Table 8: Categories of existing EVM operations.

PCs	EVM Operations	Functionality
0x00-0x0B	STOP,ADD,MUL,*etc*	Stop and Arithmetic Operations
0x10-0x1D	LT,GT,SLT,*etc*	Comparison and Bitwise Logic Operations
0x20	KECCAK256 (*a.k.a.*, SHA3)	KECCAK256 Method
0x30-0x3F	ADDRESS,BALANCE,*etc*	Transaction Properties
0x40-0x45	BLOCKHASH,COINBASE,*etc*	Block Properties
0x50-0x5B	POP,MLOAD,*etc*	Stack, Memory, Storage and Flow Operations
0x5F-0x7F	PUSH0-PUSH32	Push Operations
0x80-0x8F	DUP1-DUP16	Duplication Operations
0x90-0x9F	SWAP1-SWAP16	Exchange Operations
0xA0-0xA4	LOG0-LOG4	Logging Operations
0xF0-0xFF	CALL,RETURN,DELEGATECALL	System Operations

Algorithm 2: Semantic Units Lifting

```
1  Function SemanticUnitLifter(Bytecode, 𝕋):
2      𝕌 ← ∅
3      ℙ ←PathPreparation(Bytecode, 𝕋)
4      foreach P ∈ ℙ do
5          ℂ ← ∅
6          foreach b ∈ P do
7              UpdateConditions(ℂ)
8              foreach ins ∈ b do
9                  if IsJUMPI(ins) then
10                     ℂ ←AddCondition(ins)
11                 else if IsSAI(ins) then
12                     ℂ′ ←DataControlFlowAnalysis(ℂ)
13                     𝔹′ ←DataFlowAnalysis(ins)
14                     𝕌 ← 𝕌 ∪ (ℂ′, 𝔹′)
15     return 𝕌
```

A EVM OPERATIONS CATEGORIES

According to existing classification methods [3, 54], current EVM operations can be categorized into 11 groups. Detailed classification results and corresponding operation functions are provided in Table 8. Notably, among the more than 140 total operations, only four types of operations would affect the blockchain's state, which is outlined in Table 1. These include updating state variables of contracts, contract invocation, transfer behavior, contract creation and destruction.

B DETAILS OF SEMANTIC UNITS LIFTING

Algorithm 2 describes the details for extracting semantic units $\mathbb{U}$ from EVM bytecode `Bytecode` and its corresponding transaction information $\mathbb{T}$. First, we analyze the bytecode and construct the Control Flow Graph based on Ethersolve [12], obtaining some paths. Later, by combining these paths with transaction information, we get execution paths $\mathbb{P}$ (Line 3). For each path, we initialize the global conditions as empty (Line 5). Then, for each block in this path, based on existing conditions, when the current block is out of the control scope of the previous condition, we pop the previous condition to update the conditions (Line 7). For each instruction within the block, we update conditions and semantic information based on the instruction. Specifically, for the `JUMPI` instruction, we analyze its corresponding conditions (Line 10). For state-affected-instructions (SAI), we utilize control flow and data flow analysis to determine semantic information, ultimately merging them to semantic units (Line 12-14).

Algorithm 3: Type-aware Name Inference

```
1  Function Training(𝔾):
2      𝕊𝕍 ←CollectStateVariables(𝔾)
3      𝕊𝕍𝕃 ← Clustering(𝕊𝕍)
4      𝔾′ ← AddLabels(𝔾, 𝕊𝕍𝕃)
5      while not converged do
6          𝔾′_b ←Sampling(𝔾′)
7          model ← RGCN(𝔾′_b, model)
8      𝔾_l ←NameInference(𝔾′, model)
9      return 𝔾_l
10 Function Clustering(𝕊𝕍):
11     𝕊𝕍𝕃 ← ∅
12     𝕋 ←GroupByTypes(𝕊𝕍)
13     foreach type ∈ 𝕋 do
14         Features ←GetFeatures(type_names)
15         Labels ←HDBSCAN(Features)
16         𝕊𝕍𝕃 ← 𝕊𝕍𝕃∪AssignLabels(type_names, Labels)
17     return 𝕊𝕍𝕃
```

Table 9: Parameters used for unknown name inference.

Stage	Parameters	Value
Cluster	Type consideration	Yes
	Size	5
	Neighbors	10
Training	lr	0.063364
	n_hidden	256
	n_layers	2
	n_bases	40
	dropout	0.309769
Loss	loss_gamma	4

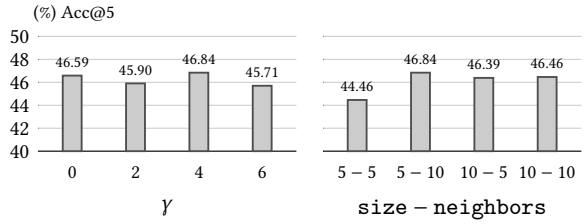

Figure 10: Acc@5 of different parameters for name inference.

