ILLINOIS INSTITUTE OF TECHNOLOGY

Towards Automated Safety Vetting of Smart Contracts in Decentralized Applications

Yue Duan
Illinois Institute of Technology

ILLINOIS INSTITUTE OF TECHNOLOGY

Smart Contract

- Physical contracts
- Trusted 3rd party
- Slow speed
- High cost

ILLINOIS INSTITUTE OF TECHNOLOGY

Smart Contract

- Spontaneous
- Fully automated
- High speed
- Secure

ILLINOIS INSTITUTE OF TECHNOLOGY

Security and Safety Issues

- Everything comes with a price!
- Syntactic-level security issues:
  - reentrancy vulnerability
  - transaction-ordering dependence
  - timestamp dependence
  - mishandled exceptions
  - etc.
- Existing detection techniques:
  - Static analysis
  - Symbolic Execution
  - Fuzzing

ILLINOIS INSTITUTE OF TECHNOLOGY

Security and Safety Issues

- Semantic-level Safety issues:

Owner can cancel at ANYTIME. Safety risk!

```
1  function bidOnAuction(uint _id) public payable {
2      uint256 ethAmountSent = msg.value;
3
4      // owner cannot bid on his/her own merchandise
5      Auction memory myAuction = auctions[_id];
6      if(myAuction.owner == msg.sender) revert();
7
8      // check whether auction has expired
9      if(block.timestamp > myAuction.deadline) revert();
10
11     // check whether previous bids exist
12     uint bidsLength = accepted[_id].length;
13     uint256 tempAmount = myAuction.startPrice;
14     Bid memory lastBid;
15     if(bidsLength > 0) {
16         lastBid = accepted[_id][bidsLength-1];
17         tempAmount = lastBid.amount;
18     }
19
20     // check if bid price is greater than the current
        highest
21     if(ethAmountSent < tempAmount) revert();
22
23     // add the new bid to auction state
24     Bid memory newBid;
25     newBid.from = msg.sender;
26     newBid.amount = ethAmountSent;
27     accepted[_id].push(newBid);
28     emit BidSuccess(msg.sender, _id);
29 }
```

```
31 function cancelAuction(uint _id) public isOwner(_id) {
32     Auction memory myAuction = auctions[_id];
33     uint bidsLen = accepted[_id].length;
34
35     // refund the last bid, if prior bids exist
36     if(bidsLen > 0) {
37         Bid memory lastBid = accepted[_id][bidsLen - 1];
38         if(!lastBid.from.send(lastBid.amount)) revert();
39     }
40     auctions[_id].active = false;
41     emit AuctionCanceled(msg.sender, _id);
42 }
```

ILLINOIS INSTITUTE OF TECHNOLOGY

Security and Safety Issues

- Semantic-level Safety issues:
  - more challenging
  - need semantic information about c[illegible]es

```
31 function cancelAuction(uint _id) public isOwner(_id) {
32     Auction memory myAuction = auctions[_id];
33     uint bidsLen = accepted[_id].le[illegible]h;
34
35     // refund the last bid, if prior bids exist
36     if(bidsLen > 0) {
37         Bid memory lastBid = accepted[_id][bidsLen - 1];
38         if(!lastBid.from.send(lastBid.amount)) r[illegible]
39     }
40     auctions[_id].active [illegible]alse;
41     emit AuctionCanceled(msg.sender, _id);
42 }
```

