# Constraint-based Recognition of Program Plans in Legacy Code

Steven Woods

Ph.D. Candidate

Department of Computer Science

University of Waterloo,

Waterloo, Ontario, CANADA

March 22, 1995

# Presentation Outline

1. Introduction to Legacy Understanding Problem
2. Overview of Constraint Satisfaction Paradigm
3. Understanding as Constraint Satisfaction
4. Previous Work and Shortcomings
5. Ongoing Work

# The Problem with Legacy Code

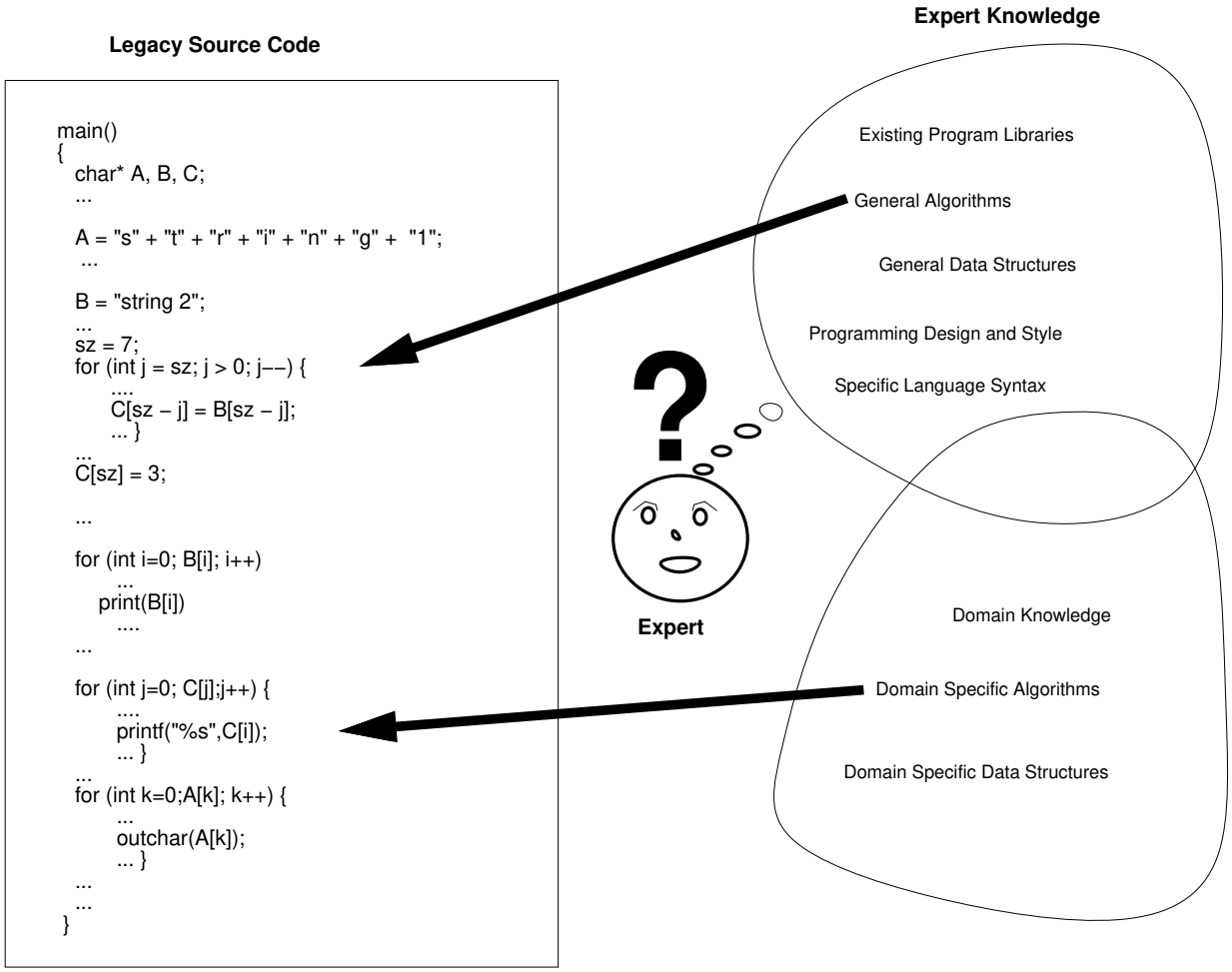
- *Large* legacy code source presents many challenges :
  - Difficult to maintain, extend, debug
  - Old, embedded unused code
  - Outdated or non-existent documentation
  - **Redundant code functionality**
  - **Dispersed, local, or non-existent knowledge of code function,libraries**
  - ... many other problems

