Automated Invariant Generation for Solidity Smart Contracts

Yi Li
Associate Professor
NTU CCDS

May 27, 2024

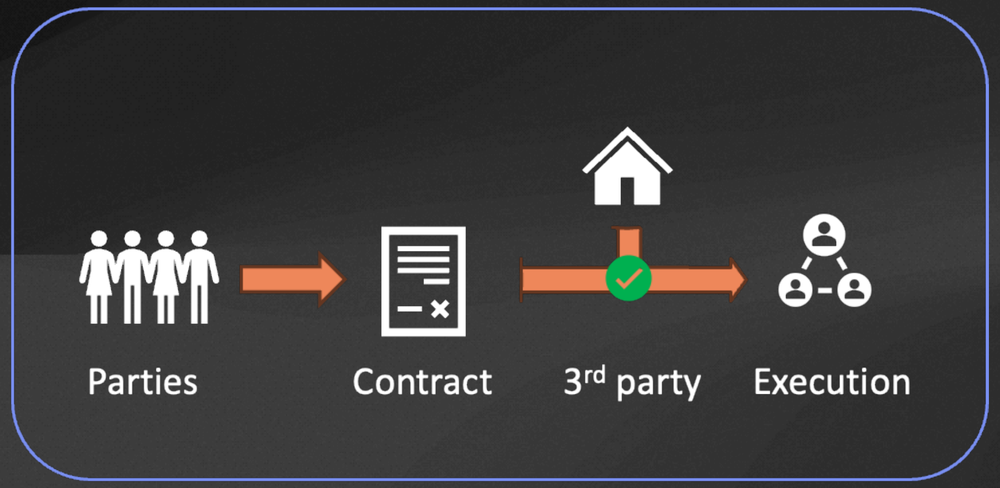
Acknowledgement

Ye Liu

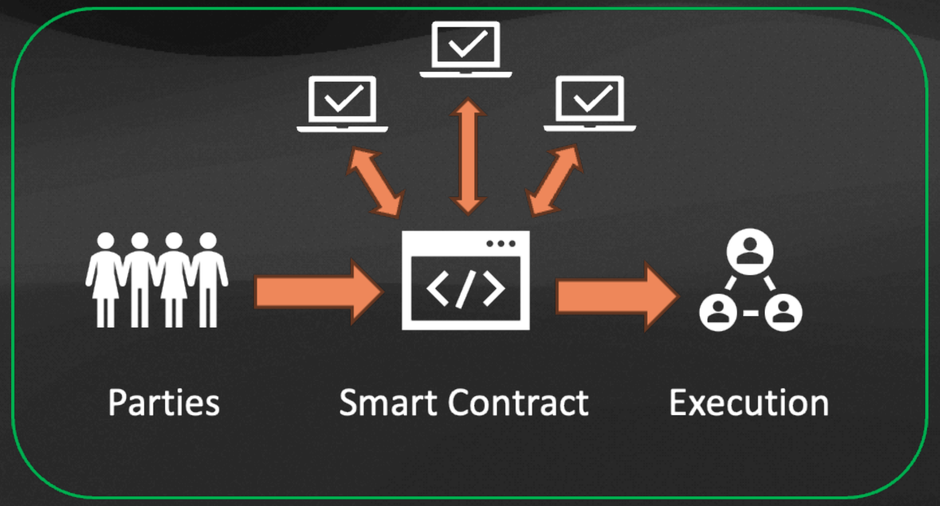
Chengxuan Zhang

Smart Contracts

No need to trust intermediaries, much lower costs



Traditional Contract

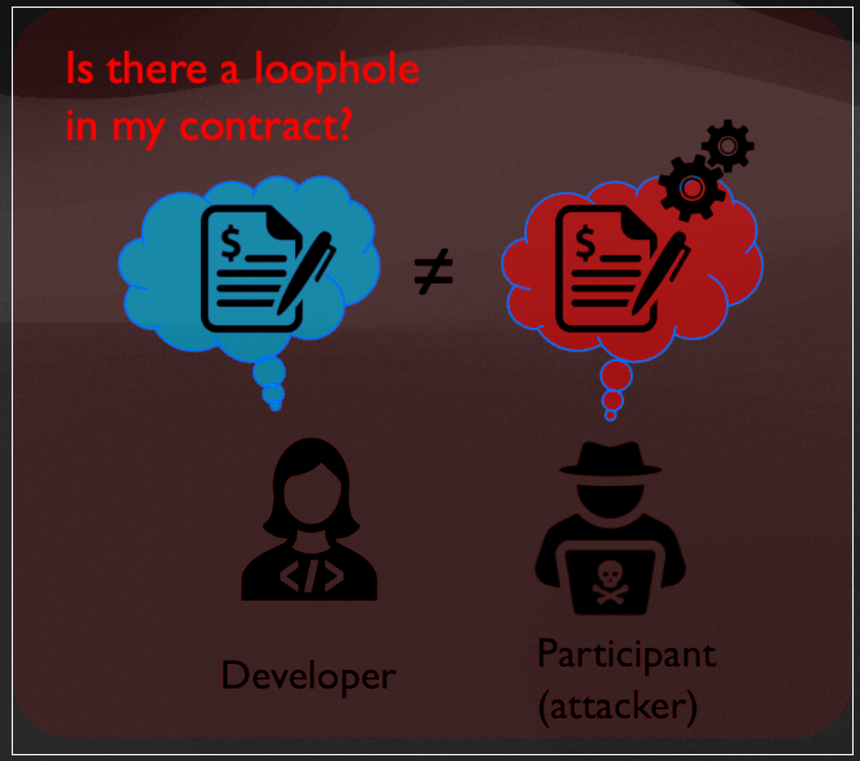


Smart Contract

Security Threats to Smart Contracts

No need to trust intermediaries, but still need to trust code is correct

- What the developer expects (specification) may be inconsistent with how the code was written (implementation)
- This may lead to security vulnerabilities (loopholes in contracts)
- How to check if the code is correct?
  - Many existing tools and techniques
  - But we need contract specifications first!
  - Problem? Nobody writes specifications :-(



A simple ERC20 contract

iToken Duplication Issue ($8M loss)

```
contract iToken … {

    function transfer(address _from, address _to, uint256 _value) public returns (bool res) {
        uint256 _balancesFrom = balances[_from];
        uint256 _balancesTo = balances[_to];

        require(_balancesFrom >= _value);
        uint256 _balancesFromNew = _balancesFrom - _value;
        balances[_from] = _balancesFromNew;

        uint256 _balancesToNew = _balancesTo + _value;
        balances[_to] = _balancesToNew;
    }
}
```

5

10

10

5

15

15!

It goes wrong when “_from == _to”!

https://fullycrypto.com/bzx-suffers-token-duplication-incident

A simple ERC20 contract

Correct implementation annotated with contract specifications (invariants)

```
contract ERC20 {
    // state variables
    uint totalSupply;
    mapping(address => uint) balances;
    mapping(address => mapping(address => uint)) allows;
    //...
    function transferFrom(
        address from,
        address to,
        uint tokens
    ) public returns (bool) {
        if (to == address(0)) {
            return false;
        }
        allows[from][msg.sender] = allows[from][msg.sender].sub(tokens);
        balances[from] = balances[from].sub(tokens);
        balances[to] = balances[to].add(tokens);
        return true;
    }
}
```

ContractInv: SumMap(balances) = totalSupply

Requires: to ≠ 0 ⇒ old(balances[from]) >= tokens

Ensures: to ≠ 0 ∧ from ≠ to ⇒

balance[from] = old(balance[from]) – tokens
∧ balance[to] = old(balance[to]) + tokens