terminate a bidding process

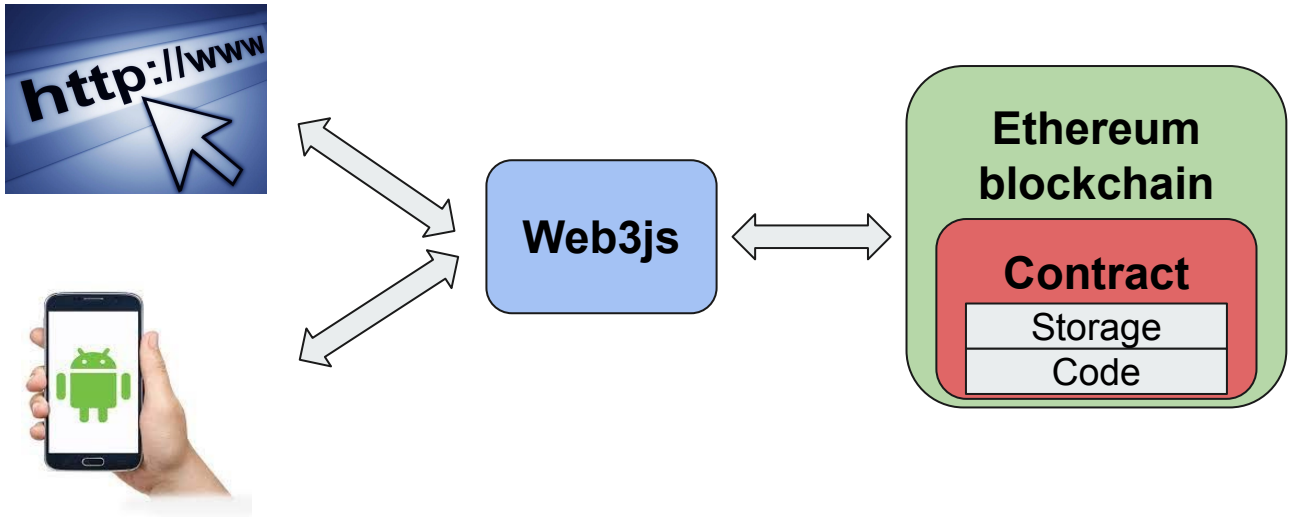
prior accepted bids

auction state

ILLINOIS INSTITUTE OF TECHNOLOGY

Security and Safety Issues

GUI Front-end middleware backend



ILLINOIS INSTITUTE OF TECHNOLOGY

Security and Safety Issues

- Existing detection techniqu
  - Formal verification,
    - Zeus [NDSS'18]
    - VerX [Oakland'20]
    - SmartPulse [Oakland'21]
- Limitation:
  - understanding of source code
  - manually crafted safety specifications for every contract
  - do not consider DApp scenario

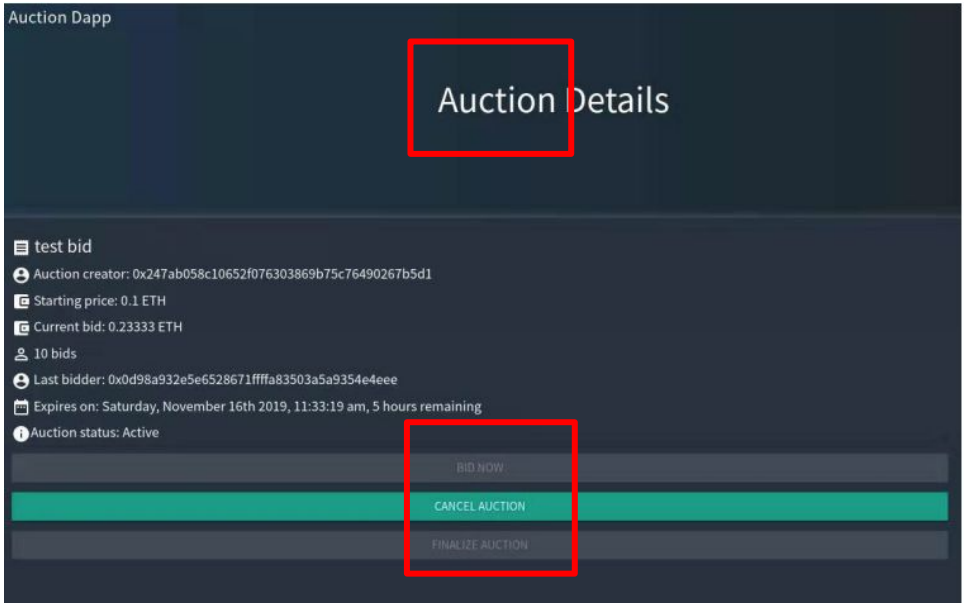
- unreliable
- terious and error-prone
- maybe unavailable

