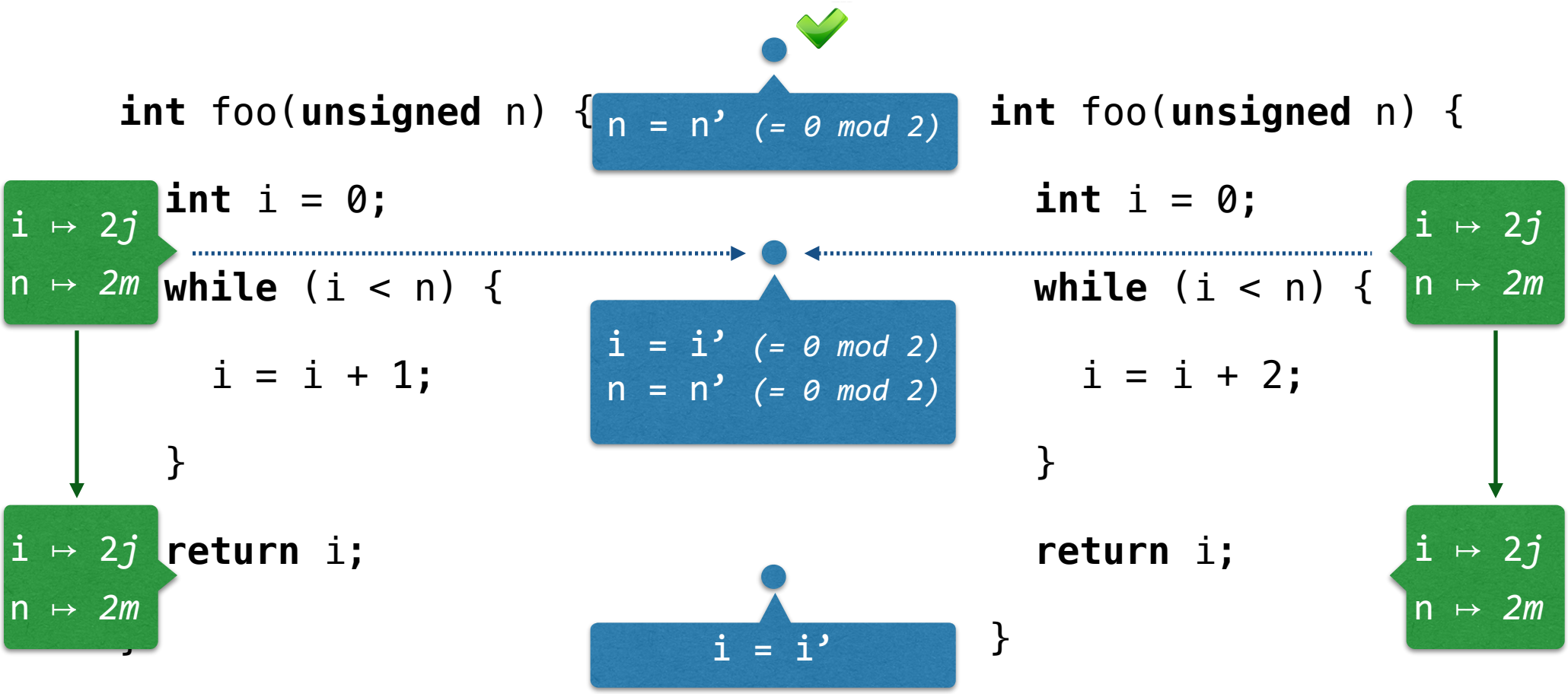
i = i'

```
int foo(unsigned n) {
  int i = 0;
  while (i < n) {
    i = i + 2;
  }
  return i;
}
```

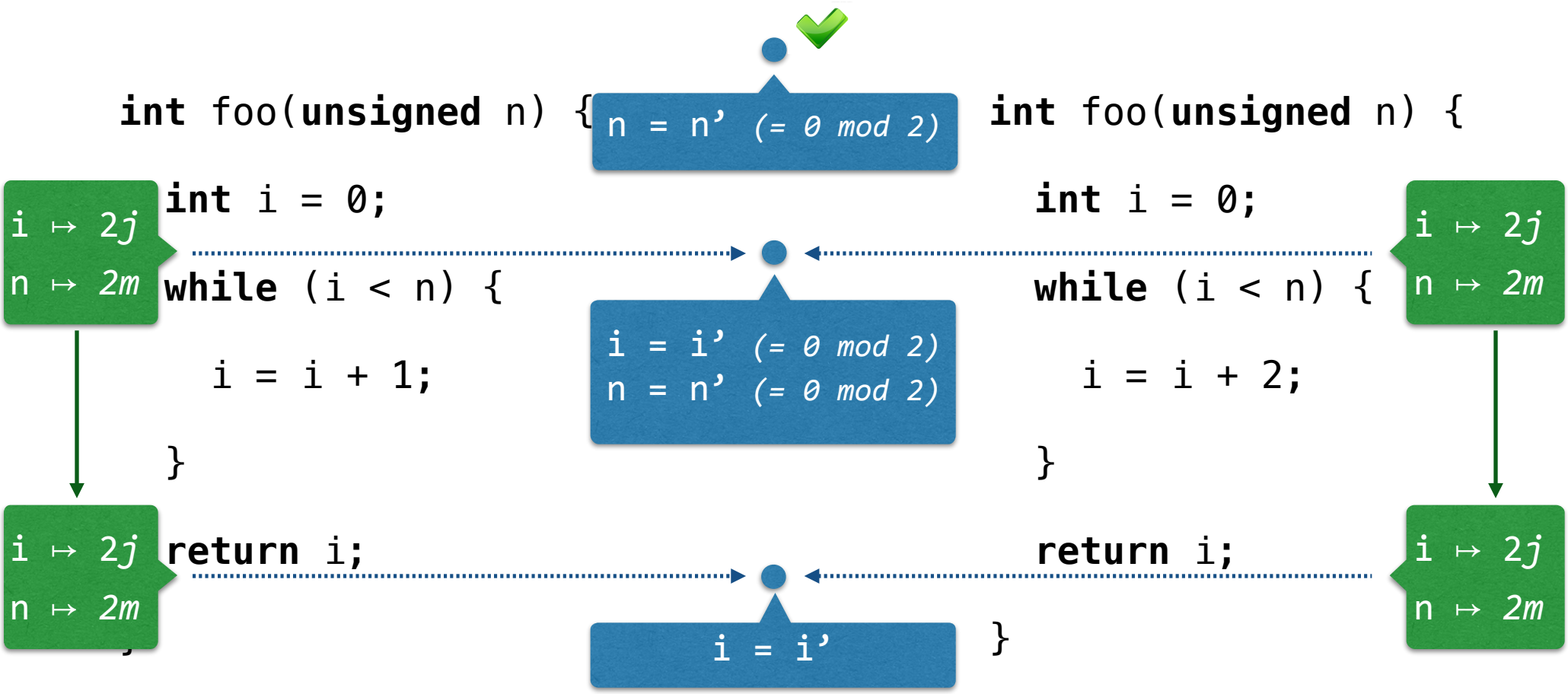
Example


int foo(unsigned n) {
int i = 0;
while (i < n) {
i = i + 1;
}
return i;
}
n = n' (= 0 mod 2)
i = i' (= 0 mod 2)
n = n' (= 0 mod 2)
i = i'
i ↦ 2j
n ↦ 2m
int foo(unsigned n) {
int i = 0;
while (i < n) {
i = i + 2;
}
return i;
}
i ↦ 2j
n ↦ 2m

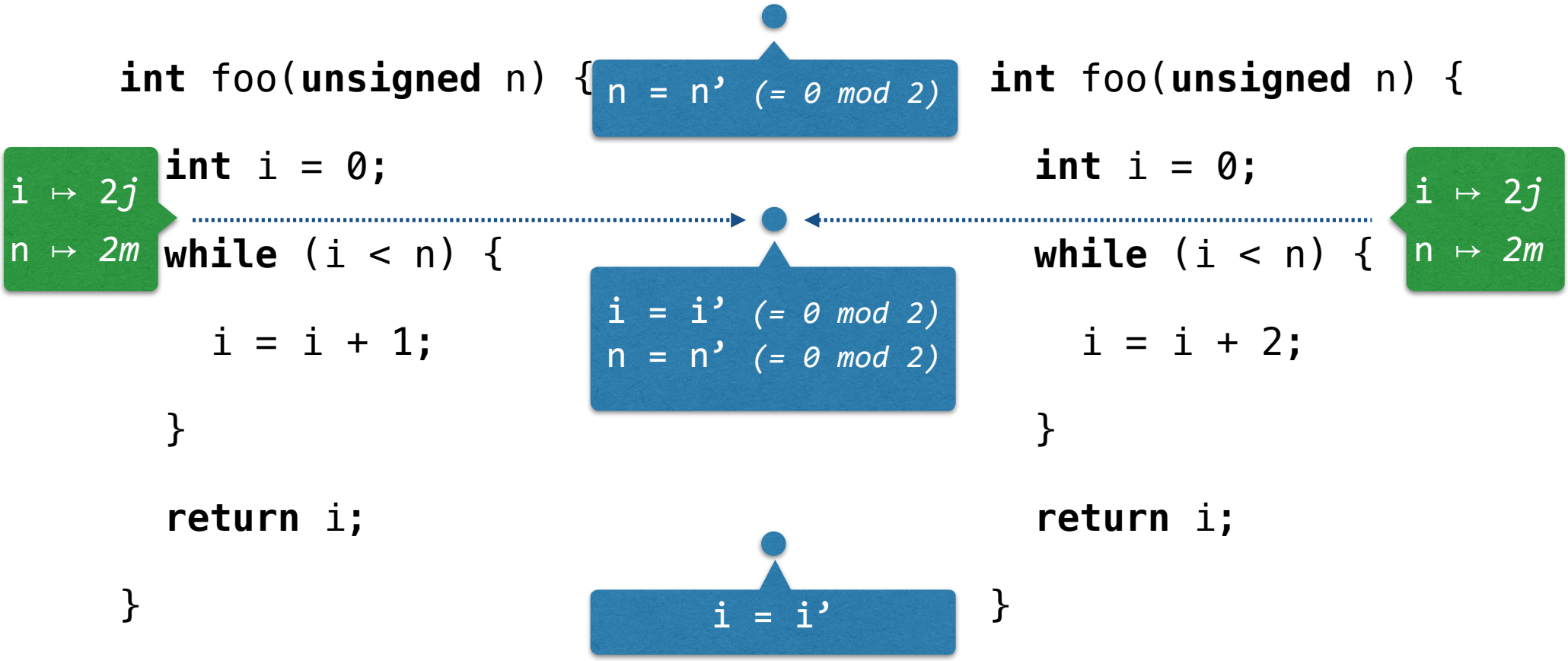
Example


int foo(unsigned n) {
int i = 0;
while (i < n) {
i = i + 1;
}
return i;
}
int foo(unsigned n) {
int i = 0;
while (i < n) {
i = i + 2;
}
return i;
}
n = n' (= 0 mod 2)
i = i' (= 0 mod 2)
n = n' (= 0 mod 2)
i = i'
i ↦ 2j
n ↦ 2m
i ↦ 2j
n ↦ 2m
i ↦ 2j
n ↦ 2m
i ↦ 2j
n ↦ 2m