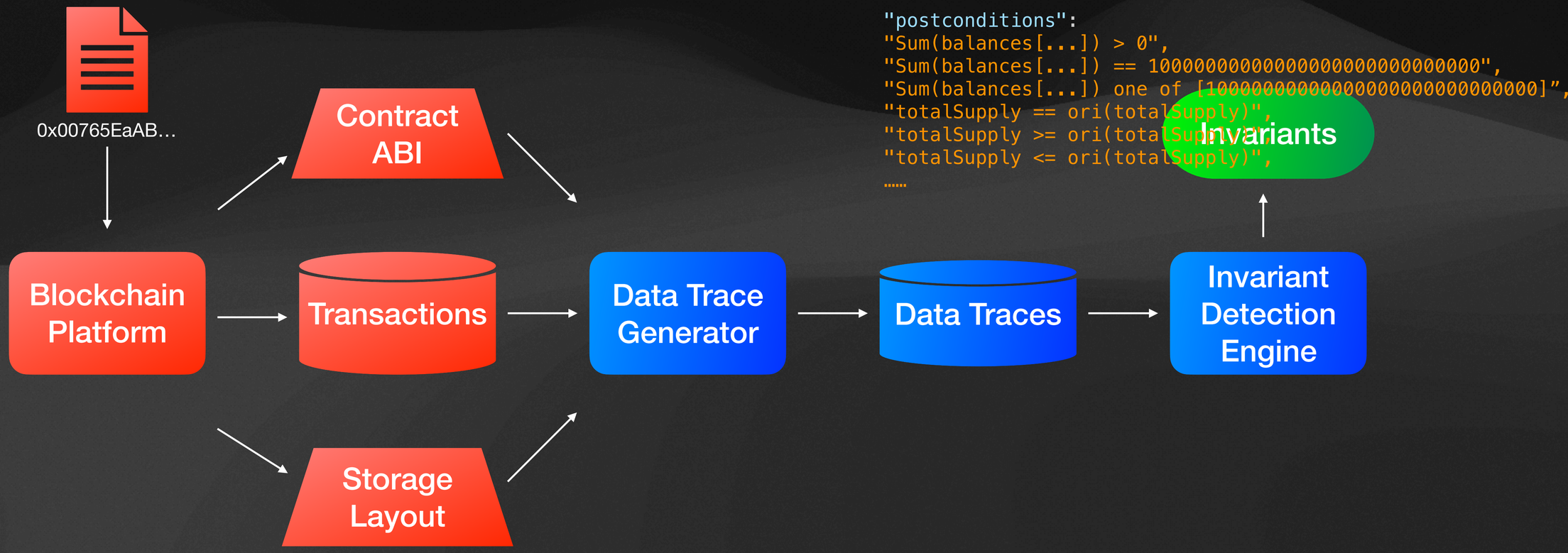
Ensures: to ≠ 0 ∧ from = to ⇒

balance[from] = old(balance[from])
∧ balance[to] = old(balance[to])

InvCon [Liu&Li, 2022]

Inferring likely specs from past executions

```
"preconditions":
"_to != 0",
"msg.sender != 0”,
"_value >= msg.value",
"_value > msg.value",
"_value != msg.value”,
"_to != _from",
……

"postconditions":
"Sum(balances[...]) > 0",
"Sum(balances[...]) == 100000000000000000000000000000",
"Sum(balances[...]) one of [100000000000000000000000000000]”,
"totalSupply == ori(totalSupply)",
"totalSupply >= ori(totalSupply)",
"totalSupply <= ori(totalSupply)",
……
```



See it live: http://52.77.235.110/invcon

Looks good, but …

- Likely invariants may not always hold, for example:
  - Inferred based on limited historical transactions

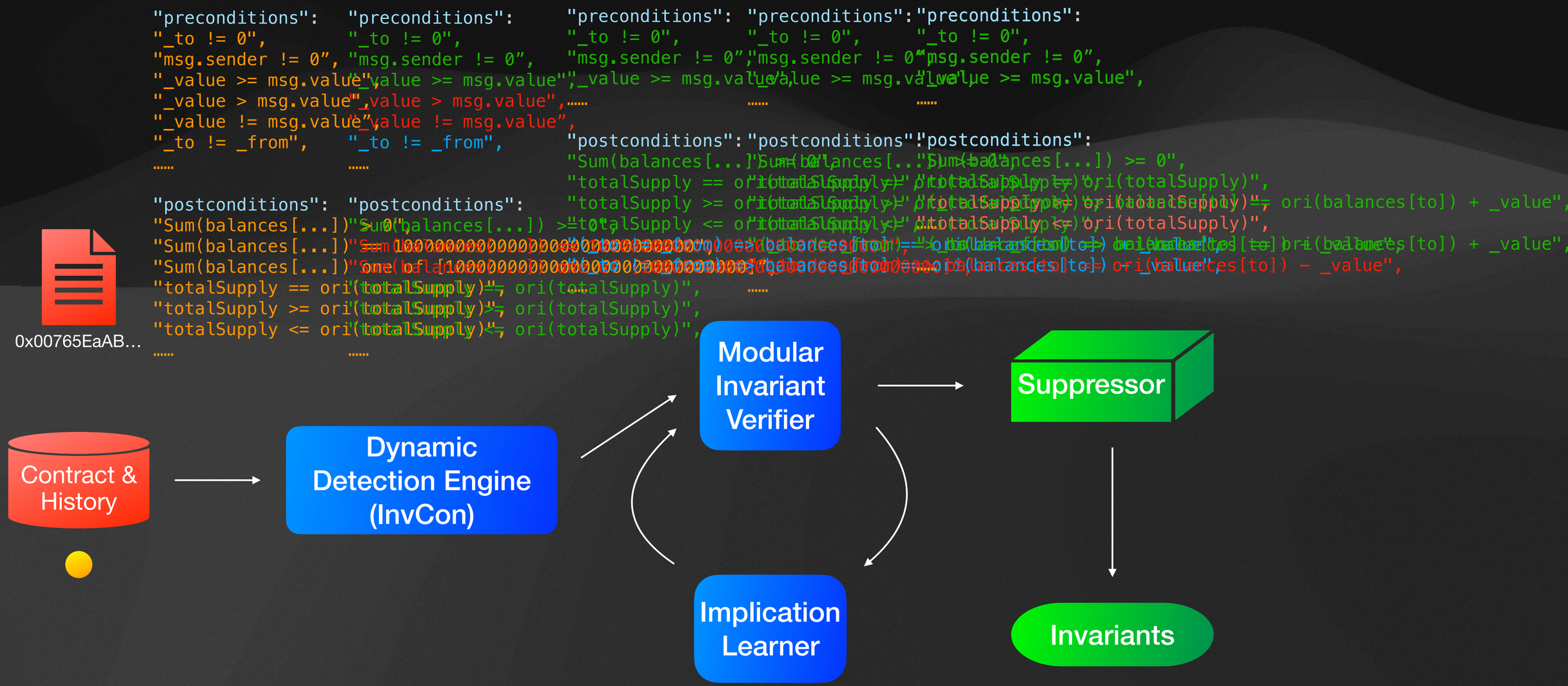
```
Sum(balances[...]) == 100000000000000000000000000000
Sum(balances[...]) one of [100000000000000000000000000000]

_to != _from
```