ILLINOIS INSTITUTE OF TECHNOLOGY

Our Insights

- Front-end UI contains semantic information

  => may help with code understanding



ILLINOIS INSTITUTE OF TECHNOLOGY

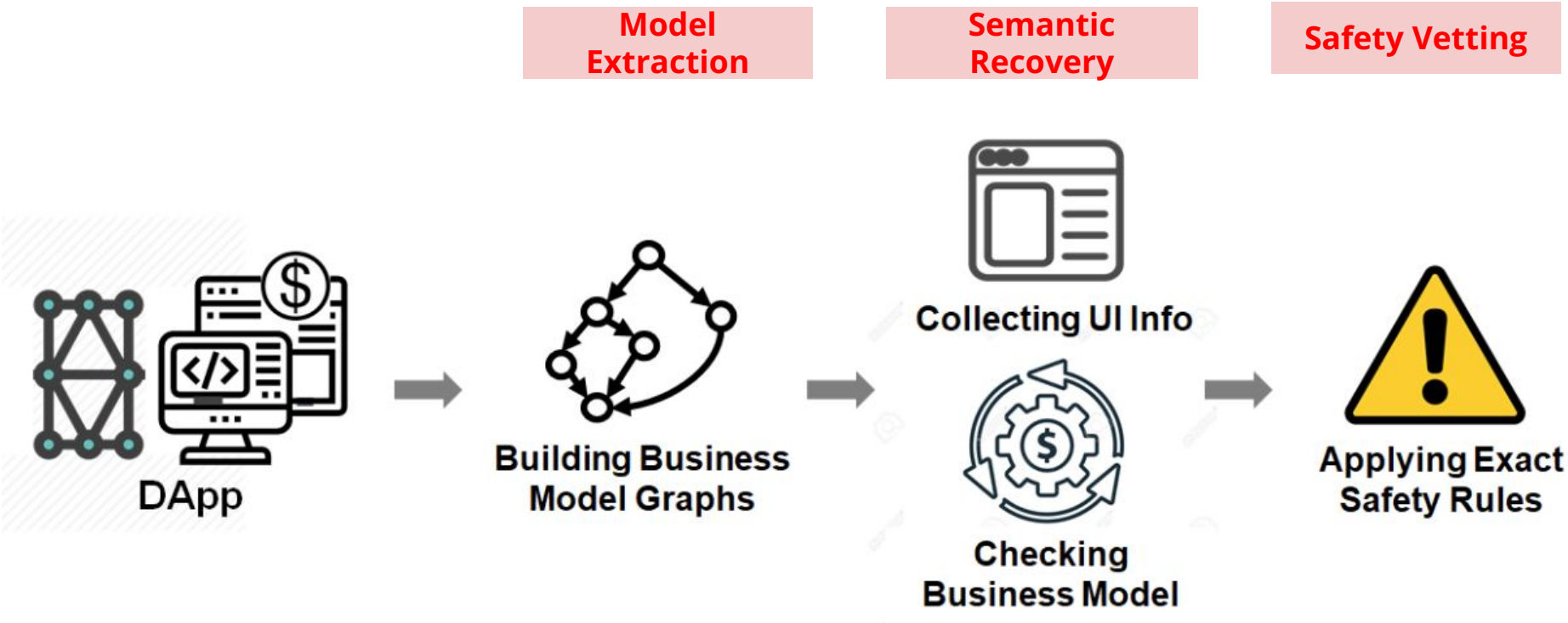
Our Insights

- Limited categories of business models
  - auction
  - voting
  - trading
  - lottery
  - wallet
  - crowdsale
  - etc.

  => may help automate safety spec crafting

ILLINOIS INSTITUTE OF TECHNOLOGY

VetSC Overview



ILLINOIS INSTITUTE OF TECHNOLOGY

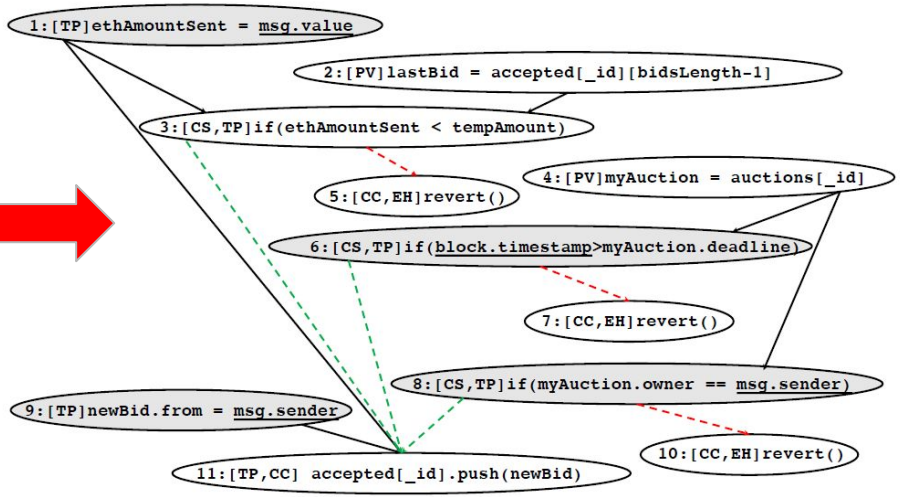
VetSC - Model Extraction

- Generate business models based on a given smart contract
- Our solution
  - Business Model Graph (BMG)
  - A directed graph G = (V, E, **α**, ***β***) over statements **Σ** and relations R
    - V: statements in **Σ**
    - E: causal dependencies between statements
    - **α**: V -> **Σ**
      - labeling function that associates nodes with labels
    - ***β***: V -> R
      - labeling function that associates edges with labels