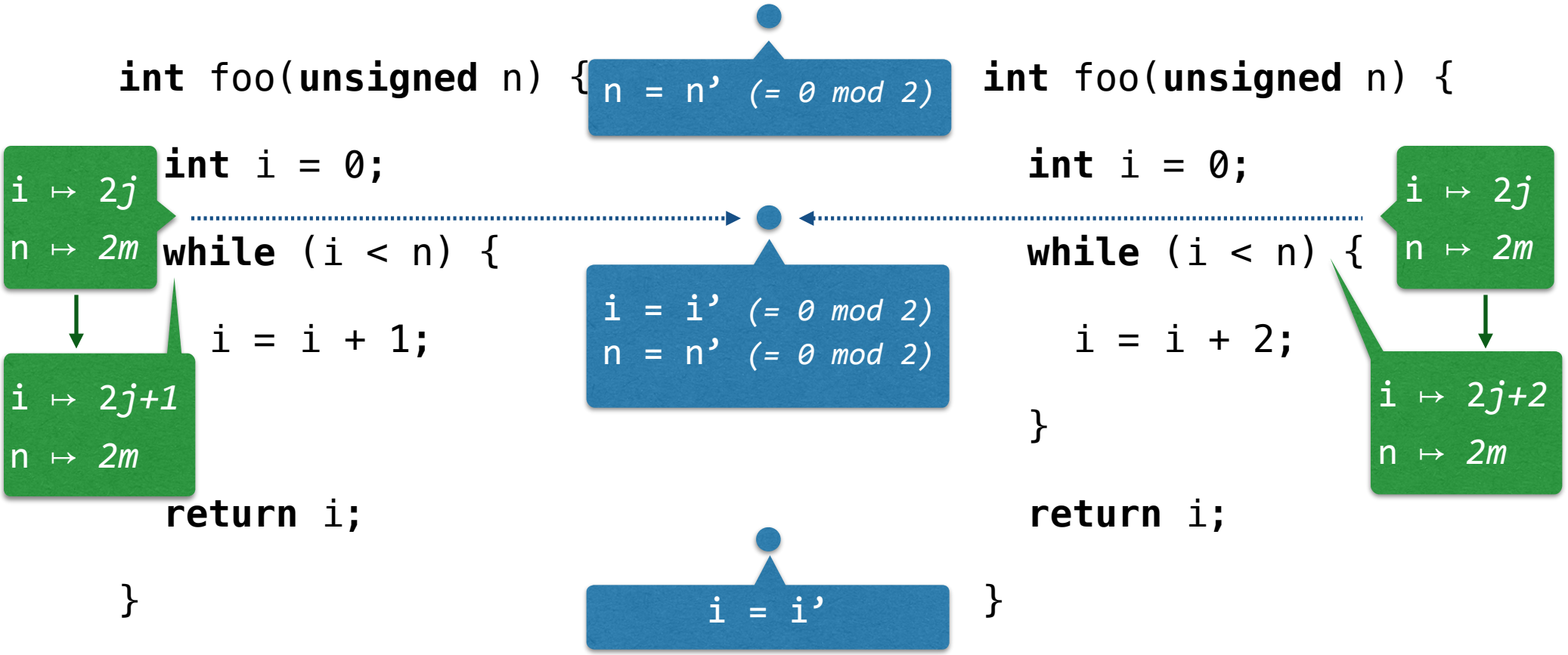
Example


int foo(unsigned n) {
int i = 0;
while (i < n) {
i = i + 1;
}
return i;
}
int foo(unsigned n) {
int i = 0;
while (i < n) {
i = i + 2;
}
return i;
}
n = n' (= 0 mod 2)
i = i' (= 0 mod 2)
n = n' (= 0 mod 2)
i = i'
i ↦ 2j
n ↦ 2m
i ↦ 2j
n ↦ 2m
i ↦ 2j
n ↦ 2m
i ↦ 2j
n ↦ 2m

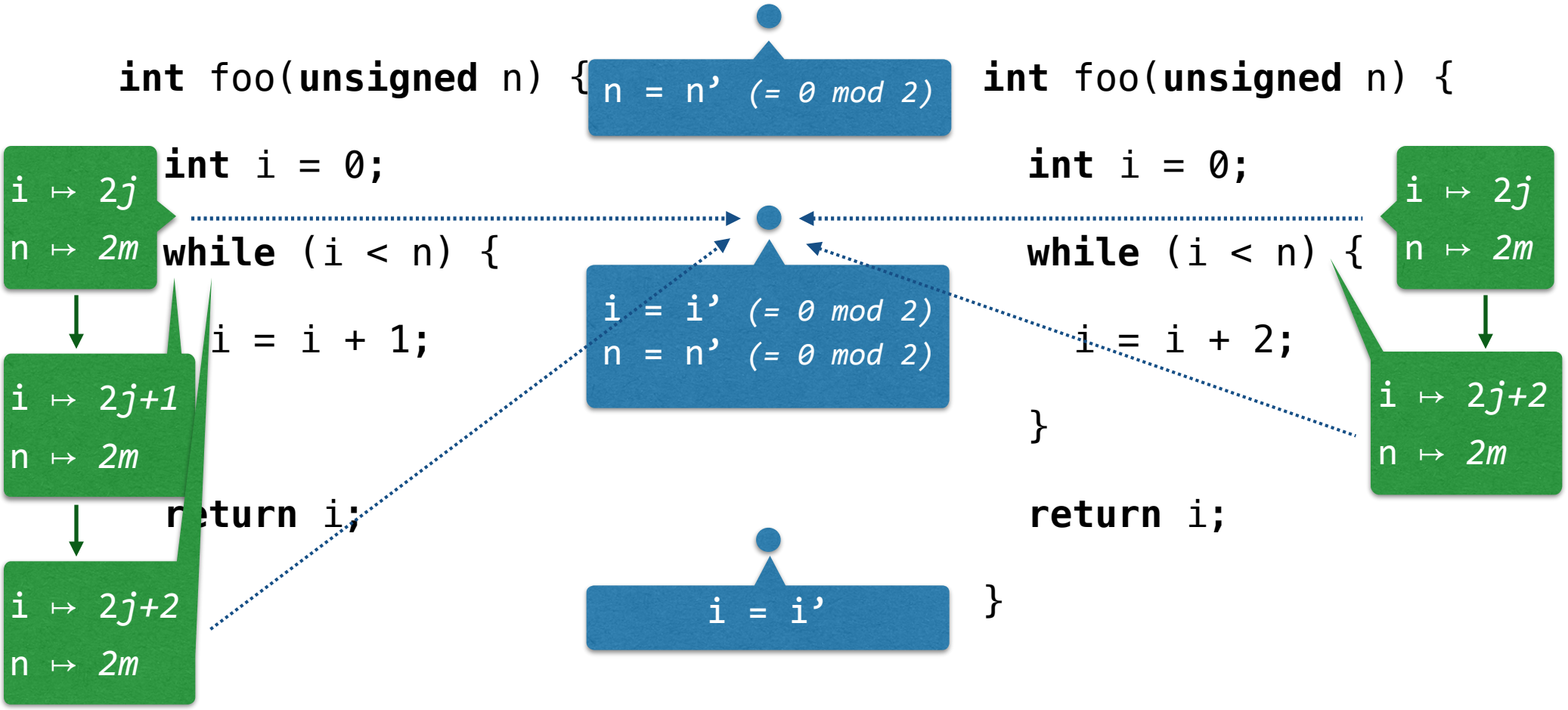
Example


int foo(unsigned n) {
int i = 0;
while (i < n) {
i = i + 1;
}
return i;
}
n = n' (= 0 mod 2)
i = i' (= 0 mod 2)
n = n' (= 0 mod 2)
i = i'
i ↦ 2j
n ↦ 2m
int foo(unsigned n) {
int i = 0;
while (i < n) {
i = i + 2;
}
return i;
}
i ↦ 2j
n ↦ 2m

Example


int foo(unsigned n) {
int i = 0;
while (i < n) {
i = i + 1;
return i;
}
int foo(unsigned n) {
int i = 0;
while (i < n) {
i = i + 2;
}
return i;
}
n = n' (= 0 mod 2)
i = i' (= 0 mod 2)
n = n' (= 0 mod 2)
i = i'
i ↦ 2j
n ↦ 2m
i ↦ 2j+1
n ↦ 2m
i ↦ 2j
n ↦ 2m
i ↦ 2j+2
n ↦ 2m

Example


int foo(unsigned n) {
int i = 0;
while (i < n) {
i = i + 1;
return i;
}
n = n' (= 0 mod 2)
i = i' (= 0 mod 2)
n = n' (= 0 mod 2)
i = i'
i ↦ 2j
n ↦ 2m
i ↦ 2j+1
n ↦ 2m
i ↦ 2j+2
n ↦ 2m
int foo(unsigned n) {
int i = 0;
while (i < n) {
i = i + 2;
}
return i;
}
i ↦ 2j
n ↦ 2m
i ↦ 2j+2
n ↦ 2m