C DETAILS OF NAME INFERENCE

algorithm 3 describes the details for inferring unknown variable names based on some known variable names, types, and the graph topology $\mathbb{G}$. First, we collect all state variable names and their types $\mathbb{SV}$ on the graph (Line 2). Later, by invoking the Clustering method, we obtain labels $\mathbb{SVL}$ (Line 3), and these labels are then added to the graph (Line 4). Utilizing these labels, we train a graph neural network `model` (Lines 5-7). Finally, the model is applied to infer unknown variable names (Line 8).

For the Clustering method, we first group different variable names based on their types (Line 12). For each type, we use the publicly available pre-trained model SentenceBert [55] and get features (Line 14). These features are applied to HDBSCAN [45] for clustering (Line 15), thereby obtaining labels (Line 16).

Table 9 shows the parameters utilized by Eddy. To obtain these parameters, we first obtain cluster parameters by freezing other parameters. Then, we use Neural Network Intelligence [46] to tune the model parameters. Finally, we explore the effect of different `loss_gamma` on the final result and select the best `loss_gamma`. Figure 10 shows the Acc@5 for different `loss_gamma` and clustering parameters.

D RESULTS FOR MANUAL EVALUATION

Table 10 shows all the cases that we can generate semantically-equal contracts base on our descriptions. The full observations are published in https://github.com/Eddy-artifacts/Eddy.

Table 10: Detailed manual evaluation results on SC_{manual}. I1 is the bytecode, I2 is the decompiled code and I3 is descriptions.