ILLINOIS INSTITUTE OF TECHNOLOGY

VetSC - Model Extraction

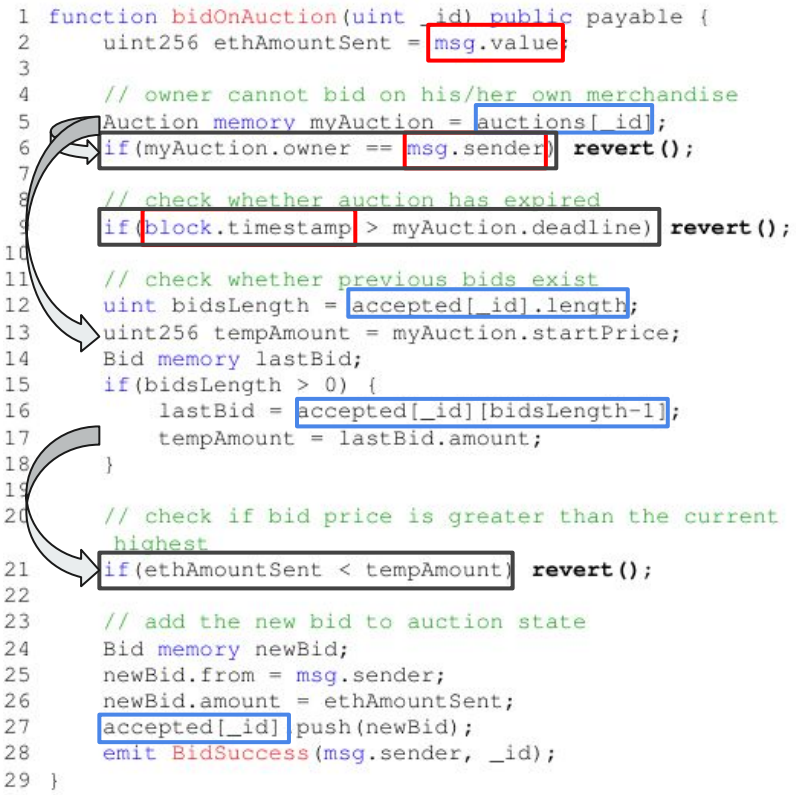
```
1  function bidOnAuction(uint _id) public payable {
2      uint256 ethAmountSent = msg.value;
3
4      // owner cannot bid on his/her own merchandise
5      Auction memory myAuction = auctions[_id];
6      if(myAuction.owner == msg.sender) revert();
7
8      // check whether auction has expired
9      if(block.timestamp > myAuction.deadline) revert();
10
11     // check whether previous bids exist
12     uint bidsLength = accepted[_id].length;
13     uint256 tempAmount = myAuction.startPrice;
14     Bid memory lastBid;
15     if(bidsLength > 0) {
16         lastBid = accepted[_id][bidsLength-1];
17         tempAmount = lastBid.amount;
18     }
19
20     // check if bid price is greater than the current
        highest
21     if(ethAmountSent < tempAmount) revert();
22
23     // add the new bid to auction state
24     Bid memory newBid;
25     newBid.from = msg.sender;
26     newBid.amount = ethAmountSent;
27     accepted[_id].push(newBid);
28     emit BidSuccess(msg.sender, _id);
29 }
```