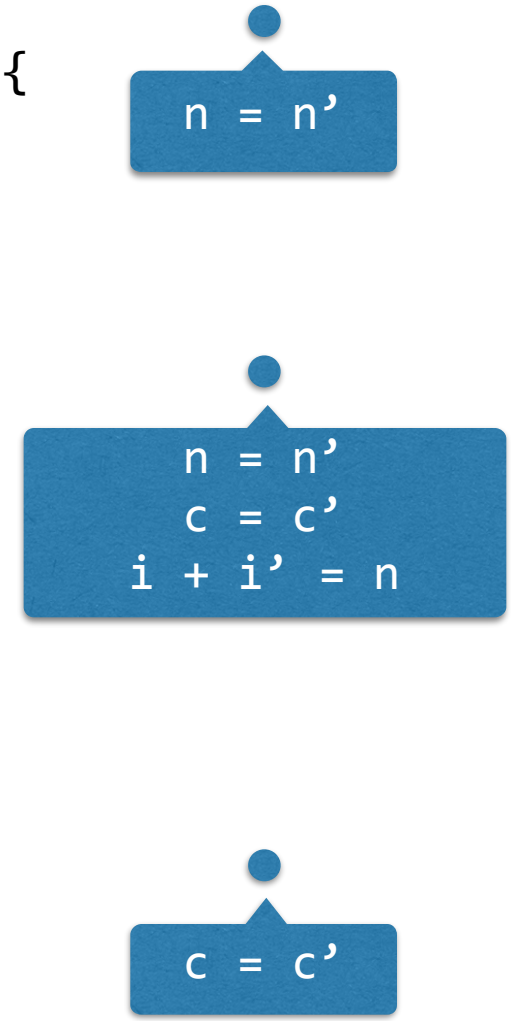
Example (out-of-order loop iter.)

```
int cnt(unsigned n) {
  int c = 0;
  int i = 0;
  while (i < n) {
    i = i + 1;
    c = c + 1;
  }
  return c;
}
```

```
int cnt(unsigned n) {
  int c = 0;
  int i = n;
  while (i > 0) {
    i = i - 1;
    c = c + 1;
  }
  return c;
}
```

Example (out-of-order loop iter.)

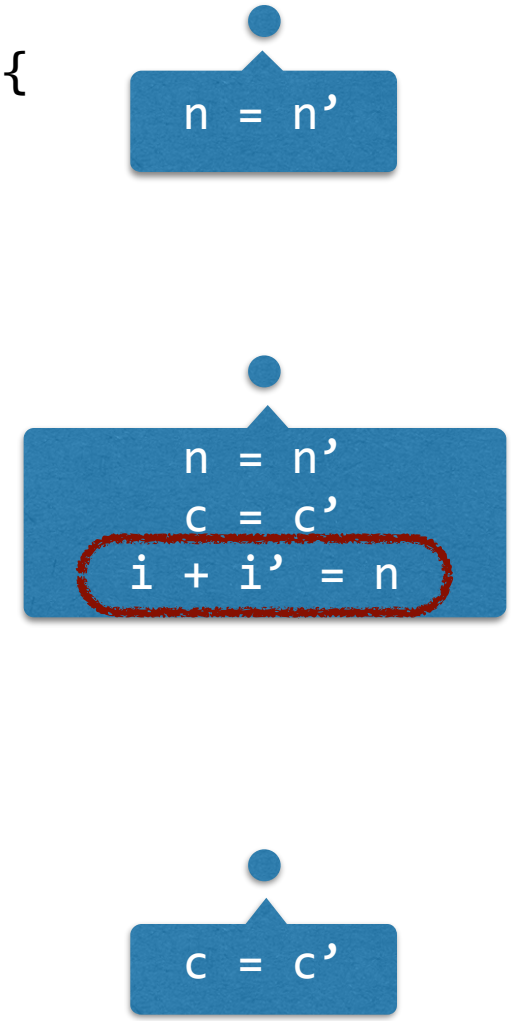
```
int cnt(unsigned n) {
  int c = 0;
  int i = 0;
  while (i < n) {
    i = i + 1;
    c = c + 1;
  }
  return c;
}
```




```
int cnt(unsigned n) {
  int c = 0;
  int i = n;
  while (i > 0) {
    i = i - 1;
    c = c + 1;
  }
  return c;
}
```

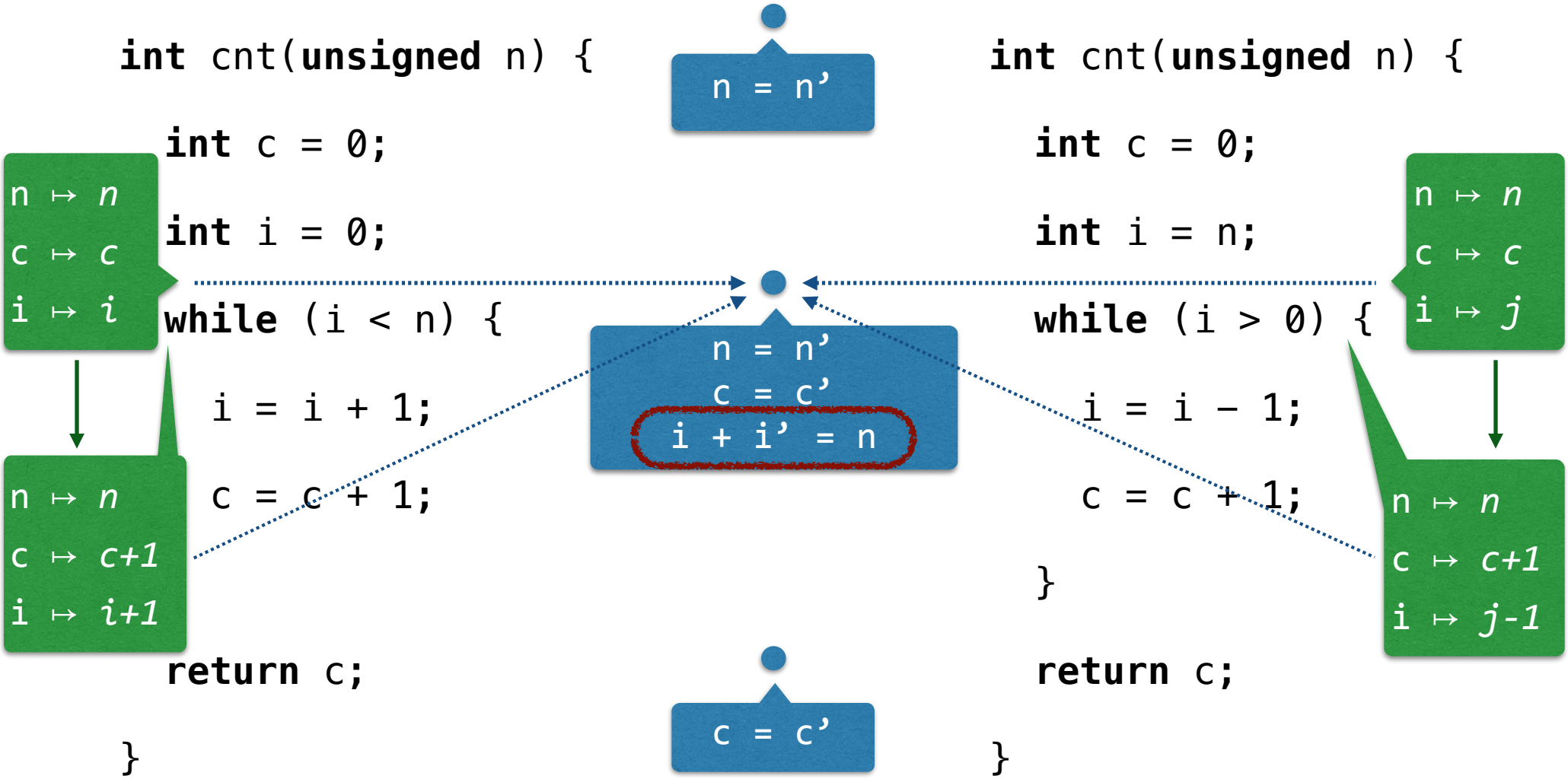
Example (out-of-order loop iter.)

```
int cnt(unsigned n) {
  int c = 0;
  int i = 0;
  while (i < n) {
    i = i + 1;
    c = c + 1;
  }
  return c;
}
```

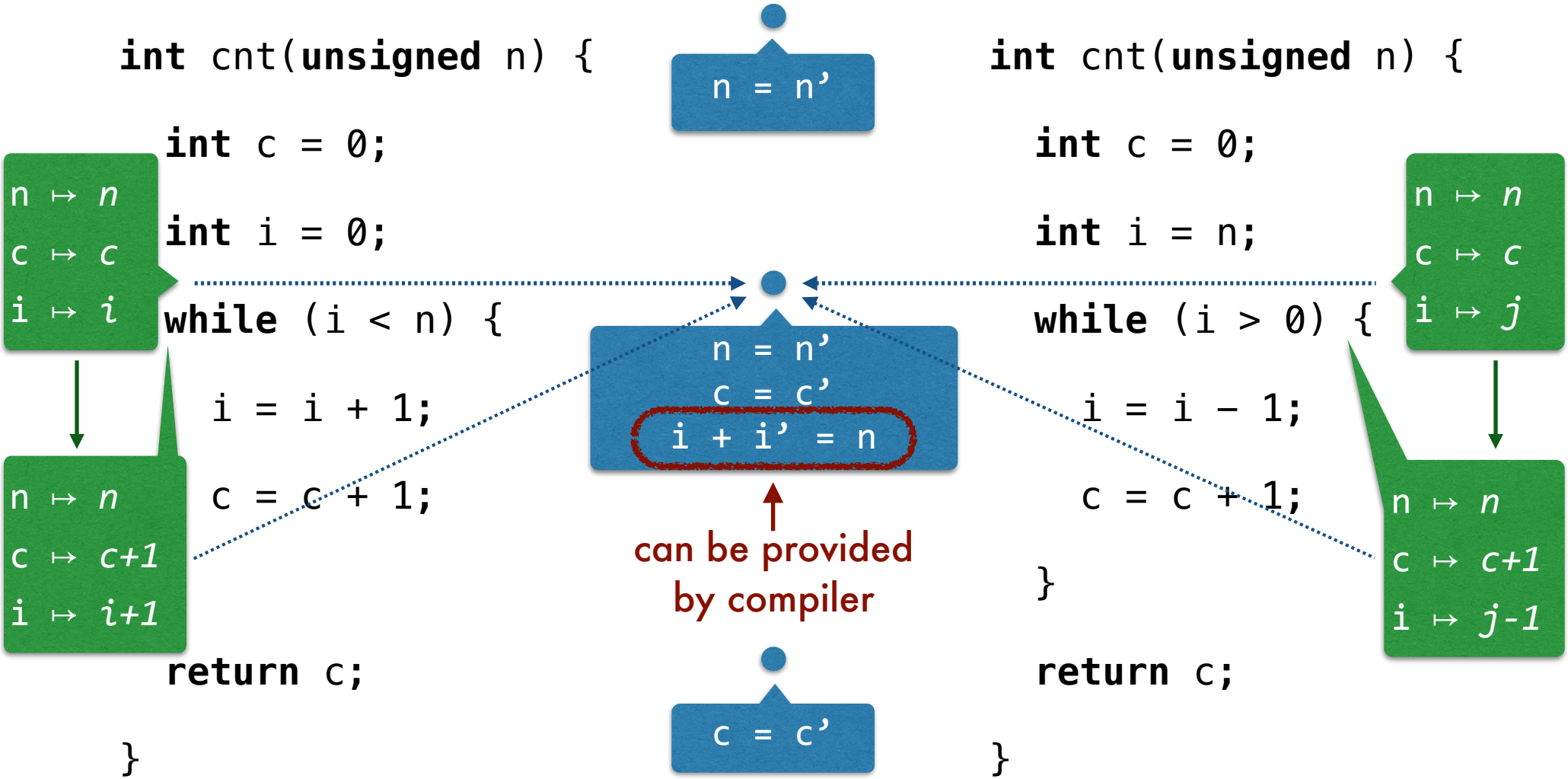



```
int cnt(unsigned n) {
  int c = 0;
  int i = n;
  while (i > 0) {
    i = i - 1;
    c = c + 1;
  }
  return c;
}
```

Example (out-of-order loop iter.)



