Heuristic Decisions in Static Analysis

- **Declarative**: what, not how
- **Correctness**: more clear

Automatically Generating Heuristics from Data

- **Automatic**: little reliance on analysis designers
- **Powerful**: machine-tuning outperforms hand-tuning
- **Stable**: can be generated for arbitrary programs

Selecting Context-Sensitivity

- **Context-insensitivity** fails to prove the queries
- **2-call-site-sensitivity** succeeds but not scale

Performance

- **Training** with 4 small programs from DaCapo, and applied to 6 large programs
- **Machine-tuning** outperforms hand-tuning

Learning Algorithm

Let \( \Pi = \{ f_1, \ldots, f_k \} \) be parameters. Each \( f_i \) expresses methods to be assigned with depth \( i \). We assign deeper depth if a method is in both \( f_i \) and \( f_j (i \neq j) \). We learn \( \Pi \) by solving the following problem.

**Optimization problem**

Find parameter \( \Pi = \{ f_1, \ldots, f_k \} \) that minimizes the cost of analysis while satisfying precision constraint over training set.

**Challenge**

Assuming that \( |S| \) is the space of possible boolean formulas over which we learn, search space of original problem is \( |S|^k \). We reduce the space into \( k + |S| \) by decomposing the original problem into \( k \) subproblems \( \Psi_i \). Each \( f_i \) is obtained from \( \Psi_i \) and we solve them from \( \Psi_1 \) to \( \Psi_k \).

Decomposed problem \( \Psi_i \)

Let \( \Pi = \{ f_1, \ldots, f_k \} \). Find formula \( f_i \) that makes \( \Pi \) minimize the cost while satisfying precision constraint over training set.

**Learning Algorithm for \( \Psi_i \)**

To solve \( \Psi_i \), we made a greedy algorithm. Let \( \{ f_1, \ldots, f_k \} \) be atomic features. Our algorithm proceeds in the following steps:

1. \( f_i \) starts from disjunctions of 2n clauses:
   \[ f_i = a_1 \lor \neg a_1 \lor \cdots \lor a_n \lor \neg a_n \]
2. Choose the most expensive clause \( c_i \) to refine.
3. Strengthen the clause \( c_i \) by conjointing an decent atom \( a_j \) with \( c_i : f_i' = c_1 \lor \cdots \lor (c_i \land a_i) \lor \cdots \lor c_p \).
4. Check if \( f_i' \) satisfies precision constraint. If it is, \( f_i = f_i' \).
5. Repeat 2~4 until \( f_i \) cannot be refined.

**Data-Driven Context-Sensitivity for Points-to Analysis**

Seun Jeong*, Minseok Jeon*, Sungdeok Cha, and Hakjoo Oh

- **Objective**: What makes a points-to analysis powerful? How to automate? Analyze sensitivities?

**Key Contributions**

- We achieve the improvement with two key ideas.
- A new expressive model (Disjunctive Model)
- Learning algorithm for new model

**Disjunctive Model**

Disjunctive model expresses set with DNF form.

**Methodology**

- **Features**:
  - Atom (Stat vs Rule)
  - Disjunctive Model(Possible):
    - \( M_1 : a_1 \lor a_2 \lor a_3 \)
  - Linear Model(Impossible):
    - \( M_3 : a_2 \lor a_3 \)

**Figure 3: Disjunctive vs Linear**

With \( \{ a_1, a_2 \} \), Disjunctive model can express the target methods, but Linear model cannot.

**Learning Algorithm**

Let \( \Pi = \{ f_1, \ldots, f_k \} \) be parameters. Each \( f_i \) expresses methods to be assigned with depth \( i \). We assign deeper depth if a method is in both \( f_i \) and \( f_j (i \neq j) \). We learn \( \Pi \) by solving the following problem.

**Optimization problem**

Find parameter \( \Pi = \{ f_1, \ldots, f_k \} \) that minimizes the cost of analysis while satisfying precision constraint over training set.

**Challenge**

Assuming that \( |S| \) is the space of possible boolean formulas over which we learn, search space of original problem is \( |S|^k \). We reduce the space into \( k + |S| \) by decomposing the original problem into \( k \) subproblems \( \Psi_i \). Each \( f_i \) is obtained from \( \Psi_i \) and we solve them from \( \Psi_1 \) to \( \Psi_k \).

**Decomposed problem \( \Psi_i \)**

Let \( \Pi = \{ f_1, \ldots, f_k \} \). Find formula \( f_i \) that makes \( \Pi \) minimize the cost while satisfying precision constraint over training set.

**Learning Algorithm for \( \Psi_i \)**

To solve \( \Psi_i \), we made a greedy algorithm. Let \( \{ f_1, \ldots, f_k \} \) be atomic features. Our algorithm proceeds in the following steps:

1. \( f_i \) starts from disjunctions of 2n clauses:
   \[ f_i = a_1 \lor \neg a_1 \lor \cdots \lor a_n \lor \neg a_n \]
2. Choose the most expensive clause \( c_i \) to refine.
3. Strengthen the clause \( c_i \) by conjointing an decent atom \( a_j \) with \( c_i : f_i' = c_1 \lor \cdots \lor (c_i \land a_i) \lor \cdots \lor c_p \).
4. Check if \( f_i' \) satisfies precision constraint. If it is, \( f_i = f_i' \).
5. Repeat 2~4 until \( f_i \) cannot be refined.