ILLINOIS INSTITUTE OF TECHNOLOGY

VetSC - Model Extraction

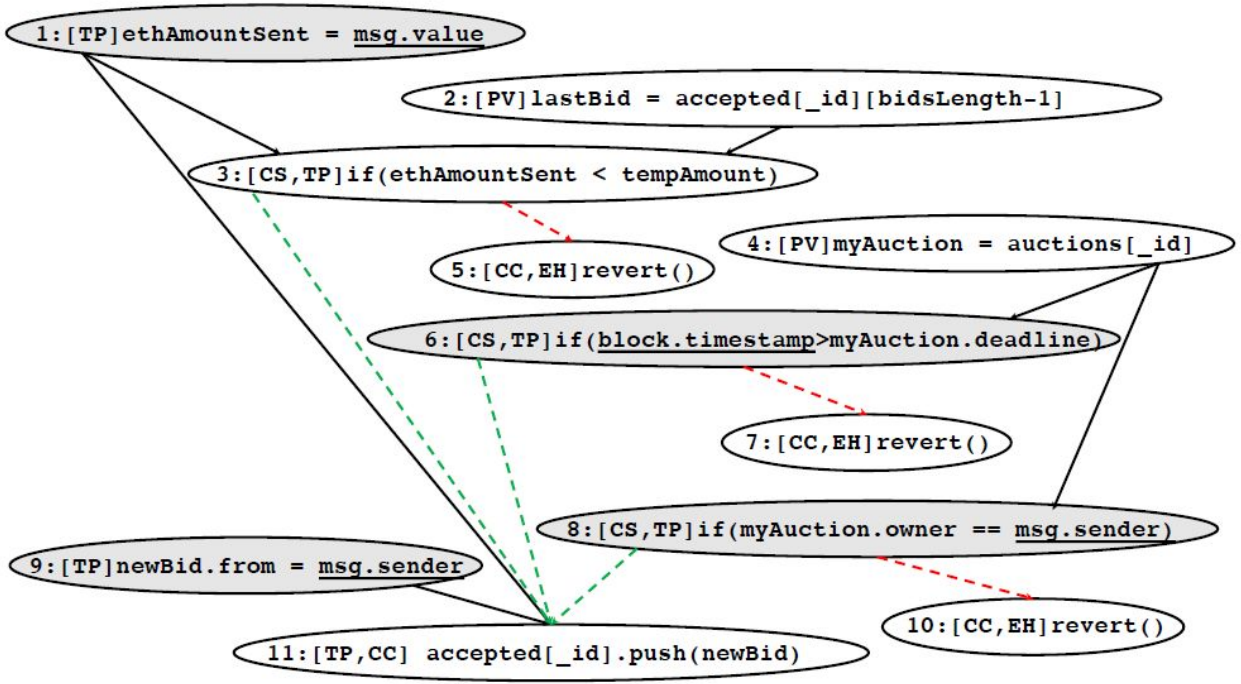
- Five key factors that define smart contract business semantics
  - transaction property
  - global variables
  - dataflow
  - condition check
  - cryptocurrency/token transfer
- Memory Aliasing
  - two global memory regions
    - storage
    - memory



```
1  function bidOnAuction(uint _id) public payable {
2      uint256 ethAmountSent = msg.value;
3
4      // owner cannot bid on his/her own merchandise
5      Auction memory myAuction = auctions[_id];
6      if(myAuction.owner == msg.sender) revert();
7
8      // check whether auction has expired
9      if(block.timestamp > myAuction.deadline) revert();
10
11     // check whether previous bids exist
12     uint bidsLength = accepted[_id].length;
13     uint256 tempAmount = myAuction.startPrice;
14     Bid memory lastBid;
15     if(bidsLength > 0) {
16         lastBid = accepted[_id][bidsLength-1];
17         tempAmount = lastBid.amount;
18     }
19
20     // check if bid price is greater than the current
       highest
21     if(ethAmountSent < tempAmount) revert();
22
23     // add the new bid to auction state
24     Bid memory newBid;
25     newBid.from = msg.sender;
26     newBid.amount = ethAmountSent;
27     accepted[_id].push(newBid);
28     emit BidSuccess(msg.sender, _id);
29 }
```

ILLINOIS INSTITUTE OF TECHNOLOGY

VetSC - Model Extraction



ILLINOIS INSTITUTE OF TECHNOLOGY

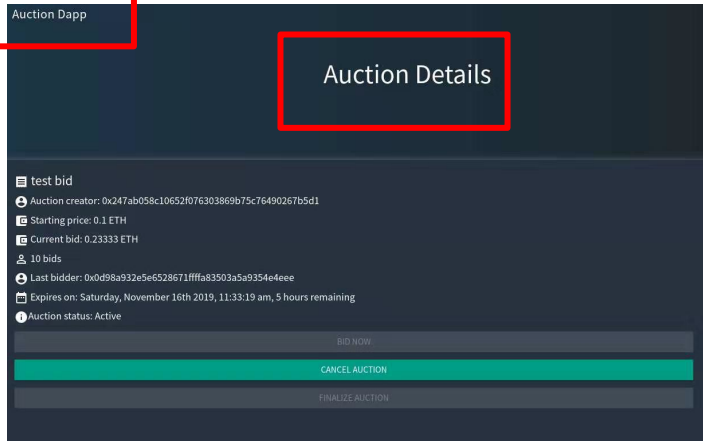
VetSC - Semantic Recovery

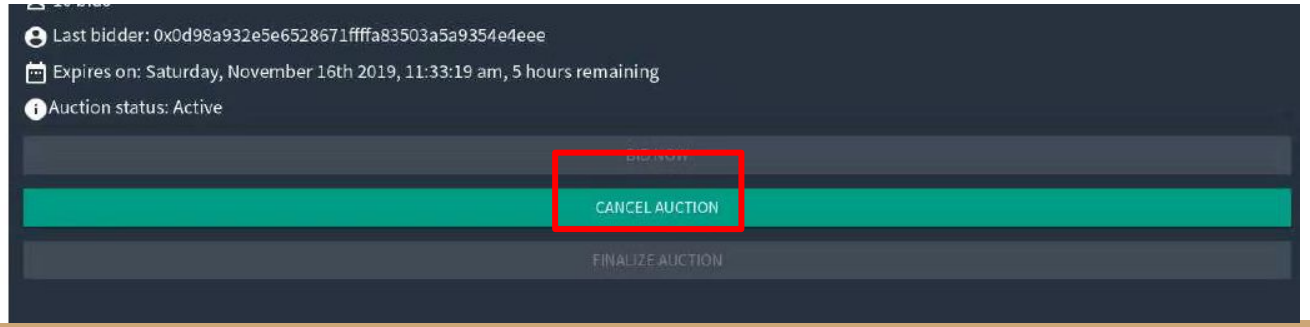
- Three levels of semantic information
  - smart contract level
    - e.g., this is a Auction DApp
  - function level
    - e.g., **cancelAuction**() is to cancel the auction.
  - variable level
    - e.g., **auction[_id].active** is the auction state.