Example (out-of-order loop iter.)



Checking compatibility

```
function transfer(address to, uint256 value)
      public returns (bool) {
  balances[msg.sender] = balances[msg.sender].sub(value);
  balances[to] = balances[to].add(value);
  emit Transfer(msg.sender, to, value);
  return true;
}
```

Solidity

```
@public
def transfer(to : address, value : uint256(wei)) -> bool:
  sender: address = msg.sender
  self.balances[sender] = self.balances[sender] - value
  self.balances[to] = self.balances[to] + value
  log.Transfer(sender, to, value)
  return True
```

Vyper

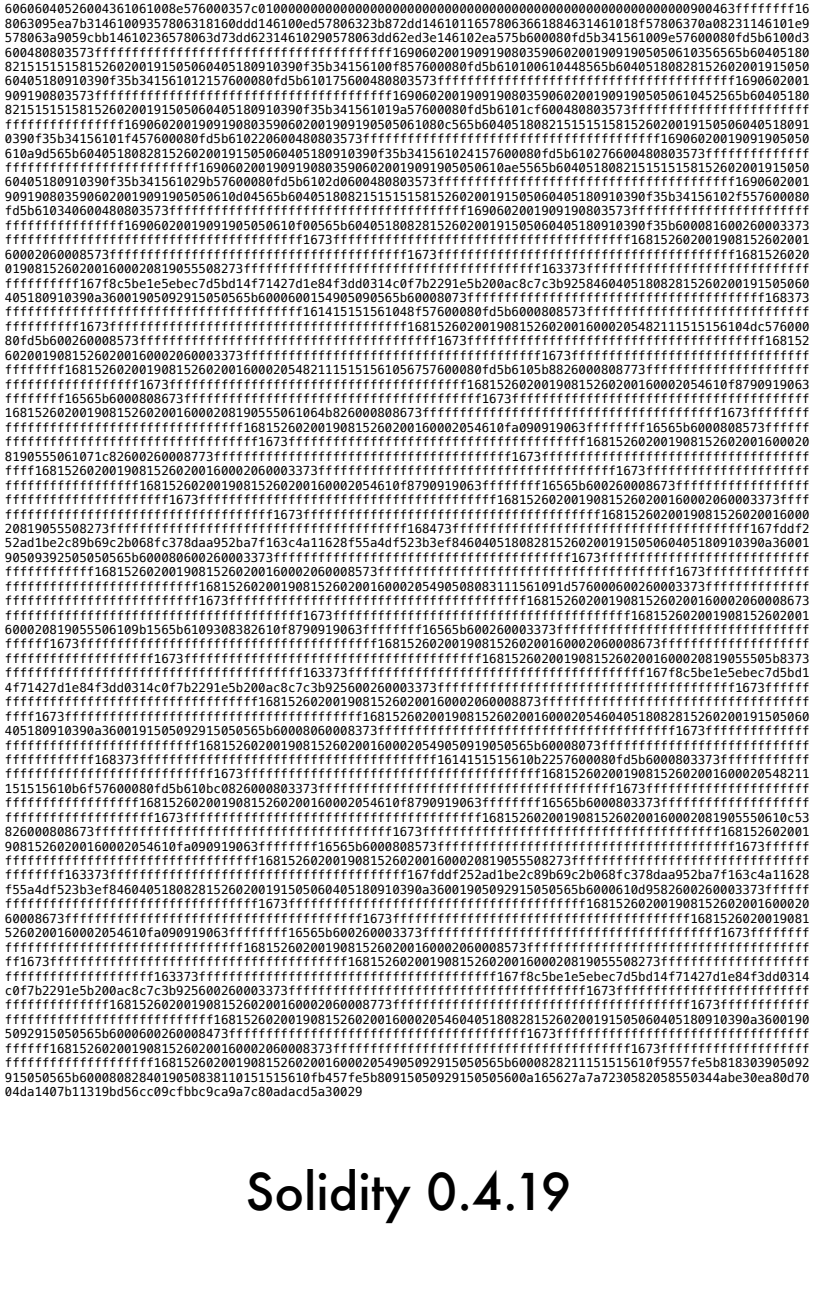
Checking compatibility

```
function transfer(address to, uint256 value)
      public returns (bool) {
  balances[msg.sender] = balances[msg.sender].sub(value);
  balances[to] = balances[to].add(value);
  emit Transfer(msg.sender, to, value);
  return true;
}
```