  - Irrelevant invariants hold by accident

```
msg.value < block.timestamp
```

InvCon+ Overview

Automatically generate statically verified invariants (based on VeriSol)



Evaluation

Effectiveness

How many expected invariants can be successfully generated?

Common ERC20 invariants

Categories	Preconditions	Postconditions
transfer(to, amount)	**[a1]** msg.sender $\neq$ 0 **[a2]** to $\neq$ address(0) **[a3]** amount $\geq$ 0 **[a4]** amount $\leq$ balances[msg.sender] **[a5]** balances[to] + amount $\leq$ MAXVALUE	**[b1]** to $\neq$ msg.sender $\implies$ balances[msg.sender] = old(balances[msg.sender]) - amount **[b2]** to $\neq$ msg.sender $\implies$ balances[to] = old(balances[to]) + amount **[b3]** to = msg.sender $\implies$ balances[to] = old(balances[to]) **[b4]** to = msg.sender $\implies$ balances[msg.sender] = old(balances[msg.sender]) **[b5]** totalSupply = old(totalSupply)
transferFrom (from, to, amount)	**[a6]** from $\neq$ address(0) **[a7]** to $\neq$ address(0) **[a8]** amount $\geq$ 0 **[a9]** amount $\leq$ balances[from] **[a10]** amt $\leq$ allowed[from][msg.sender] **[a11]** balances[to] + amount $\leq$ MAXVALUE	**[b6]** allowed[from][msg.sender] = old(allowed[from][msg.sender]) - amount **[b7]** from $\neq$ to $\implies$ balances[from] = old(balances[from]) - amount **[b8]** from $\neq$ to $\implies$ balances[to] = old(balances[to]) + amount **[b9]** from = to $\implies$ balances[from] = old(balances[from]) **[b10]** allowed[from][msg.sender] = old(allowed[from][msg.sender]) - amount **[b11]** totalSupply = old(totalSupply)
approve(spender, amount)	**[a12]** amount $\geq$ 0 **[a13]** spender $\neq$ address(0)	**[b12]** allowed[msg.sender][spender] = amount **[b13]** totalSupply = old(totalSupply)
increaseAllowance(spender, amount)	**[a14]** spender $\neq$ address(0) **[a15]** amount $\geq$ 0 **[a16]** allowed[msg.sender][spender] + amount $\leq$ MAXVALUE	**[b14]** allowed[msg.sender][spender] = old(allowed[msg.sender][spender]) + amount **[b15]** totalSupply = old(totalSupply)
decreaseAllowance(spender, amount)	**[a17]** spender $\neq$ address(0) **[a18]** amount $\geq$ 0 **[a19]** allowed[msg.sender][spender] $\geq$ amount	**[b16]** allowed[msg.sender][spender] = old(allowed[msg.sender][spender]) - amount **[b17]** totalSupply = old(totalSupply)
mint(account, amount)	**[a20]** account $\neq$ address(0) **[a21]** amount $\geq$ 0 **[a22]** balances[account] + amount $\leq$ MAXVALUE	**[b18]** balances[account] = old(balances[account]) + amount **[b19]** totalSupply = old(totalSupply) + amount
burn(from, amount)	**[a23]** from $\neq$ address(0) **[a24]** amount $\geq$ 0 **[a25]** balances[from] $\geq$ amount	**[b20]** balances[from] = old(balances[from]) - amount **[b21]** totalSupply = old(totalSupply) + amount
pause()	**[a26]** paused = false	**[b22]** paused = true
unpause()	**[a27]** paused = true	**[b23]** paused = false
Contract Invariant	**[c1]** totalSupply = SumMap(balances)	