## Dealing with Legacy Code

- Maintainers attempt to alleviate these problems by :
  - $\longrightarrow$ Rewriting all or some systems if practical
  - $\longrightarrow$ Consolidating code functionality where possible
  - $\longrightarrow$ Simplifying code via shared or commerical libraries if available
- Maintainers, knowledgeable about the domain, must somehow *Understand* a given legacy source

But what do we mean by "*Understand* legacy code" ?

# Understanding Legacy Code : Conceptualization

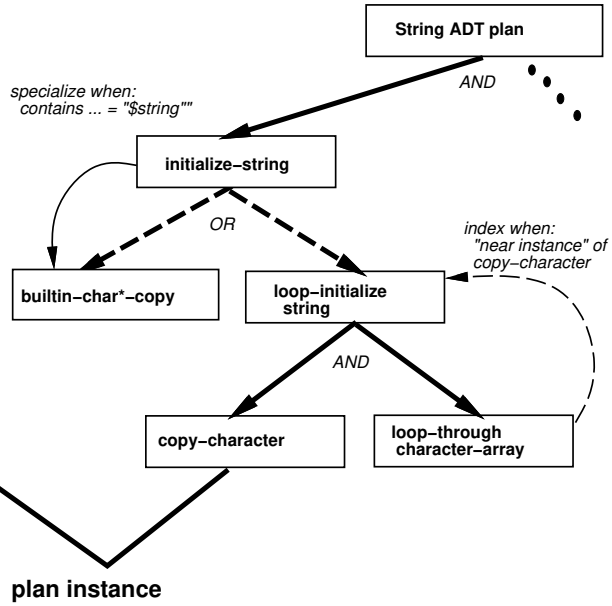


# Understanding Legacy Code : Knowledge Mapping

**Legacy Source Code**

```
main()
{
  char* A, B, C;
  ...
  A = "s" + "t" + "r" + "i" + "n" + "g" +  "1";
   ...
  B = "string 2";
  ...
  sz = 7;
  for (int j = sz; j > 0; j––) {
      ....
      C[sz – j] = B[sz – j];
      ... }
  ...
  C[sz] = 3;
  ...
  for (int i=0; B[i]; i++)
      ...
      print(B[i])
      ....
  ...
  for (int j=0; C[j];j++) {
      ....
      printf("%s",C[i]);
      ... }
  ...
  for (int k=0;A[k]; k++) {
      ...
      outchar(A[k]);
      ... }
  ...}
```

**Program Plan Library (excerpt)**




→ **How may we automate some of this mapping process ?**

# Approach : Constraint Satisfaction Paradigm

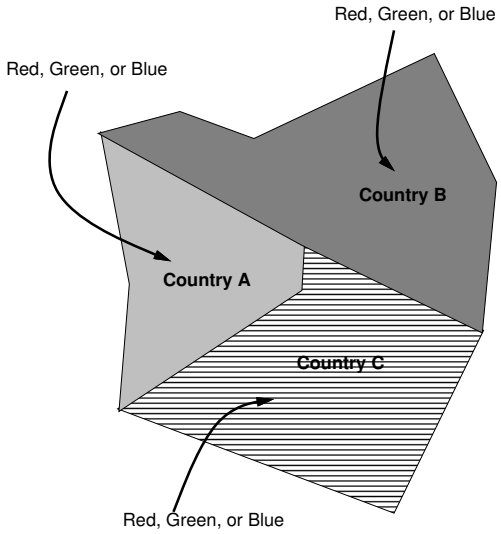



Figure 1: Map Color Example.