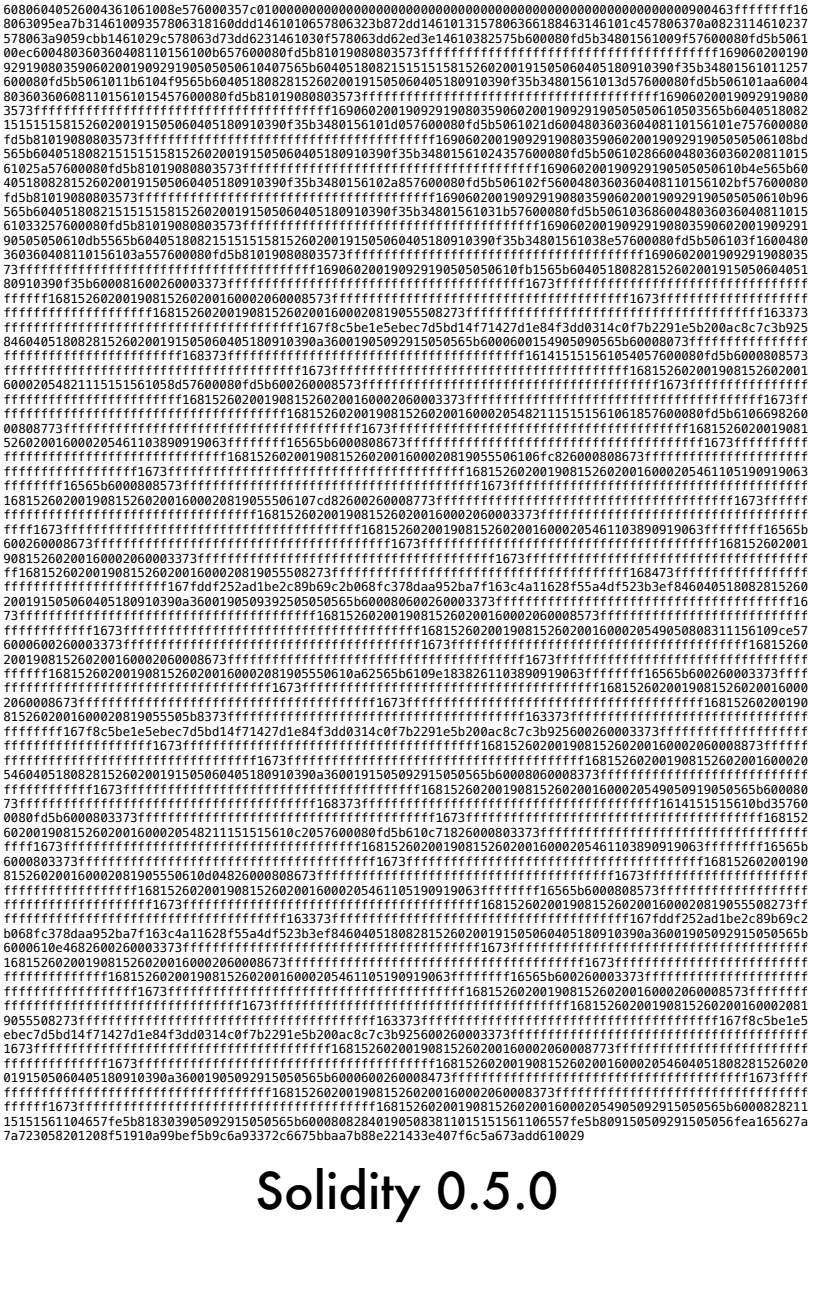
Solidity 0.4

Solidity 0.5

Checking compatibility



Solidity 0.4.19



Solidity 0.5.0

Checking compatibility

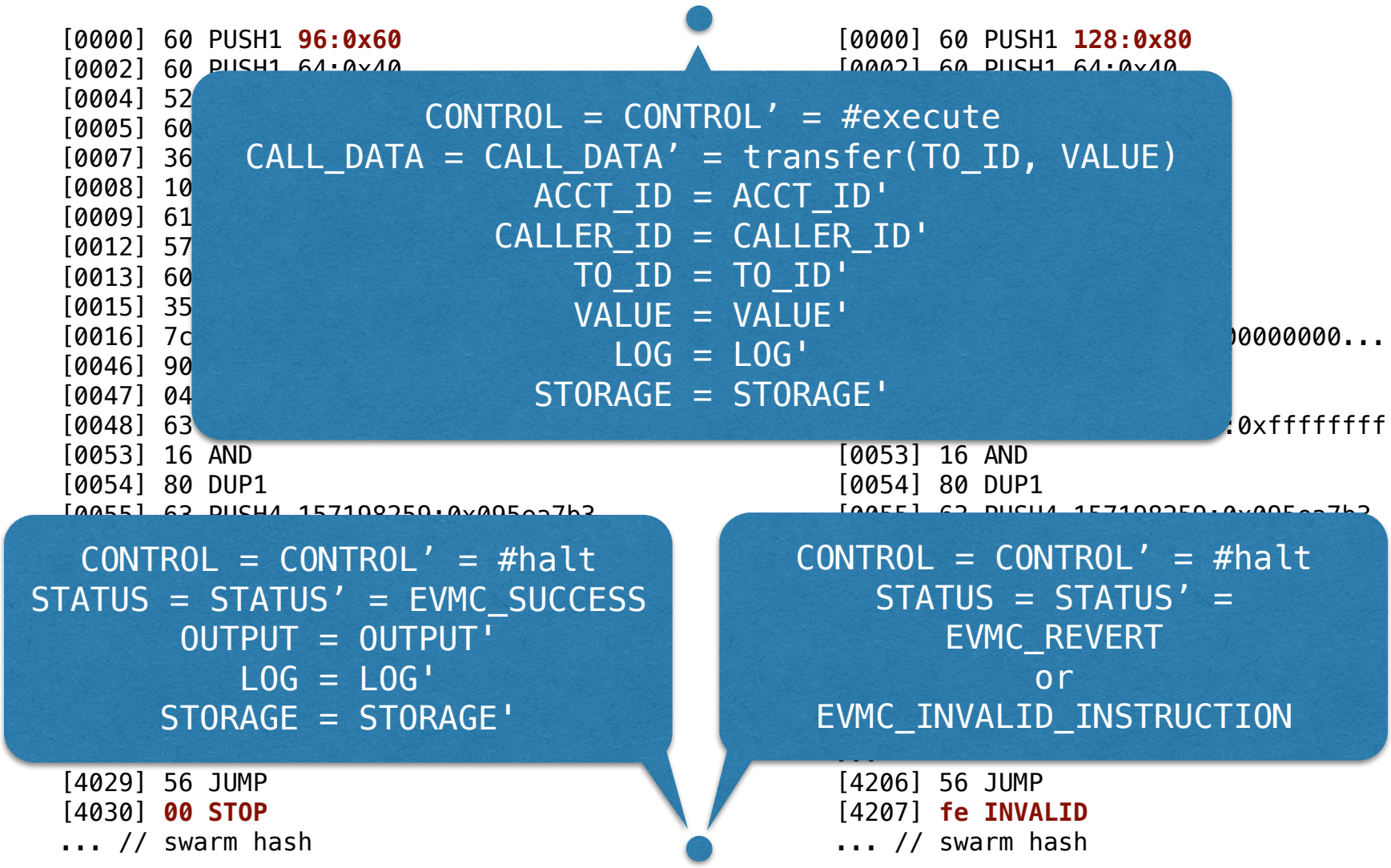
```
[0000] 60 PUSH1 96:0x60
[0002] 60 PUSH1 64:0x40
[0004] 52 MSTORE
[0005] 60 PUSH1 4:0x04
[0007] 36 CALLDATASIZE
[0008] 10 LT
[0009] 61 PUSH2 142:0x008e
[0012] 57 JUMPI
[0013] 60 PUSH1 0:0x00
[0015] 35 CALLDATALOAD
[0016] 7c PUSH29 0x010000000000000...
[0046] 90 SWAP1
[0047] 04 DIV
[0048] 63 PUSH4 4294967295:0xffffffff
[0053] 16 AND
[0054] 80 DUP1
[0055] 63 PUSH4 157198259:0x095ea7b3
[0060] 14 EQ
[0061] 61 PUSH2 147:0x0093
[0064] 57 JUMPI
[0065] 80 DUP1
[0066] 63 PUSH4 404098525:0x18160ddd
[0071] 14 EQ
[0072] 61 PUSH2 237:0x00ed
...
[4029] 56 JUMP
[4030] 00 STOP
... // swarm hash
```

Solidity 0.4.19

```
[0000] 60 PUSH1 128:0x80
[0002] 60 PUSH1 64:0x40
[0004] 52 MSTORE
[0005] 60 PUSH1 4:0x04
[0007] 36 CALLDATASIZE
[0008] 10 LT
[0009] 61 PUSH2 142:0x008e
[0012] 57 JUMPI
[0013] 60 PUSH1 0:0x00
[0015] 35 CALLDATALOAD
[0016] 7c PUSH29 0x010000000000000...
[0046] 90 SWAP1
[0047] 04 DIV
[0048] 63 PUSH4 4294967295:0xffffffff
[0053] 16 AND
[0054] 80 DUP1
[0055] 63 PUSH4 157198259:0x095ea7b3
[0060] 14 EQ
[0061] 61 PUSH2 147:0x0093
[0064] 57 JUMPI
[0065] 80 DUP1
[0066] 63 PUSH4 404098525:0x18160ddd
[0071] 14 EQ
[0072] 61 PUSH2 262:0x0106
...
[4206] 56 JUMP
[4207] fe INVALID
... // swarm hash
```