Contract	Language	New	I1	I2	I3
0x23ea10cc1e6ebdb499d24e45369a35f43627062f	Solidity	0	0	1	1
0x899f9a0440face1397a1ee1e3f6bf3580a6633d1	Solidity	0	0	0	1
0x2956356cd2a2bf3202f771f50d3d14a367b48070	Solidity	0	0	0	1
0xa6d93468c1f16f2e1009ec321251f0b3147117b3	Solidity	0	0	1	1
0x44e081cac2406a4efe165178c2a4d77f7a7854d4	Solidity	0	0	0	1
0x5535a72556727c221c567e0fc4208c5a99dba1cc	Solidity	0	0	0	1
0x53b04999c1ff2d77fcdde98935bb936a67209e4c	Solidity	0	0	0	1
0x5da60592329d7651ff3415ef7f3d3ed91d1c2f24	Solidity	0	0	1	1
0x93d812bf90a575d628e246b0966505a9e466f534	Solidity	0	0	1	1
0x1bb28e79f2482df6bf60efc7a33365703bcf1536	Solidity	0	0	1	1
0x2cbc6812cff0b1113bf2808ffce6d83b97afd345	Solidity	0	0	1	1
0xe87227adf0fd3f6e580e2825069a0f8e8da66ad0	Solidity	0	0	0	1
0x14f109d126c58b5bf55cdc51b9bd90b21e6a6de5	Solidity	0	0	0	1
0x0e915b35cc269b2dfc8bbd8e4a88ed4884a53efc	Solidity	0	0	0	1
0xc0950ee7568e1ab93c6784ddcaa9b590239eeec9	Solidity	0	0	0	1
0x22e5f62d0fa19974749faa194e3d3ef6d89c08d7	Solidity	0	0	0	1
0x549b38e8e5fb1ba4acd4187d8b7c8a04deb53206	Solidity	0	0	0	1
0x4363b5d64f228c819dc706889b09a0dc76e22fb0	Solidity	0	0	0	1
0x2baac9330cf9ac479d819195794d79ad0c7616e3	Solidity	0	0	0	1
0xd0955bd45d5eef5bedce84a3471c825288a14b01	Solidity	0	0	0	1
0xac83edfd549ca5eaf55b21fe936dd9529403152a	Solidity	0	0	1	1
0xa8df33a40fe2e3278e4d94a974f70778043fbd20	Solidity	0	0	0	1
0xcbe98a2b1f756bebe53d41eb3b94e566a0777ede	Solidity	0	0	0	1
0xbb27cdd72fc42a538727f29f23d47972b8dfa405	Solidity	0	0	0	1
0x2193b9f2b5e5051e41c32c71b25587d6a6f045c8	Solidity	0	0	0	1
0xfe38739c4b70cd6d1d3674e70d17fb1999cb4f9f	Solidity	0	0	0	1
0x9e4ad79049282f942c1b4c9b418f0357a0637017	Solidity	0	0	0	1
0xb45d7bc4cebcab98ad09babdf8c818b2292b672c	Solidity	0	0	0	1
0x8c8ccb81d436b0f3017664441c39cbefbd64650f	Solidity	0	0	0	1
0x41634a7cb08ee639a751973e63577ddcfe9e7101	Solidity	0	0	0	1
0x1beb353fa1e215457ff16fceec07f7eab9f06565	Solidity	0	0	1	1
0xcc1320a48c76385e17e4e1c759ce8ea1d11687c2	Solidity	0	0	0	1
0x68af0f18c974a9603ec863fefcebb4ceb2589070	Solidity	0	0	0	1
0x3cab5f3e3bbe86c69f4e43339920f61ab8193300	Solidity	0	0	0	1
0x2f5c8d7259b48078c0b6c0a7cf6269e3817680e5	Solidity	0	0	0	1
0x6a57883b5748bf3631ac2e0d43bf0d6f6cbcd16b	Solidity	0	0	0	1
0xac0007d373df6ee6f0df62e85084d5af4db3740f	Solidity	0	0	0	1
0x83ba853b8196bb51c148dd762a827c703b156450	Solidity	0	0	0	1
0x95cc9c7bf003d948dea3a5a95116f7fd9ca43778	Solidity	0	0	0	1
0x54b0de285c15d27b0daa687bcbf40cea68b2807f	Solidity	0	0	0	1
0x47423b0fdb181ecab813c908307e9795c0272db7	Solidity	0	0	1	1
0x6f95f1ce0c8d0257fe993b2358aca29ea0506943	Solidity	0	0	0	1
0x48e4562b66df6824811c1171fdf9d5814a5b42ef	Solidity	0	0	0	1
0x6f303642844f734ad4176d0dfe93ef7e0776ef46	Solidity	0	0	0	1
0xa04e5b78fbd31caec8f8af126d00a57f56c1f7ae	Solidity	0	0	0	1
0xad078f05b048ca1a5025d1943cce35da6bf42aa4	Solidity	0	0	0	1
0xd8f1c224002ce1782a3100793fb1ddc102f4723f	Solidity	0	0	0	1
0xfb0f7189b354660e649ae14261a9fe0e8febf369	Solidity	0	0	0	1
0x8ecb6d18b2163d955fb9171fec1fb4f9382001bc	Solidity	0	0	1	1
0x8207bcaacd247a03fbd68e1a941092fa678cf492	Solidity	0	0	0	1
0x05b3abd9031a31a45121bda59c7bb52fc7db2590	Solidity	0	0	1	1
0xb222e54b336bf4834953eac0a5bd0c1aae2079e9	Solidity	0	0	1	1
0x6cc12c718c924c63b9fa90cff0722054e408049a	Solidity	0	0	1	1
0xcd19a08b139a4cc47bae0d9de41241f5056988c7	Solidity	0	0	1	1
0x37f387fe48c9569500b514a1c7e1f6b04be8d421	Solidity	0	0	0	1
0x39cfd754c85023648bf003bea2dd498c5612abfa	Solidity	0	0	0	1
0x3a0e9acd953ffc0dd18d63603488846a6b8b2b01	Solidity	0	0	0	1
0xa0cc94083c43a027071f6ccaee7251fbd818b7c6	Vyper	0	0	0	1
0x0d8fc15b6fefc278ff642861df51b45607330871	Vyper	0	0	0	1
0xc0a47dfe034b400b47bdad5fecda2621de6c4d95	Vyper	0	0	0	1
0x1abf3e1c4875b5bcfaa63344ae729aa30c93a74b	Vyper	0	0	0	1
0x076a7c93343579355626f1426de63f8827c9b9b2	Vyper	0	0	0	1
0x64d56f087d87cdaeac8119c69c48d0d440d560a7	Vyper	0	0	0	1
0xce616a55dd45ec16c3db4151ea4cb7f8f30b999a	Solidity	1	0	0	1
0xa83c2fa8e9f8e96a40f13e0129bf52ea316b6b25	Solidity	1	0	0	1
0x0e8eb5459194def55d88b0f992cc2e13794181dd	Solidity	1	0	0	1
0x7b1eadc263871fbc8c4ed152e844a79fa08ccf81	Solidity	1	0	0	1
0x8f90837968f9631ff79514dc1a264fb008f3802b	Solidity	1	0	0	1
0x99f2bd643c5c96cfbddb69d4dfc5c0dec4551c33	Solidity	1	0	0	1
0x7cc551b23ded82fa4109c62b6cdc97592539de95	Solidity	1	0	0	1
0xb5bcb4386ae3565bb5ba0c16dac1670ff53f0960	Solidity	1	0	1	1
0x050f9a317ee0602ccb0a4cf99c1567b21c121031	Solidity	1	0	0	1
0xf377960dbde17e28a817e4479133d67d576bf476	Solidity	1	0	1	1
0xe5db444db675746636a17e938ce3c2051e9b96d3	Solidity	1	0	0	1
0xa13f530f00fab432a1c3989449d4a350b3c25429	Solidity	1	0	0	1
0x22616b4429c410b543bb7fc050a6a83890ecc6f3	Solidity	1	0	0	1
0xdbdd2db0bc00930cf874af8281a9590d6e7b72e1	Solidity	1	0	0	1
0xb3e5bd1ea9cc6e44939b6925272061e401b5f365	Solidity	1	0	0	1
0xa05fef8806f83a732df491235f195355a004a1dc	Solidity	1	0	1	1
0xecb168408adebd5383944c082853128bc7925773	Solidity	1	0	1	1
0x30115ccf6d77f71a577e98ce2732d462f1464959	Solidity	1	0	0	1
sum			0	18	81