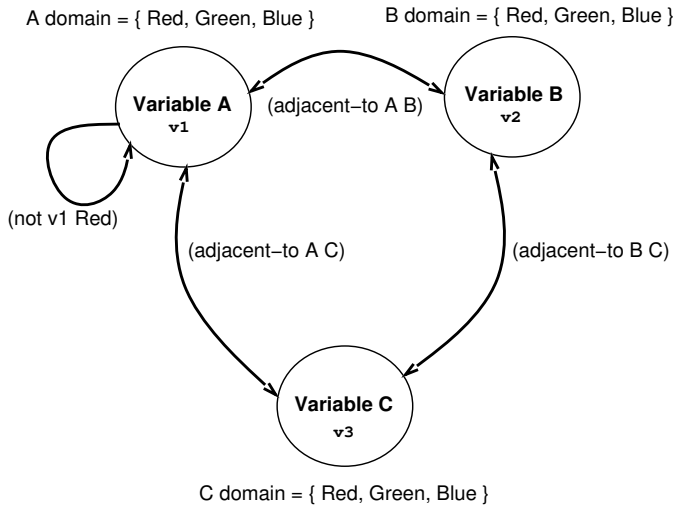



Figure 2: Map Color CSP.

- Known solution approaches (Search, Propagation, Hybrid)
- Well constrained problems $\longrightarrow$ reasonable performance
- Domain knowledge (i.e. plan library) $\longrightarrow$ tight constraints

# Part 1 : Program Understanding as CSP

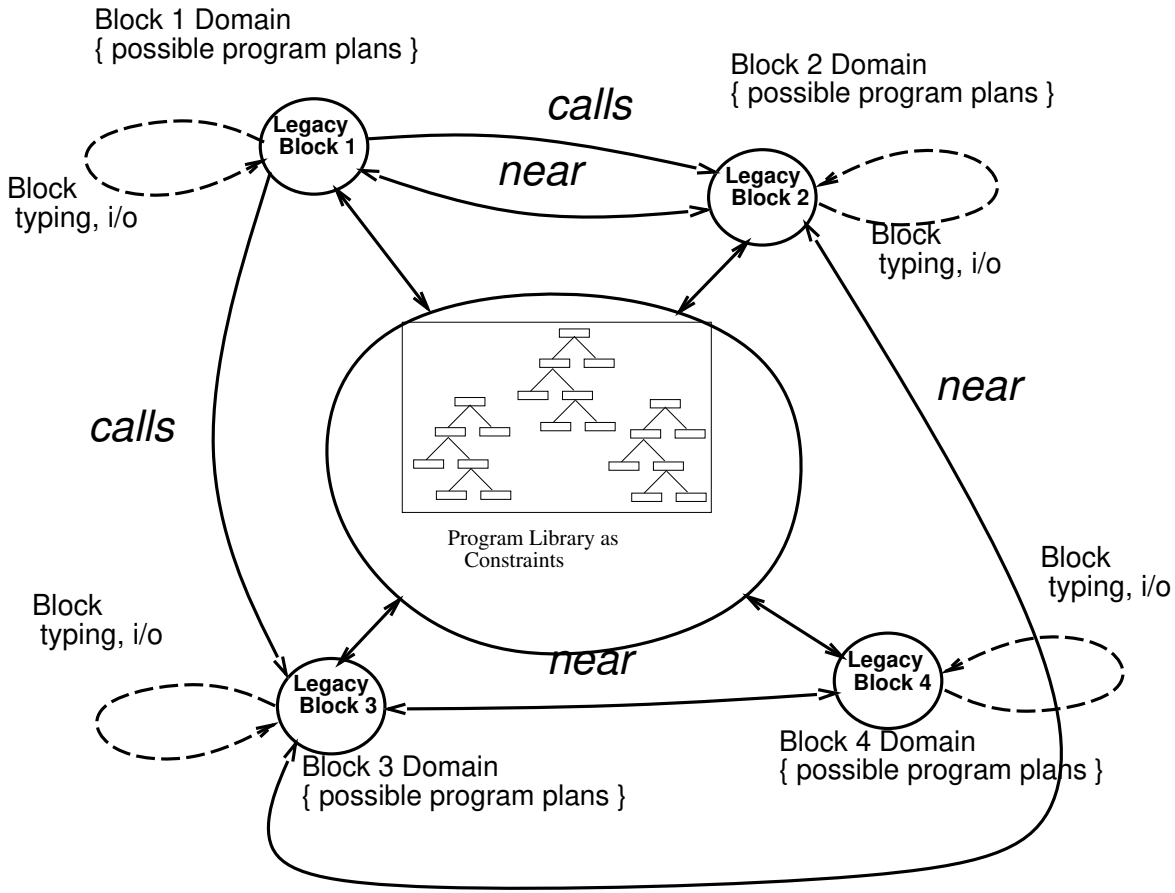



Figure 3: Mapping *Sliced* Legacy Code to program plans.