Effectiveness

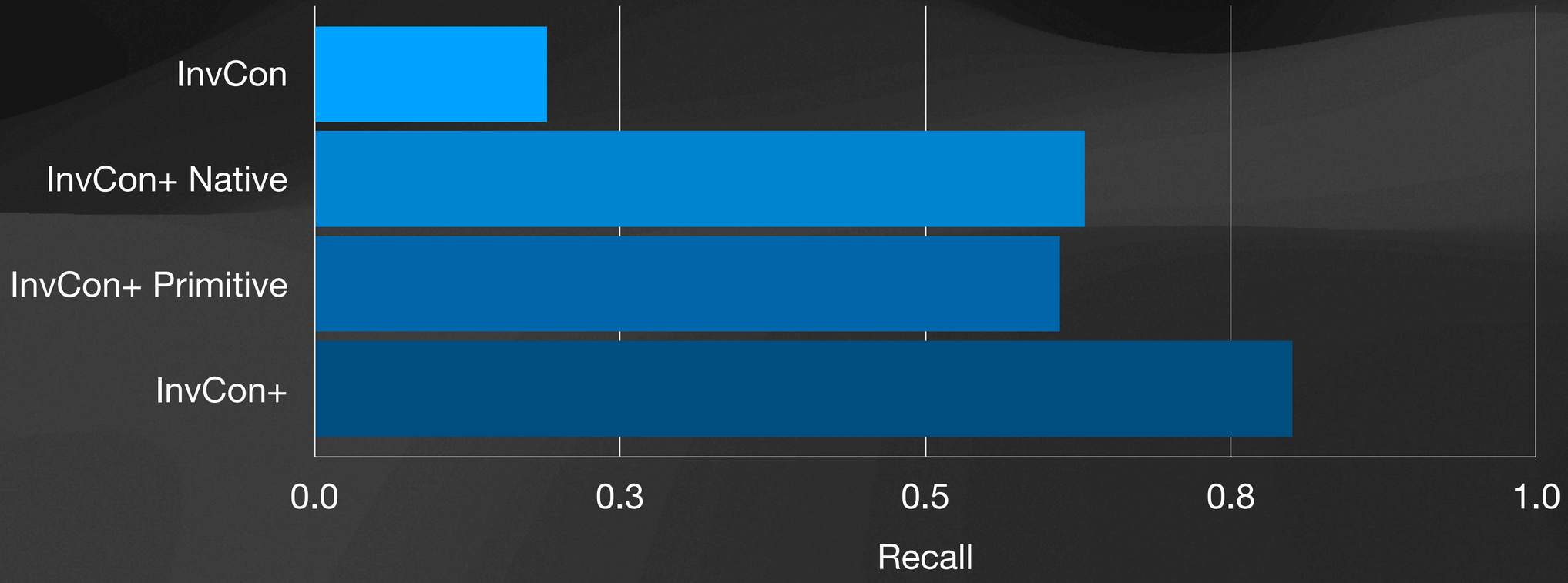
How many expected invariants can be successfully generated?

Categories	Preconditions	Postconditions
(safe)-transferFrom(from, to, tokenId)	**[a28]** from = _tokenOwner[tokenId] **[a29]** from $\neq$ address(0) **[a30]** to $\neq$ address(0) **[a31]** (msg.sender = from $\vee$ msg.sender = _tokenApprovals[tokenId] $\vee$ _operatorApprovals[from][msg.sender] = true)	**[b24]** from $\neq$ to $\implies$ _ownedTokensCount[from] = old(_ownedTokensCount[from]) - 1 **[b25]** from $\neq$ to $\implies$ _ownedTokensCount[to] = old(_ownedTokensCount[to]) + 1 **[b26]** from = to $\implies$ _ownedTokensCount[from] = old(_ownedTokensCount[from]) **[b27]** from = to $\implies$ _ownedTokensCount[to] = old(_ownedTokensCount[to]) **[b28]** _tokenOwner[tokenId] = to **[b29]** _tokenApprovals[tokenId] = address(0)
approve(to, tokenId)	**[a32]** _tokenOwner[tokenId] $\neq$ address(0) **[a33]** (msg.sender = _tokenOwner[tokenId] $\vee$ _operatorApprovals[_tokenOwner[tokenId]][msg.sender] = true)	**[b30]** _tokenApprovals[tokenId] = to
setApproveForAll(operator, _approved)	**[a34]** operator $\neq$ msg.sender	**[b31]** _operatorApprovals[msg.sender][operator] = _approved
Contract Invariant	**[c2]** len(_tokenOwner) = SumMap(_ownerTokenCount)	

