Supplementary Material for Hermes

A PROOF OF DETECTION EQUIVALENCE BETWEEN SEG AND LSEG

We now formally prove that Hermes, using Algorithms 1 and 2 over the LSEG G_{lazy}, achieves the same bug detection results as performing bug search over the SEG G_{full}, given the same source-sink specification. The proof proceeds by showing that both searches explore structurally corresponding sets of source-sink paths, and that such correspondence implies detection equivalence.

A.1 Path Isomorphism Between SEG and LSEG

Definition A.1 *(Path Isomorphism).* Let $\pi_1 = \langle n_0^1, \ldots, n_k^1 \rangle$ be a path in G_{full}, and $\pi_2 = \langle n_0^2, \ldots, n_k^2 \rangle$ a path in G_{lazy}. We say $\pi_1 \sim \pi_2$ (they are *isomorphic*) iff for all $i \in [0, k]$:

- n_i^1 and n_i^2 are non-interface nodes, then they refer to the same program value, i.e., n_i^1 and n_i^2 satisfy $\mathsf{value}(n_i^1) = \mathsf{value}(n_i^2)$, where $\mathsf{value}(n)$ denotes the program value of node n.
- n_i^1 and n_i^2 are interface nodes, then they refer to the same symbolic memory location; equivalently, $\mathsf{accessPath}(n_i^1) = \mathsf{accessPath}(n_i^2)$. Here, $\mathsf{accessPath}(n)$ denotes the access path associated with the symbol value carried by n (Remark 3.1).

Lemma A.1 *(Path Condition Preservation under Isomorphism).* Let $\pi_1 = \langle n_0^1, \ldots, n_k^1 \rangle$ and $\pi_2 = \langle n_0^2, \ldots, n_k^2 \rangle$ be two isomorphic paths in G_{full} and G_{lazy}, respectively, such that $\pi_1 \sim \pi_2$. Then their path conditions are equivalent:

$$\Phi_{\pi_1} = \Phi_{\pi_2}$$

where Φ_π denotes the path condition of the value-flow path π, as defined in Definition 3.3.

Proof. By the construction of G_{lazy} (Algorithm 2), any value flow $n_i^2 \longrightarrow n_{i+1}^2 \in \pi_2$ falls into one of the following two cases:

- It is presented as an intra-procedural value flow with the same control dependencies as its counterpart $n_i^1 \longrightarrow n_{i+1}^1$, since intra-procedural flows in G_{lazy} and G_{full} are constructed identically.
- It is resolved via `ResolveFlows` (Algorithm 2, Line 15 and 27), which constructs and inserts a value flow $n_i^2 \longrightarrow n_{i+1}^2$ such that $n_i^1 \longrightarrow n_{i+1}^1 \in G_{full}$ and $\mathsf{accessPath}(n_{i+1}^1) = \mathsf{accessPath}(n_{i+1}^2)$. As access paths uniquely determine both the loaded values and their guard constraints, the control dependencies along $n_i^2 \longrightarrow n_{i+1}^2$ mirror exactly those in G_{full}.

In both cases, each value flow in π_2 has an exact counterpart in π_1 with identical control dependencies. Therefore, we conclude that $\Phi_{\pi_1} = \Phi_{\pi_2}$. □

Author's address:

XXXX-XXXX/2025/10-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

A.2 Detection Equivalence under Path Isomorphism

Lemma A.2 *(Path Set Isomorphism Implies Detection Equivalence).* Let $\mathcal{P}_{full}$ and $\mathcal{P}_{lazy}$ be the sets of source-sink paths collected using Algorithm 1 from G_{full} and G_{lazy}, respectively. Suppose that:

$$\forall \pi_1 \in \mathcal{P}_{full},\ \exists\, \pi_2 \in \mathcal{P}_{lazy} \text{ s.t. } \pi_1 \sim \pi_2, \text{ and vice versa.}$$

Then the bug detection results (Definition 3.3) are equivalent for $\mathcal{P}_{full}$ and $\mathcal{P}_{lazy}$.