Solidity 0.5.0

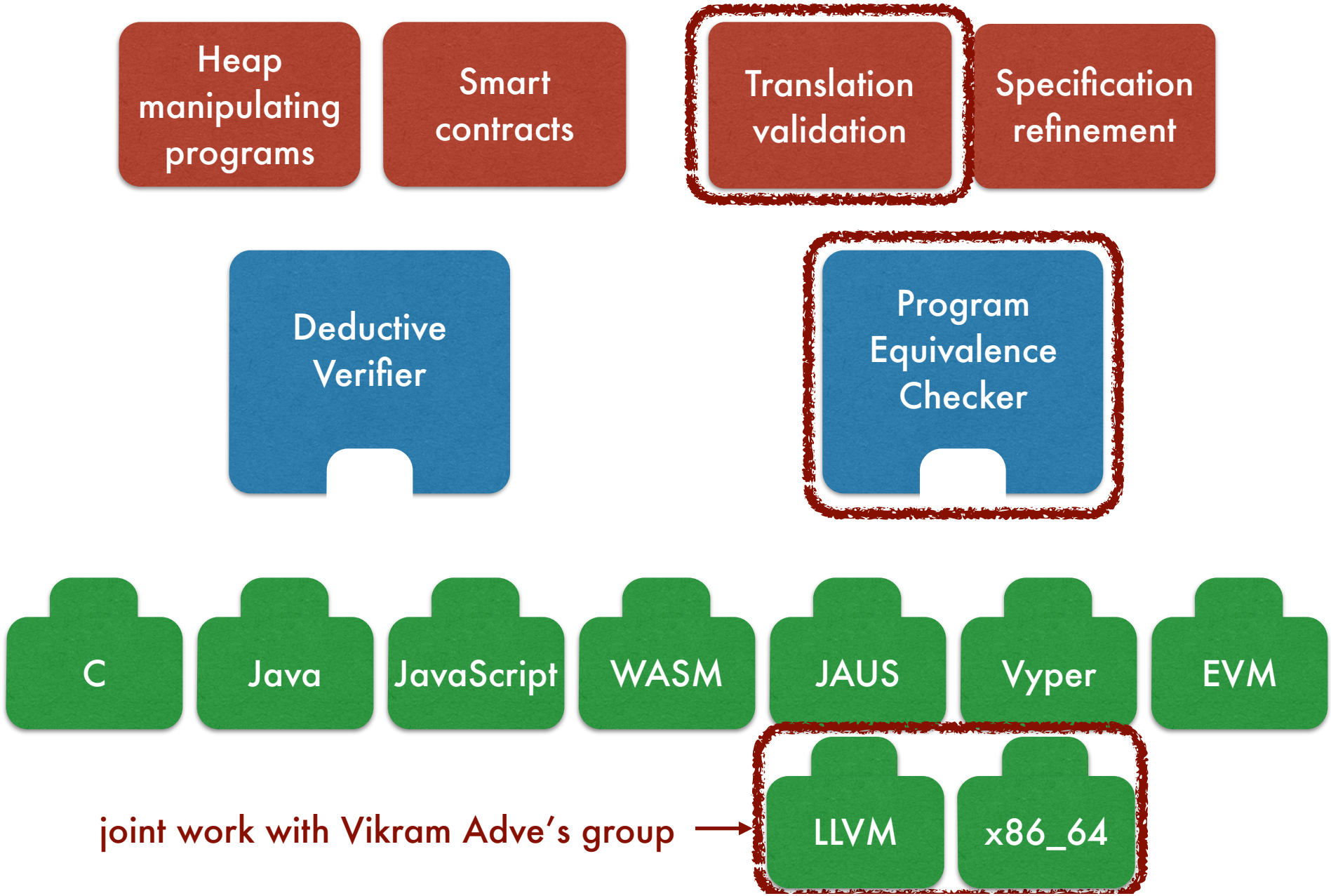
Checking compatibility



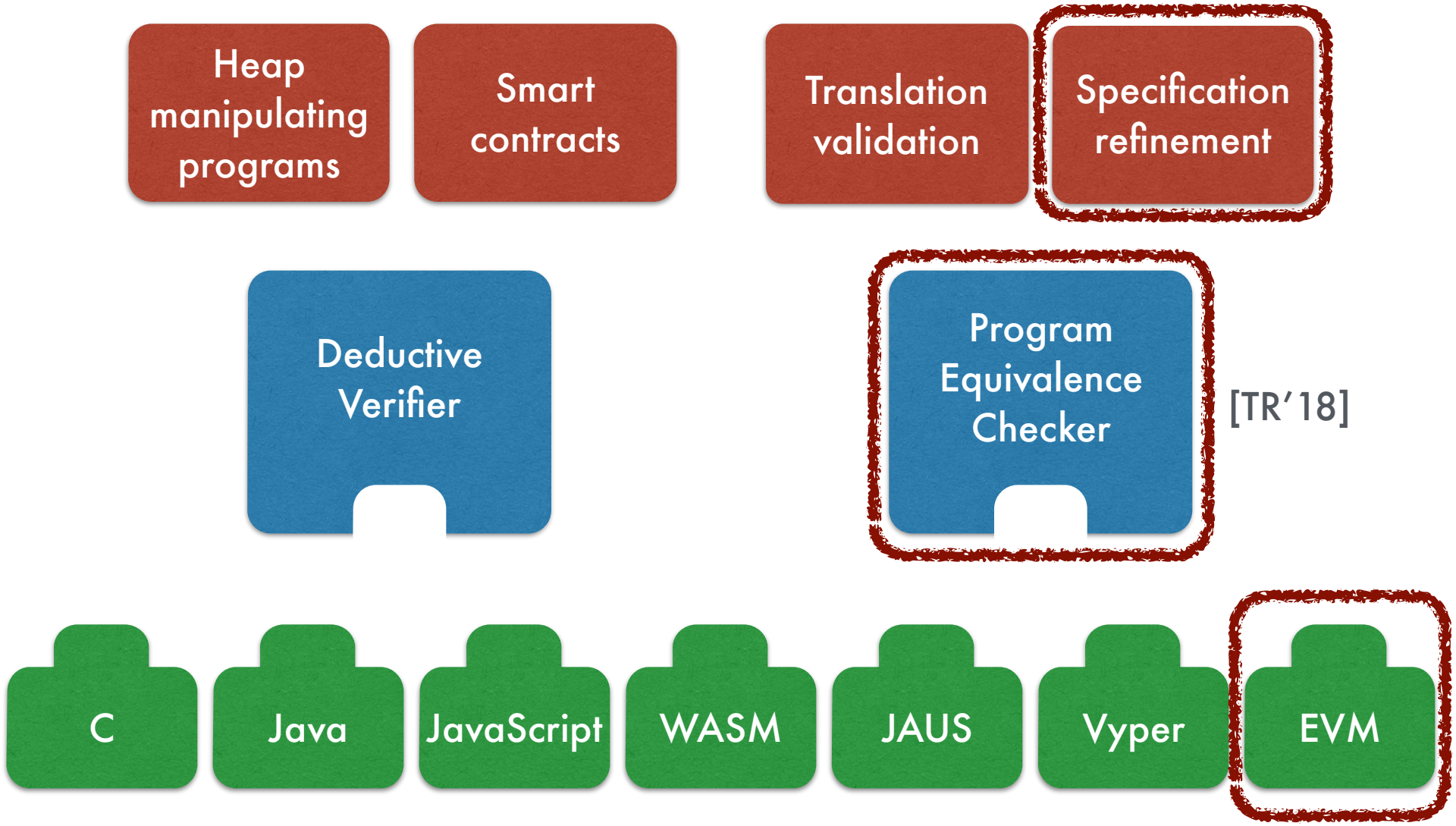
Solidity 0.4.19

Solidity 0.5.0

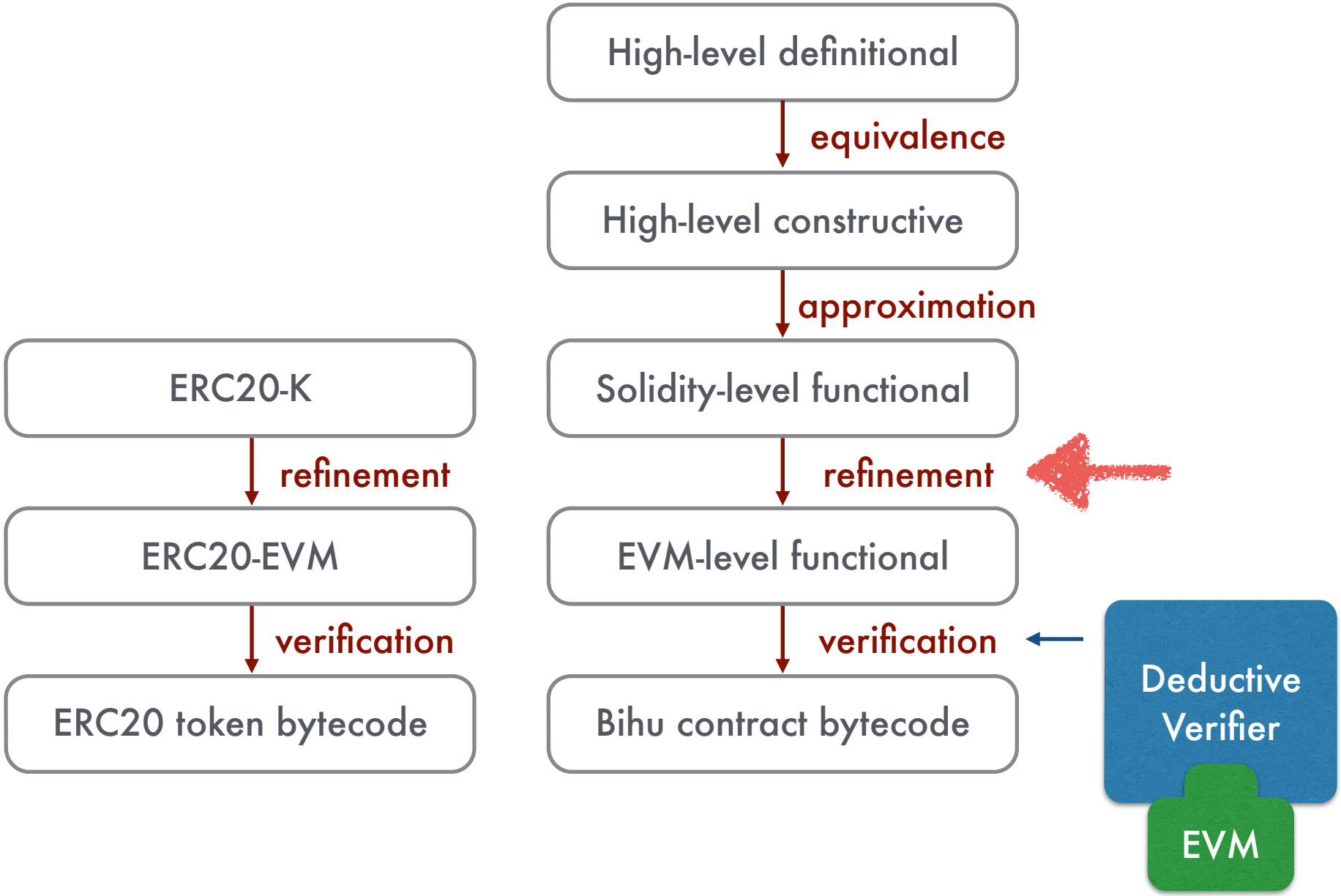
Translation validation for LLVM



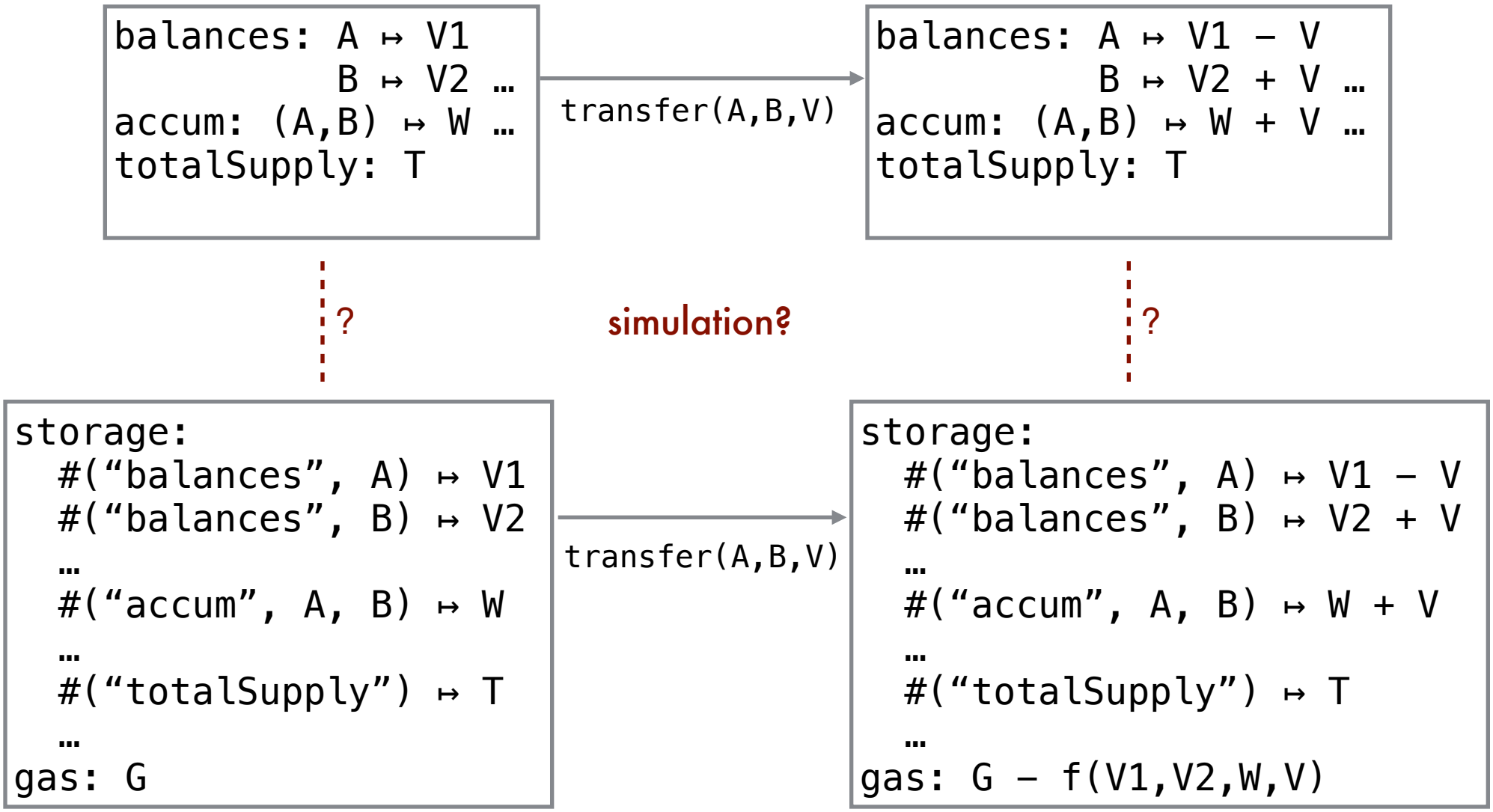
Evaluating new formal method



Validating specification refinement

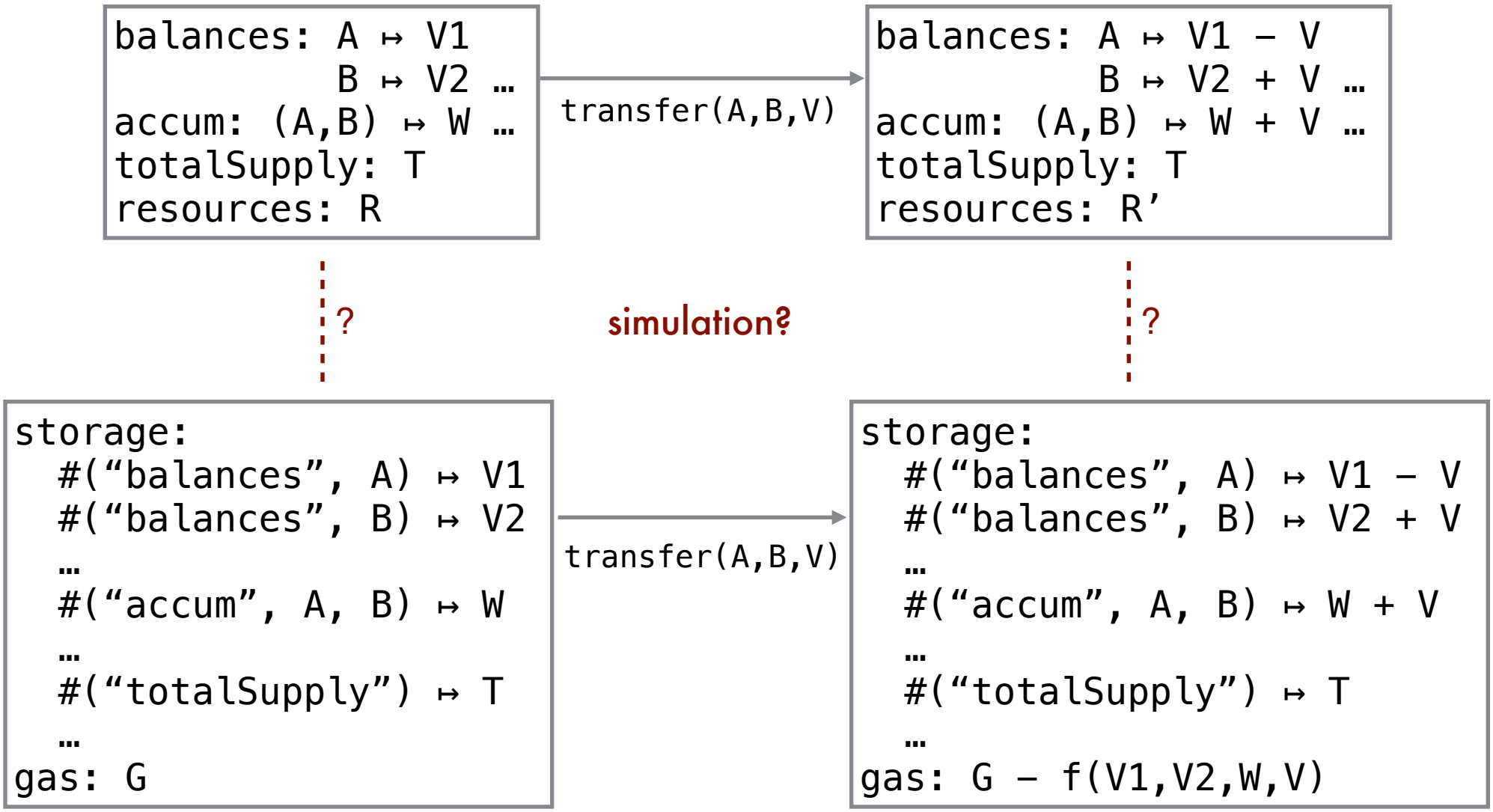


Soundness of specification refinement



e.g., gas consumption is needed for the high-level spec?

Soundness of specification refinement



e.g., gas consumption is needed for the high-level spec?

Soundness of specification refinement

```
balances: A ↦ V1
          B ↦ V2 …
accum: (A,B) ↦ W …