Common ERC721 invariants

Effectiveness

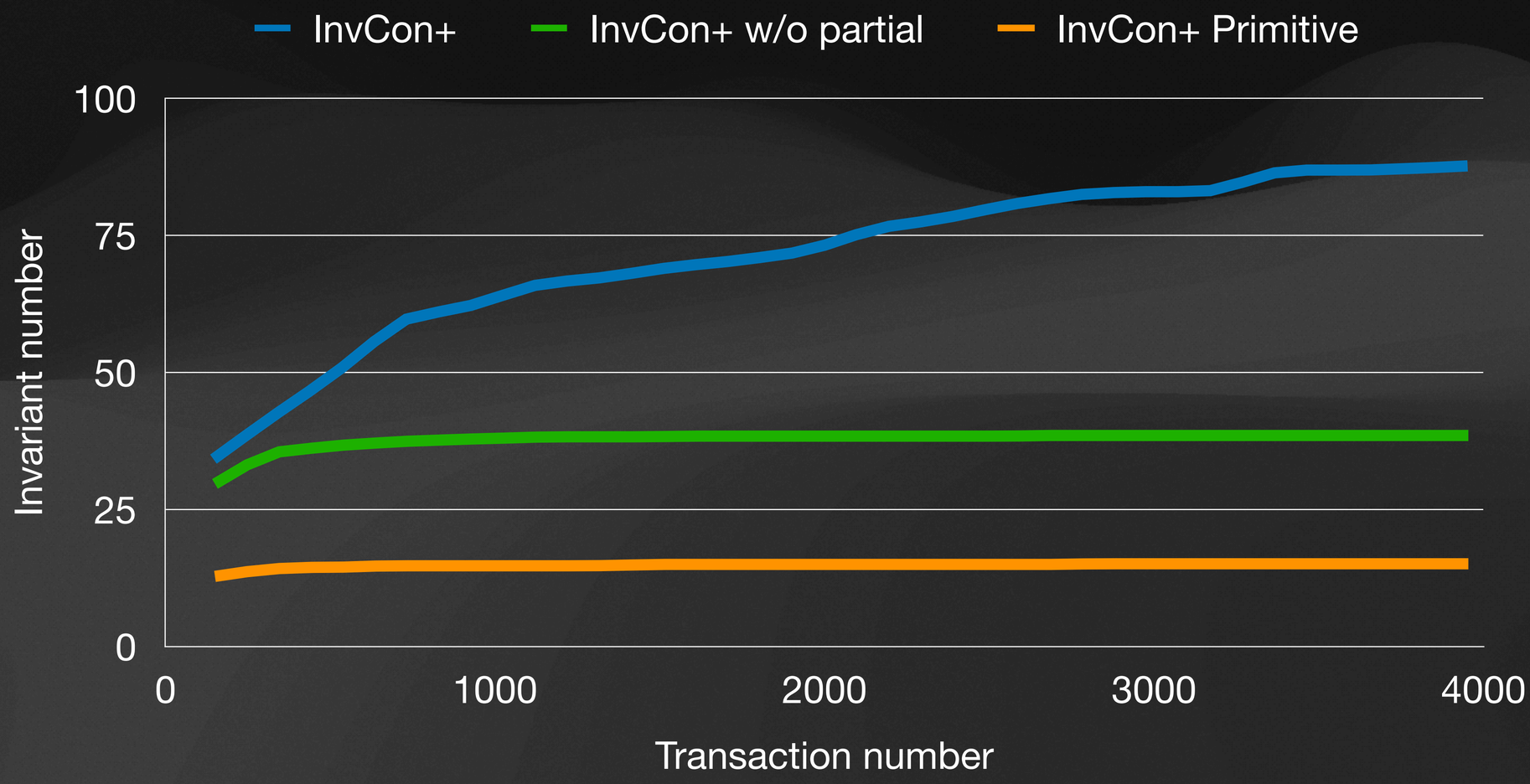
How many expected invariants can be successfully generated?



The comparison result on ERC20 contracts.

Scalability

How does the length of transaction histories used affect the performance of InvCon+?



The averaged number of invariants generated with different number of transactions.

Security Applications

- Dataset:
  - Awesome Buggy ERC20 Tokens
  - Real-world vulnerabilities in ERC20 smart contracts with financial losses
- Results:
  - Invariants detected by InvCon+ useful for preventing most real-world vulnerabilities

