**Goal**

Global analysis of million lines of programs: Techniques for reducing analysis time and memory

**Reducing Analysis Time**

**Key Idea: Sparse Analysis**

In global analysis of imperative programs, the analysis is typically sparse in space and time.

Sparse analysis directly follows the actual semantic dependences.

**Challenges**

- The dependences are unavailable prior the analysis
- The use of conservative dependences needs cares
- Thus, previous techniques either lose precision or need problem-specific safety devices

**Sparse Analysis Framework**

\[ \hat{F} : \hat{D} \rightarrow \hat{D} \Rightarrow \hat{F}_s : \hat{D} \rightarrow \hat{D} \]

\[ \text{lfp} \hat{F} = \text{lfp} \hat{F}_s \]

**Reducing Memory Consumption**

Our current performance bottleneck is in memory cost, especially in representation of dependence graphs
- `def-use` graph for Vim72 (350KLOC) > 30GB

**Problem**

Abstract locations on `def-use` graph are highly redundant
- `sendmail-8.13`, locations: 17,270, `#define`: 37,943,586
- sharing in applicative languages are not enough

**Binary Decision Diagrams**

- An efficient representation of boolean functions
- Powerful in reduction and sharing

**Encoding of Def-use Graph**

- An efficient representation of boolean functions
- Powerful in reduction and sharing

For `sendmail-8.13.6` (130KLOC) the use of `BDDs` can represent similar sets efficiently