ILLINOIS INSTITUTE OF TECHNOLOGY

VetSC - Semantic Recovery

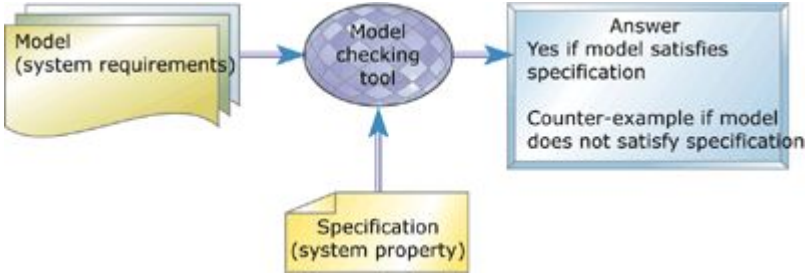
- Smart contract level
  - e.g., this is a Auction DApp
  - HTML parsing ⇒ NLP technique ⇒ semantics
- Function level
  - e.g., **cancelAuction**() is to cancel the auction.
  - GUI text ⇒ NLP technique ⇒ semantics





VetSC - Semantic Recovery

- Variable level
  - formalize it as a **model checking problem**
  - check extracted contract models against essential logic specifications



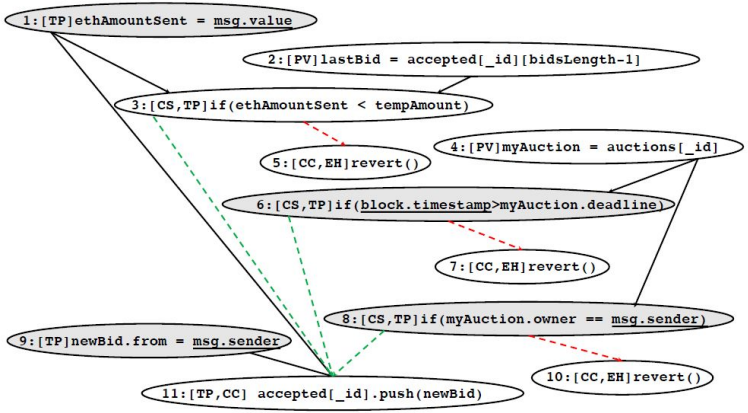
VetSC - Semantic Recovery

- Goal: find a unique mapping μ
  - $\mu = P_{\phi E} \rightarrow GV$ such that $M_{\text{BMG}} \models \mu/\phi_E$,where
    - $P_{\phi E}$ is a set containing all the variables p used in propositions in essential specifications ϕ_E;
    - GV is a set containing all the global variables in BMG;
    - M_{BMG} is the transition system that directly presents BMG;
    - $M_{\text{BMG}} \models \mu/\phi_E$ means that the specification ϕ_E holds over the transition system M_{BMG} under the unique mapping μ, when the variables P_ϕ are mapped to global variables GV .

ILLINOIS INSTITUTE OF TECHNOLOGY

VetSC - Semantic Recovery

Function	Spec Type	Formal Spec
Bidding	Essential#1	$\Box(current\_bid > highest\_bid \rightarrow \Diamond(highest\_bid := current\_bid \wedge highest\_bidder := current\_bidder))$
	Safety#1	$\Box(current\_time > deadline \rightarrow \Diamond(execution\_state := revert))$
	Safety#2	$\Box(auction.active == false \rightarrow \Diamond(execution\_state := revert))$
	Safety#3	$\Box current\_bidder == auction.owner \rightarrow \Diamond(execution\_state := revert))$
Cancel	Essential#1	$auction.active := false$
	Safety#1	$\Box(requester != auction.owner \rightarrow \Diamond(execution\_state := revert))$
	Safety#2	$\Box(highest\_bidder != null \rightarrow \Diamond(execution\_state := revert))$