ID	Vulnerability Types	Total	Detected
v1	batchTransfer-overflow	13	Yes
v2	totalsupply-overflow	521	Yes
v3	verify-invalid-by-overflow	2	Yes
v4	owner-control-sell-price-for-overflow	1	Yes
v5	owner-overweight-token-by-overflow	9	Yes
v6	owner-decrease-balance-by-mint-by-overflow	487	Yes
v7	excess-allocation-by-overflow	1	Yes
v8	excess-mint-token-by-overflow	9	Yes
v9	excess-buy-token-by-overflow	4	Yes
v10	verify-reverse-in-transferFrom	79	Yes
v11	pauseTransfer-anyone	1	No
v12	transferProxy-keccak256	10	Yes
v13	approveProxy-keccak256	10	Yes
v14	constructor-case-insensitive	4	N/A
v15	custom-fallback-bypass-ds-auth	1	N/A
v16	custom-call-abuse	144	N/A
v17	setowner-anyone	3	Yes
v18	allowAnyone	4	Yes
v19	approve-with-balance-verify	18	Yes
v20	check-effect-inconsistency	1	Yes
v21	constructor-mistyping	4	N/A
v22	fake-burn	2	Yes
v23	getToken-anyone	3	N/A
v24	constructor-naming-error	1	N/A

Detecting security vulnerabilities

CVE-2018-10299 from BeautyChain (BEC)

batchTransfer function in the BEC contract

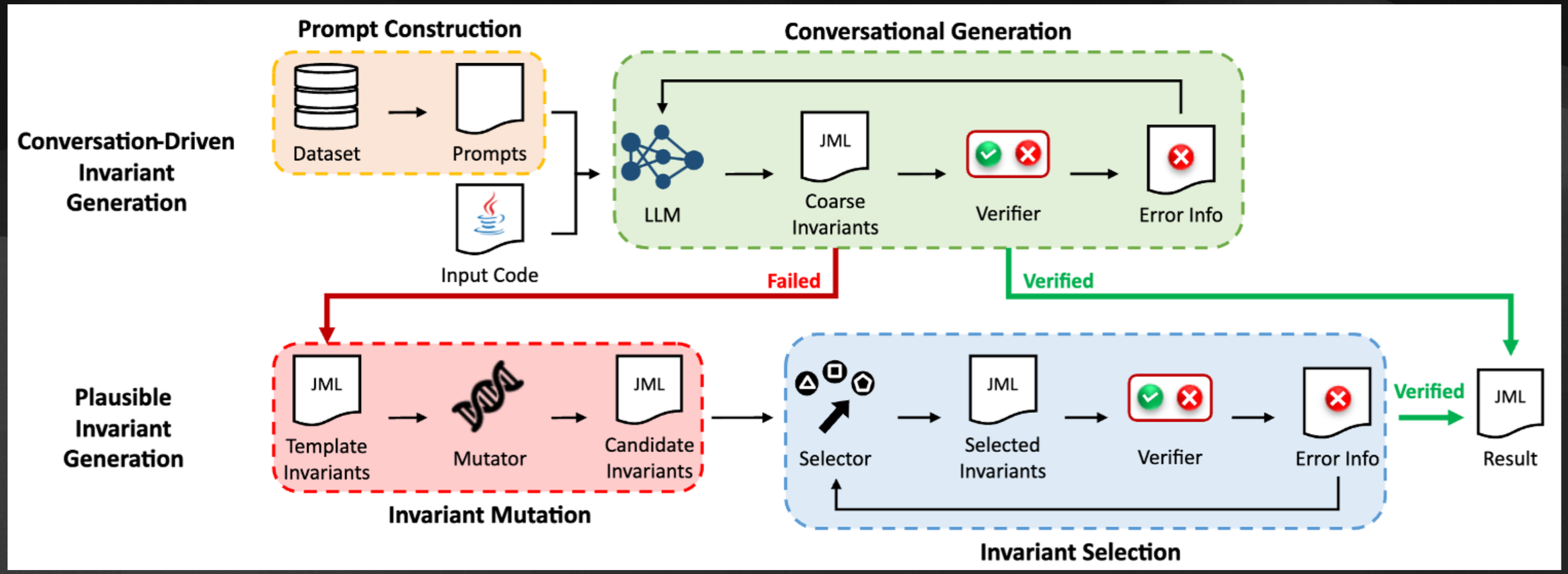
```
function batchTransfer(
    address[] _receivers,
    uint256 _value
) public whenNotPaused returns (bool) {
    uint cnt = _receivers.length;
    uint256 amount = uint256(cnt) * _value;
    require(cnt > 0 && cnt <= 20);
    require(_value > 0 && balances[msg.sender] >= amount);

    [msg.sender] = balances[msg.sender].sub(amount);
    for (uint i = 0; i < cnt; i++) {
        balances[_receivers[i]] = balances[_receivers[i]].add(_value);
        Transfer(msg.sender, _receivers[i], _value);
    }
    return true;
}
```

Violates “totalSupply == SumMapping(balances)” !!!