totalSupply: T
resources: R
```

?

```
storage:
  #(“balances”, A) ↦ V1
  #(“balances”, B) ↦ V2
  …
  #(“accum”, A, B) ↦ W
  …
  #(“totalSupply”) ↦ T
  …
gas: G
```

e.g., gas consumptio

$(b, a, t) \sim_s s$

s includes b, a, and t:

$$\forall i \in \mathsf{dom}(b).\ b(i) = s(\mathsf{hash}_b(i))$$
$$\forall (o, s) \in \mathsf{dom}(a).\ a(o, s) = s(\mathsf{hash}_a(o, s))$$
$$t = s(\mathsf{hash}_t)$$

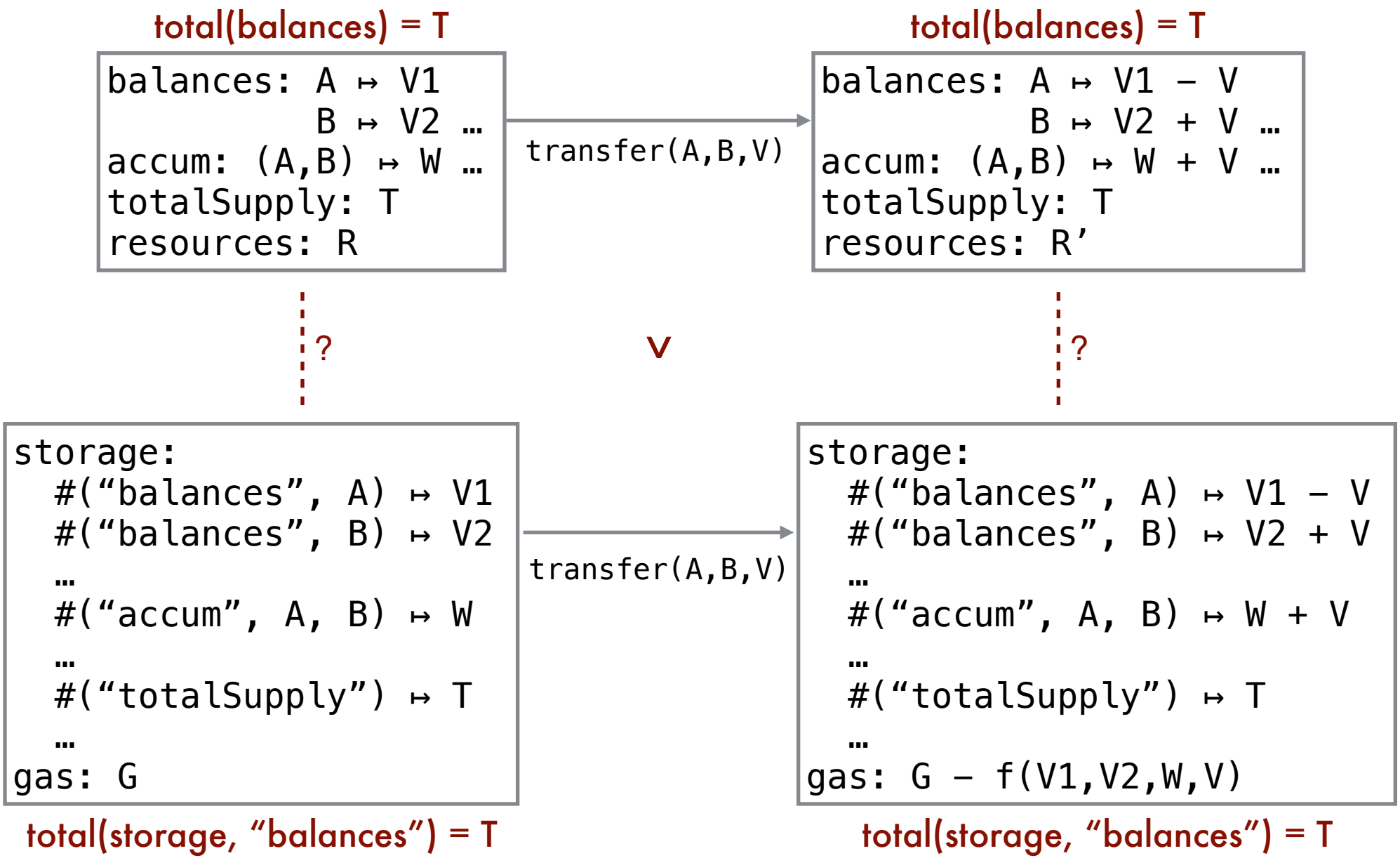
s includes *only* b, a, and t:

$$\forall l.\ s(l) \neq 0 \implies (\ \exists i \in \mathsf{dom}(b).\ \mathsf{hash}_b(i) = l$$
$$\vee\ \exists (o, s) \in \mathsf{dom}(a).\ \mathsf{hash}_a(o, s) = l$$
$$\vee\ \mathsf{hash}_t = l)$$