NuSMV Model

ILLINOIS INSTITUTE OF TECHNOLOGY

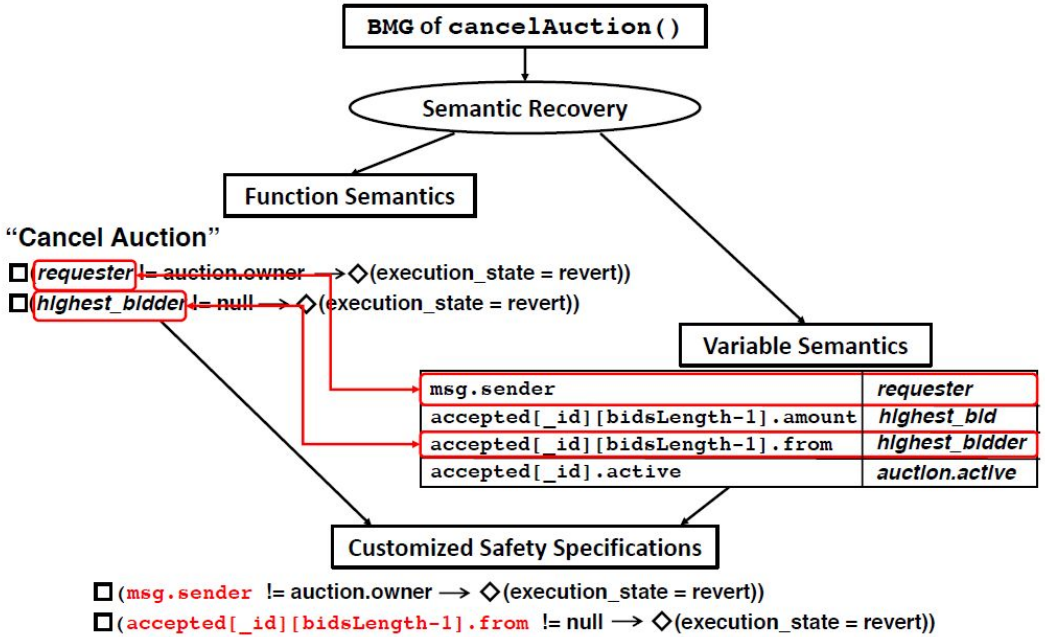
VetSC - Semantic Recovery

Function/Domain	Smart Contract Variable	Spec Concept
`bidOnAction`	`msg.value`	*current_bid*
`bidOnAction`	`msg.sender`	*current_bidder*
`bidOnAction`	`newBid.amount`	*highest_bid*
`bidOnAction`	`newBid.from`	*highest_bidder*
`cancelAction`	`msg.sender`	*requester*
contract-wide	`accepted[_id][bidsLength-1].amount`	*highest_bid*
contract-wide	`accepted[_id][bidsLength-1].from`	*highest_bidder*
contract-wide	`accepted[_id].active`	*auction.active*

ILLINOIS INSTITUTE OF TECHNOLOGY

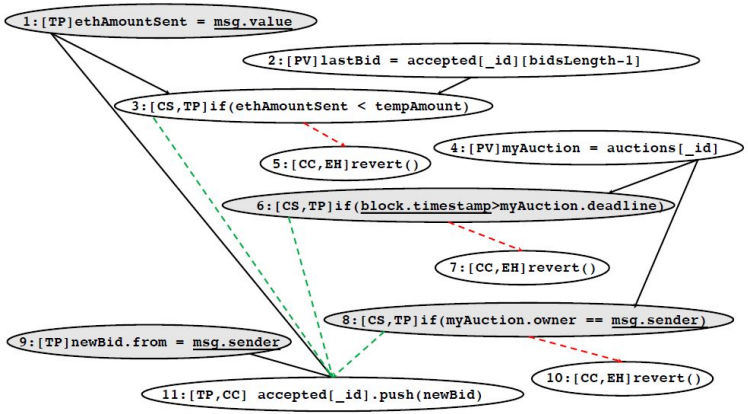
VetSC - Semantic Recovery

Cancel	Essential#1	$auction.active := false$
	Safety#1	$\Box(requester \mathrel{!}= auction.owner \rightarrow \Diamond(execution\_state := revert))$
	Safety#2	$\Box(highest\_bidder \mathrel{!}= null \rightarrow \Diamond(execution\_state := revert))$



ILLINOIS INSTITUTE OF TECHNOLOGY

VetSC - Safety Vetting



Check against

Customized Safety Specifications

□(`msg.sender` != auction.owner ⟶ ◇(execution_state = revert))

□(`accepted[_id][bidsLength-1].from` != null ⟶ ◇(execution_state = revert))

ILLINOIS INSTITUTE OF TECHNOLOGY

Evaluation

- Dataset
  - 34 DApps
  - 494 Solidity functions
- Baseline technique
    - VerX [Oakland'20]

