Review of

Pointer Analysis: Haven’t We Solved This Problem Yet?
By Michael Hind

Kevin Depue
February 16, 2006
What problem did the paper address?

**Big Picture Problem**

How do we gain an understanding of how pointers affect program behavior?

**Why problem is hard**

- In general, pointer analysis is an undecidable problem.
- To make the problem decidable, we must make conservative approximations.
- Have to make sacrifices between cost and precision.

**Specific Problem**

Why hasn’t this problem been solved, and what issues/open problems remain?
Why should we care?

Pointer analysis is important because it is used in:
- Optimizing compilers
- Program understanding tools
- Error detection

The paper brings the current approaches to pointer analysis together into one paper. This is important because over 75 papers and 9 Ph.D. theses have been written on pointer analysis.

It provides a direction and context for future research in the area.
What is the approach used to solve the problem?

The author evaluates and compares the existing approaches, and identifies issues and open questions.

Issues
- Terminology
- Metrics
- Reproducible results

Open Questions
- Scalability – How do we make pointer analysis precise/feasible for large programs?
- Improving precision – How can we improve precision without losing scalability?
- Designing an analysis for a client’s needs
- Flow-sensitivity
- Context-sensitivity – Does context-sensitivity improve precision?
- Heap modeling – Does heap modeling scale for large programs?
- Modeling aggregates
- Demand-driven/incremental analysis
- Pointer analysis for object-oriented languages (e.g., dynamic dispatch)
- Incomplete programs
- Engineering insights
An Issue: Metrics

How do we measure the precision of a pointer analysis?

**direct metric:** Average number of objects aliased to pointer expressions in a program.

**Flaws:**
- An analysis models an unbounded number of dynamic objects. This can corrupt the average.
- The direct metric does not necessarily explain the effect on a client analysis.

**Alternatives:**
- Compare the direct metric to worst-case assumptions
- Client-driven pointer analysis
  - **Example:** In the [LIN] paper (Tuesday) we see that the *number of program errors* is used as the precision measure.
- Dynamic profiling
What is the approach used to solve the problem?

The author evaluates and compares the existing approaches, and identifies issues and open questions.

Issues
• Terminology
• Metrics
• Reproducible results

Open Questions
• Scalability – How do we make pointer analysis precise/feasible for large programs?
• Improving precision – How can we improve precision without losing scalability?
• Designing an analysis for a client’s needs
• Flow-sensitivity
• Context-sensitivity – Does context-sensitivity improve precision?
• Heap modeling – Does heap modeling scale for large programs?
• Modeling aggregates
• Demand-driven/incremental analysis
• Pointer analysis for object-oriented languages (e.g., dynamic dispatch)
• Incomplete programs
• Engineering insights
An Open Question: Modeling Aggregates

Are aggregates of an object distinguished from that object or collapsed into one object? Distinguishing aggregates will increase precision and cost, but by how much is not well known. Is the increase in precision worth the increase in cost?

Example:

```c
struct A {
    int* x;
    int* y;
};

int main() {
    ..

    int w = 6;
    int z = 5;
    A a;

    a.x = &w;
    a.y = &z;

    ..
}
```

Aggregates collapsed into one object:

{a->w, a->z}

Aggregates distinguished:

{a.x->w}, {a.y->z}
How does the paper support the conclusions it reaches?

Conclusions

- Client-driven pointer analysis may be a good approach to help the cost/precision problem (3-4 experts mention this in the paper).
- Metrics used to measure precision should be appropriate to the particular client analysis being investigated.
- Programmer annotations as a means to increase precision and lower cost should be explored.
- As long as we have to worry about the cost of pointer analysis, it will remain an unsolved problem.
- *Pointer analysis needs a lot of work!*

The author validates his conclusions based on the literature and opinions of experts in the field.
Future Research

All the open questions noted in the paper can be considered future research opportunities.

Programmer annotations as a means to increase precision and lower cost should be explored.

Could a pointer analysis be driven by an external client? (Yes, as we saw on Tuesday)
Critique

This is a survey paper.

The paper does a good job of describing the current state of the field (as of 2001). The author saved us a lot of time.

The author gathered quotes and opinions from experts in the field. The result is a variety of different viewpoints and possible solution approaches, which helps eliminate bias on behalf of the author.

The paper provides a good framework for future research.

The paper could have given some concrete examples of the different open questions.
Relation to CS653

OpenAnalysis implements a pointer analysis.

Static data-flow analyses require knowledge about pointers.