A bit more on Generating Specs

Automated Generation of Formal Specifications using LLMs

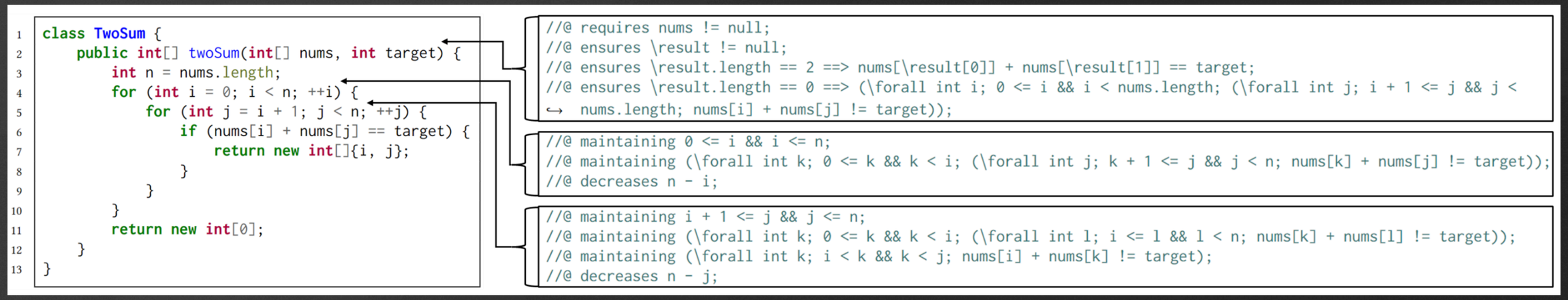
***SpecGen*: Ma et al. https://arxiv.org/abs/2401.08807**



Enhanced Spec Generation Capability

SpecGen: Ma et al. https://arxiv.org/abs/2401.08807

- Existing invariant generation approaches struggle on non-trivial cases
  - Invariant templates based on heuristic are often limited
  - E.g., Daikon and Houdini only generates “nums != null”, “\result[i] >= 0”, etc.



```java
class TwoSum {
    public int[] twoSum(int[] nums, int target) {
        int n = nums.length;
        for (int i = 0; i < n; ++i) {
            for (int j = i + 1; j < n; ++j) {
                if (nums[i] + nums[j] == target) {
                    return new int[]{i, j};
                }
            }
        }
        return new int[0];
    }
}
```

```
//@ requires nums != null;
//@ ensures \result != null;
//@ ensures \result.length == 2 ==> nums[\result[0]] + nums[\result[1]] == target;
//@ ensures \result.length == 0 ==> (\forall int i; 0 <= i && i < nums.length; (\forall int j; i + 1 <= j && j <
↪  nums.length; nums[i] + nums[j] != target));
```

```
//@ maintaining 0 <= i && i <= n;
//@ maintaining (\forall int k; 0 <= k && k < i; (\forall int j; k + 1 <= j && j < n; nums[k] + nums[j] != target));
//@ decreases n - i;
```

```
//@ maintaining i + 1 <= j && j <= n;
//@ maintaining (\forall int k; 0 <= k && k < i; (\forall int l; i <= l && l < n; nums[k] + nums[l] != target));
//@ maintaining (\forall int k; i < k && k < j; nums[i] + nums[k] != target);
//@ decreases n - j;
```

Evaluation

***SpecGen*: Ma et al. https://arxiv.org/abs/2401.08807**

TABLE II: Number of programs that successfully pass the verifier and average success probability.

Approach		SV-COMP (265)		*SpecGenBench*										Overall (385)	
				Sequential (26)		Branched (23)		Single-path Loop (24)		Multi-path Loop (26)		Nested Loop (21)			
		Num.	Prob.	Num.	Prob.	Num.	Prob.	Num.	Prob.	Num.	Prob.	Num.	Prob.	Num.	Prob.
Daikon		51	-	10	-	10	-	0	-	1	-	0	-	72	-
Houdini		56	-	14	-	11	-	10	-	4	-	3	-	98	-
Purely LLM-based	0-shot	81	18.28%	23	74.19%	17	58.55%	5	7.08%	7	18.13%	2	3.33%	135	22.93%
	2-shot	83	18.79%	20	61.06%	17	53.91%	8	19.29%	13	26.58%	4	3.81%	145	23.48%
	4-shot	94	19.40%	23	73.85%	20	57.33%	10	23.40%	12	24.95%	5	6.24%	164	25.25%
	Conversational	146	30.95%	23	82.49%	20	75.43%	12	27.02%	13	35.38%	4	9.20%	218	35.95%
SpecGen		**179**	**40.41%**	**24**	**92.31%**	**20**	**79.57%**	**23**	**73.75%**	**20**	**60.38%**	**13**	**36.55%**	**279**	**59.97%**

Questions?

Usages of InvCon+

- Infer specs for deployed contracts
- Derive specs interactively during internal testing

Repository: https://github.com/ntu-SRSLab/InvCon