ILLINOIS INSTITUTE OF TECHNOLOGY

Evaluation

#	Name	Unsafe Func Name	Code Logic	Major Widget Text/Context	UI == Logic?	Safety Issue in Smart Contracts	Violated Policy	Source Analyzability
1	cryptoatoms.org	-	-	-	Yes	-	-	Yes
2	proofoflove.digital	-	-	-	Yes	-	-	Yes
3	snailking	-	-	-	Yes	-	-	Yes
4	cryptominingwar	-	-	-	Yes	-	-	Yes
5	market.start.solar	-	-	-	Yes	-	-	No (Missing Source)
6	etheroll	-	-	-	Yes	-	-	No (Inlined Bytecode)
7	cryptokitties	`bid()`	Auction-Bid	*"buy"*	Ambiguity	N/A	N/A	Yes
8	hyperdragons	-	-	-	Yes	-	-	No (Missing Source)
9	dice2.win	-	-	-	Yes	-	-	No (Inlined Bytecode)
10	all-for-one.club	`drawNow()`	Lottery-Draw	*"Draw"*	Yes	Drawing for an expired lottery	Lottery-Draw-S2	No (Inlined Bytecode)
		`play()`	Lottery-Buy	*"pay 1 ETH"*	Yes	Buying an expired ticket	Lottery-Buy-S1	No (Inlined Bytecode)
11	openberry-ac	`placeBid()`	Auction-Bid	*"place BID"*	Yes	Bidding for an expired auction	Auction-Bid-S1	Yes
		`finalizeAuction()`	Auction-Close	*"handle Finalize"*	Yes	Closing a non-expired auction	Auction-Close-S1	Yes
						Closing an active auction	Auction-Close-S2	
12	create-react-dapp	`voteForCandidate()`	Voting-Vote	*"vote Rama/Nick/Jose"*	Yes	Voting for an expired election	Voting-Vote-S1	Yes
						Double voting	Voting-Vote-S2	
13	ethereum-voting	`vote()`	Voting-Vote	*"Vote"*	Yes	Voting for an expired election	Voting-Vote-S1	Yes
14	ethereum-wallet	-	-	-	Yes	-	-	Yes
15	heiswap.exchange	-	-	-	Yes	-	-	No (Inlined Bytecode)
16	Lottery-DApp	`makeGuess()`	Lottery-Buy	*"Buy", "Lottery"*	Yes	Buying an expired ticket	Lottery-Buy-S1	Yes
		`closeGame()`	Lottery-Draw	*"Close Game", "Lottery"*	Yes	Drawing for an expired lottery	Lottery-Draw-S2	Yes
17	mastering-e-a-d	`cancelAuction()`	Auction-Cancel	*"CANCEL AUCTION"*	Yes	Seller cancel after bidding starts	Auction-Cancel-S2	Yes
18	multisender.app	-	-	-	Yes	-	-	Yes
19	note_dapp	-	-	-	Yes	-	-	Yes
20	metacoin	-	-	-	Yes	-	-	Yes
21	simple-vote	`vote()`	N/A	*"Start a vote"*	No Impl.	N/A	N/A	Yes
22	truffle-voting	`vote()`	Voting-Vote	*"Approve/Against/Abstain"*	Yes	Voting for an expired election	Voting-Vote-S1	Yes
23	Gnosis Safe	-	-	-	Yes	-	-	Yes
24	vote-dapp	-	-	-	Yes	-	-	Yes
25	EVotingDApp	-	-	-	Yes	-	-	Yes
26	Election	`vote()`	Voting-Vote	*"Vote"*	Yes	Voting for an expired election	Voting-Vote-S1	Yes
27	Election-DAPP	`vote()`	Voting-Vote	*"Approve/Against/Abstence"*	Yes	Voting for an expired election	Voting-Vote-S1	Yes
28	Vote	`vote()`	Voting-Vote	*"Submit"*	Yes	Voting for an expired election	Voting-Vote-S1	Yes
29	VotingDapp	`vote()`	Voting-Vote	*"Vote"*	Yes	Voting for an expired election	Voting-Vote-S1	Yes
30	VoteDapp	`vote()`	Voting-Vote	*"Vote"*	Yes	Voting for an expired election	Voting-Vote-S1	Yes
31	voting-DApp	`vote()`	Voting-Vote	*"Vote"*	Yes	Voting for an expired election	Voting-Vote-S1	Yes
32	VoteMe	-	-	-	Yes	-	-	Yes
33	Overview	`invest()`	CS-Invest	*"Buy tokens", "Crowdsale"*	Yes	Invest an expired crowdsale	CS-Invest-S2	Yes
34	Crowdsale	-	-	-	Yes	-	-	Yes

ILLINOIS INSTITUTE OF TECHNOLOGY

Evaluation

- Source analyzability

```
1  function appendInt(buffer memory buf, uint data, uint len)
2      internal constant returns(buffer memory) {
3      ...
4      assembly {
5          let bufptr := mload(buf)
6          let buflen := mload(bufptr)
7          let dest := add(add(bufptr, buflen), len)
8          mstore(dest, or(and(mload(dest), not(mask)), data)
       )
9          mstore(bufptr, add(buflen, len))
10     }
11     return buf;
12 }
```

- Comparison with VerX

Table 6: VETSC vs. VerX

Analysis Step	VETSC	VerX
(a) High-level Specs	One-Time Manual Effort	None
(b) Semantic Recovery	Automated (UI-Guided)	Manual Effort for Each Func
(c) Specs Customization	Automated	Manual Effort for Each Func
(d) Safety Verification	Automated	Automated