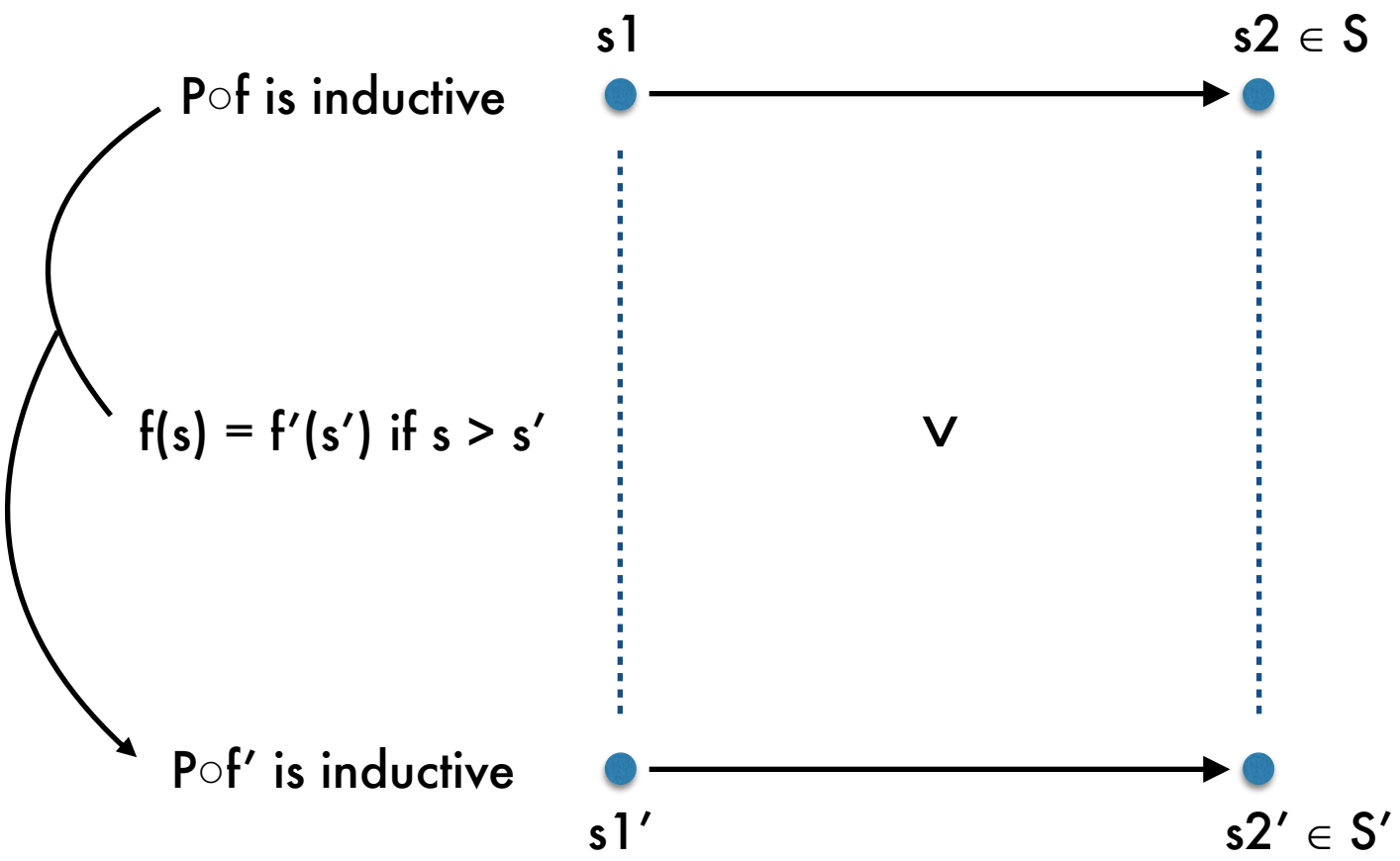
No hash collision:

$$\forall i \in \mathsf{dom}(b), (o, s) \in \mathsf{dom}(a).\ \mathsf{hash}_b(i) \neq \mathsf{hash}_a(o, s) \neq \mathsf{hash}_t \neq$$

Property preservation



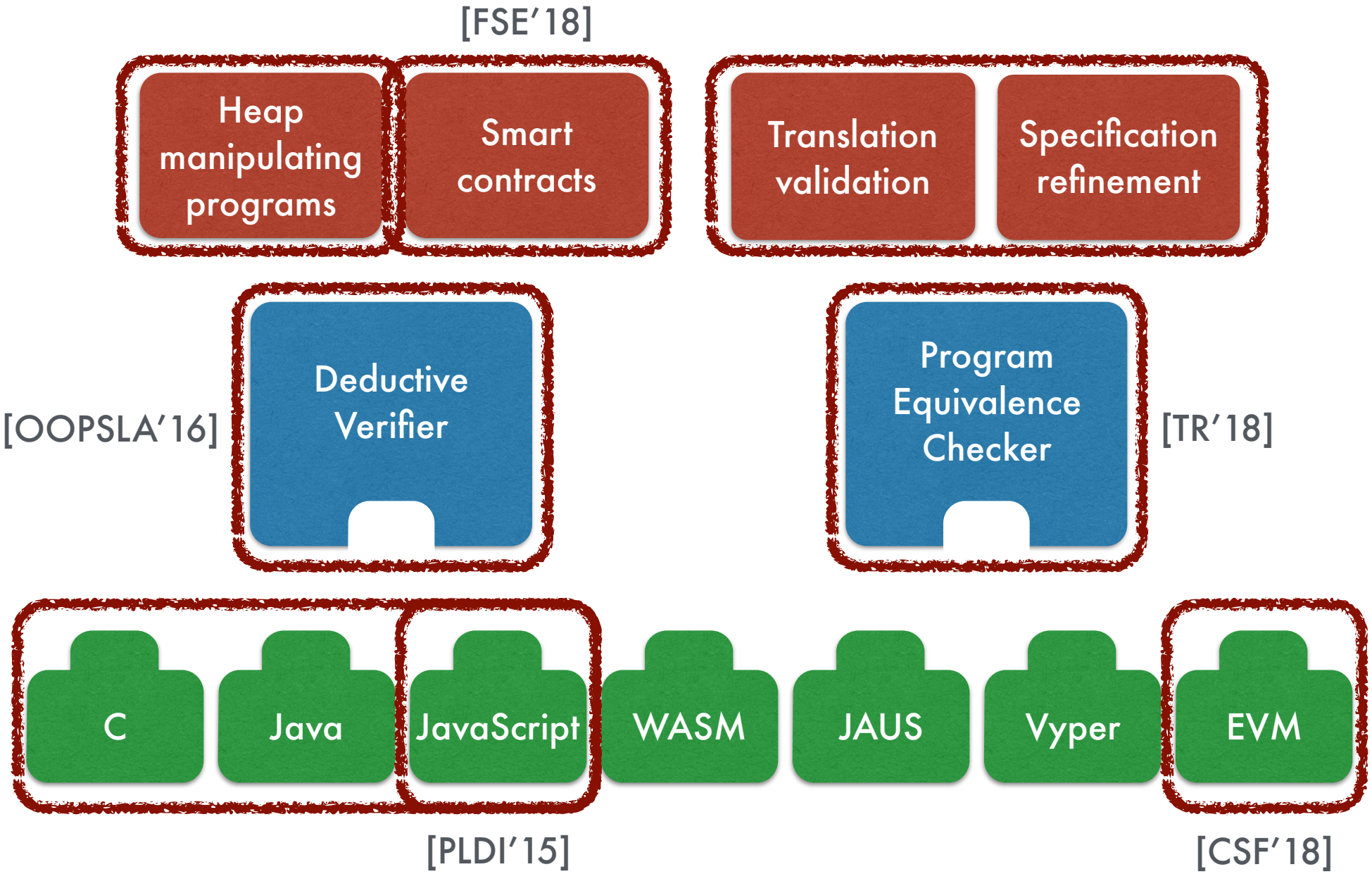
Property preservation



$P : D \to bool$
$f : S \to D$
$f' : S' \to D$

$P \circ f(s) = P(f(s))$
$P \circ f'(s') = P(f'(s'))$

Thesis work


[FSE'18]
Heap manipulating programs
Smart contracts
Translation validation
Specification refinement
[OOPSLA'16]
Deductive Verifier
Program Equivalence Checker
[TR'18]
C
Java
JavaScript
WASM
JAUS
Vyper
EVM
[PLDI'15]
[CSF'18]

Future directions

- Verifying cryptographic primitives
- Translation validation of compilers
- Verifiable computing (for approximate computing)
- Semantics coverage guided fuzzing

Thank you

ERC20-K vs ERC20-EVM configurations

```
<k> $PGM:K </k>
<caller> 0 </caller>
<accounts>
  <account multiplicity="*">
    <id> 0 </id>
    <balance> 0 </balance>
  </account>
</accounts>
<allowances>
  <allowance multiplicity="*">
    <owner> 0 </owner>
    <spenders>
      <allow multiplicity="*">
        <spender> 0 </spender>
        <amount> 0 </amount>
      </allow>
    </spenders>
  </allowance>
</allowances>
<log> Events: </log>
<supply> 0 </supply>
```

```
<k> {K} </k>
<exit-code> 1 </exit-code>
<mode> NORMAL </mode>
<schedule> DEFAULT </schedule> // TODO: pick a right one
<ethereum>
  <evm>
    <output> _ </output>
    <memoryUsed> 0 => _ </memoryUsed>
    <callDepth> CALL_DEPTH </callDepth>
    <callStack> _ => _ </callStack>
    <interimStates> _ </interimStates>
    <substateStack> _ </substateStack>
    <callLog> .Set </callLog> // for vmtest only
    <txExecState>
      <program> #asMapOpCodes(#dasmOpCodes(#parseByteStack({CODE}), DEFAULT)) </program>
      <programBytes> #parseByteStack({CODE}) </programBytes>
      <id> ACCT_ID </id>
      <caller> CALLER_ID </caller>                              |    <previousHash> _ </previousHash>
      <callData> {CALLDATA} </callData>                         |    <ommersHash> _ </ommersHash>
      <callValue> 0 </callValue>                                |    <coinbase> _ </coinbase>
      <wordStack> .WordStack => _ </wordStack>                  |    <stateRoot> _ </stateRoot>
      <localMem> {LOCALMEM} </localMem>                         |    <transactionsRoot> _ </transactionsRoot>
      <pc> 0 => _ </pc>                                         |    <receiptsRoot> _ </receiptsRoot>
      <gas> {GAS} </gas>                                        |    <logsBloom> _ </logsBloom>
      <previousGas> _ => _ </previousGas>                       |    <difficulty> _ </difficulty>
      <static> false </static> // NOTE: non-static call        |    <number> _ </number>
    </txExecState>                                              |    <gasLimit> _ </gasLimit>
    <substate>                                                  |    <gasUsed> _ </gasUsed>
      <selfDestruct> _ </selfDestruct>                          |    <timestamp> _ </timestamp>
      <log> {LOG} </log>                                        |    <extraData> _ </extraData>
      <refund> {REFUND} </refund> // TODO: more detail          |    <mixHash> _ </mixHash>
    </substate>                                                 |    <blockNonce> _ </blockNonce>
    <gasPrice> _ </gasPrice>                                    |    <ommerBlockHeaders> _ </ommerBlockHeaders>
    <origin> ORIGIN_ID </origin>                                |    <blockhash> _ </blockhash>
  </evm>
  <network>
    <activeAccounts> ACCT_ID |-> false _:Map </activeAccounts>
    <accounts>
      <account>
        <acctID> ACCT_ID </acctID>
        <balance> _ </balance>
        <code> #parseByteStack({CODE}) </code>
        <storage> {STORAGE} </storage>
        <nonce> _ </nonce>
      </account>
   // ... // TODO: fix
    </accounts>
    <txOrder> _ </txOrder>
    <txPending> _ </txPending>
    <messages> _ </messages>
  </network>
</ethereum>
```