Proof. By Definition 3.3, the detection result is determined by the satisfiability of Φ_π for paths π connecting σ_{src} to σ_{sink}. Under the stated hypothesis, all potential source-sink paths are in 1-to-1 isomorphic correspondence across the two graphs. By Lemma A.1, their path conditions are equal. Therefore, the satisfiability of one implies that of the other. Thus, detection results are identical. □

A.3 Path Set Equivalence between SEG and LSEG

Lemma A.3 *(Path Set Equivalence between SEG and LSEG).* Let $\mathcal{P}^{\tau}_{full}$ and $\mathcal{P}^{\tau}_{lazy}$ be the sets of all source-sink paths under specification τ in G_{full}, G_{lazy}, respectively. Then:

$$\mathcal{P}^{\tau}_{full} \cong \mathcal{P}^{\tau}_{lazy}.$$

That is, for every path $\pi \in \mathcal{P}^{\tau}_{full}$, there exists a unique $\pi' \in \mathcal{P}^{\tau}_{lazy}$ such that $\pi \sim \pi'$, and vice versa.

Proof. Let $\pi \in \mathcal{P}^{\tau}_{full}$ be an arbitrary source-sink path in G_{full}. We aim to show that there exists a corresponding source-sink path $\pi' \in \mathcal{P}^{\tau}_{lazy}$ such that $\pi \sim \pi'$. We then prove the reverse direction, i.e., for any $\pi' \in \mathcal{P}^{\tau}_{lazy}$, there exists a corresponding $\pi \in \mathcal{P}^{\tau}_{full}$.

Forward direction: We distinguish two cases depending on whether π contains any inter-procedural value flows.

Intra-procedural Case. Suppose $\pi \in \mathcal{P}^{\tau}_{full}$ is an intra-procedural source-sink path. Since all intra-procedural value flows are initially constructed in G_{lazy} (Definition 3.5), there exists a path $\pi' \in \mathcal{P}^{\tau}_{lazy}$ such that $\pi \sim \pi'$.

Inter-procedural Case. Suppose π contains one or more inter-procedural value flows. We decompose π into alternating intra- and inter-procedural components as follows:

$$\pi = \pi_0 \cdot e_1 \cdot \pi_1 \cdots e_k \cdot \pi_k.$$

Here, $k \geq 1$, each π_i is an intra-procedural subpath, and each $e_j = (n_j \rightarrow n_{j+1})$ is an inter-procedural value flow.

By Definition 3.5, each intra-procedural subpath π_i is already present in G_{lazy}. For each inter-procedural edge $e_j = (n_j \rightarrow n_{j+1})$, it must be one of the following two types:

- **Case 1 (Input value flow):** $n_i \in N_{in},\ \ n_{i+1} \in N_{param}$. This value flow is resolved by Algorithm 2, Line 8-15. Hermes computes the access path of the parameter node n_{i+1} based on the actual argument at the callsite, and inserts the value flow $n'_i \rightarrow n'_{i+1}$ into G_{lazy}.
- **Case 2 (Return value flow):** $n_i \in N_{ret},\ n_{i+1} \in N_{out}$. Hermes resolves this via Algorithm 2, Line 19-27, where VAG-guided loading recovers the value returned by the callee. It then creates a value flow $n'_i \rightarrow n'_{i+1}$ linking the auxiliary-return node to its auxiliary-output node.

Therefore, all components of π—both intra- and inter-procedural—are present in G_{lazy}, either initially or added on demand. By concatenating these components in the same order, we obtain a source-sink path $\pi' \in \mathcal{P}^{\tau}_{lazy}$ such that $\pi \sim \pi'$.

Reverse direction: We now show that for every $\pi' \in \mathcal{P}^{\tau}_{lazy}$, there exists a corresponding $\pi \in \mathcal{P}^{\tau}_{full}$ such that $\pi \sim \pi'$.

By construction, the SEG G_{full} contains all possible inter-procedural value flows. The Lazy SEG G_{lazy} is a strict subgraph of G_{full} that initially excludes interface nodes along with their associated value flows, and only includes them on demand during bug search via Algorithm 2. Therefore, any path π' constructed in G_{lazy} is also present in G_{full}. The corresponding path $\pi \in \mathcal{P}^{\tau}_{full}$ exists by definition and satisfies $\pi \sim \pi'$.

Hence, we conclude:

$$\mathcal{P}^{\tau}_{full} \cong \mathcal{P}^{\tau}_{lazy}. \qquad \square$$

A.4 Detection Equivalence between SEG and LSEG

Theorem A.1 *(Detection Equivalence between LSEG and SEG).* Given the same source-sink specification τ, Hermes detects the same bugs as a SEG-based bug search using Definition 3.3.

Proof. This follows from Lemma A.3, which establishes a bijective correspondence between source-sink paths in the two graphs, and Lemma A.2, which shows that such correspondence preserves detection outcomes under Definition 3.3. $\square$