# Part 2 : Plan Template/Cliché Matching as CSP

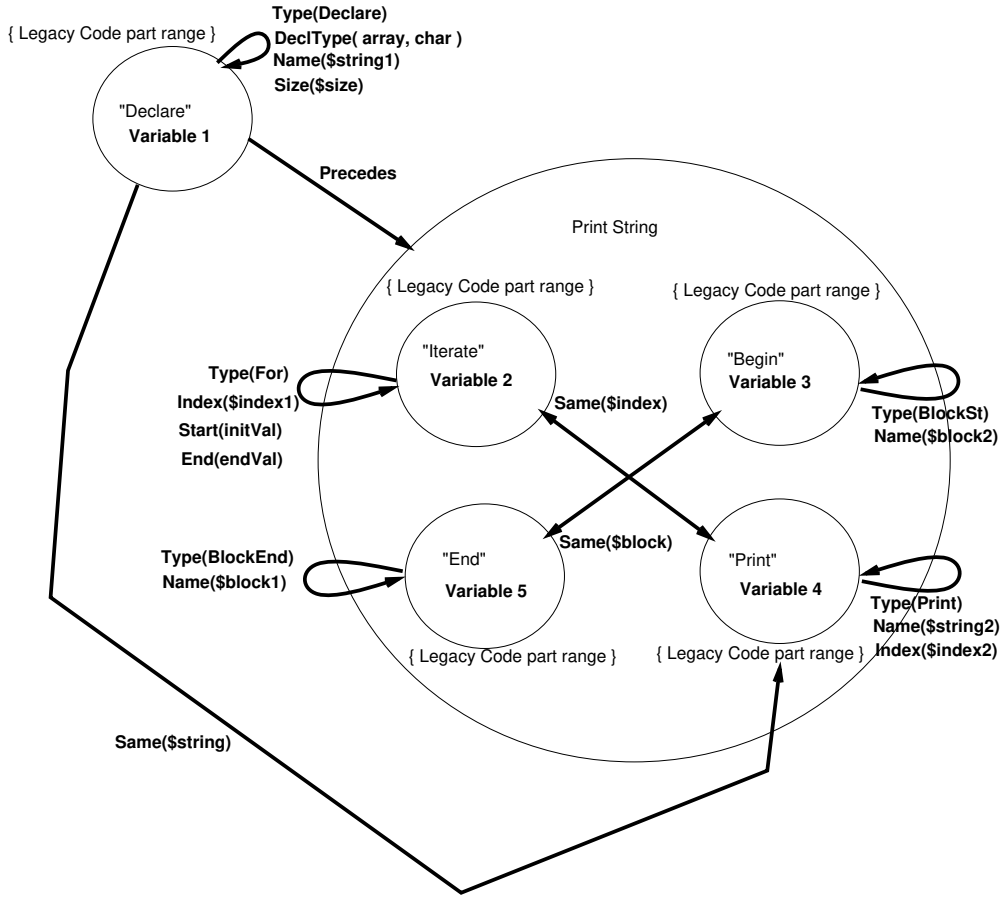



Figure 4: Partial String ADT CSP (From SLIDE 6).

## Related Efforts

- Quilici (1994) Memory-based recognition
  - $\longrightarrow$ No unified model, heuristic tricks only
  - $\longrightarrow$ No empirical results
- Kozaczynski & Ning (1994) Concept recognition
  - $\longrightarrow$ No unified model, primarily controlled top-down
  - $\longrightarrow$ No empirical results
- Wills (1992) Cliché concept recognition
  - $\longrightarrow$ Awkward representation of clichés (grammars)
  - $\longrightarrow$ Does not constrain search with knowledge

# Empirical Results : Standard Backtracking

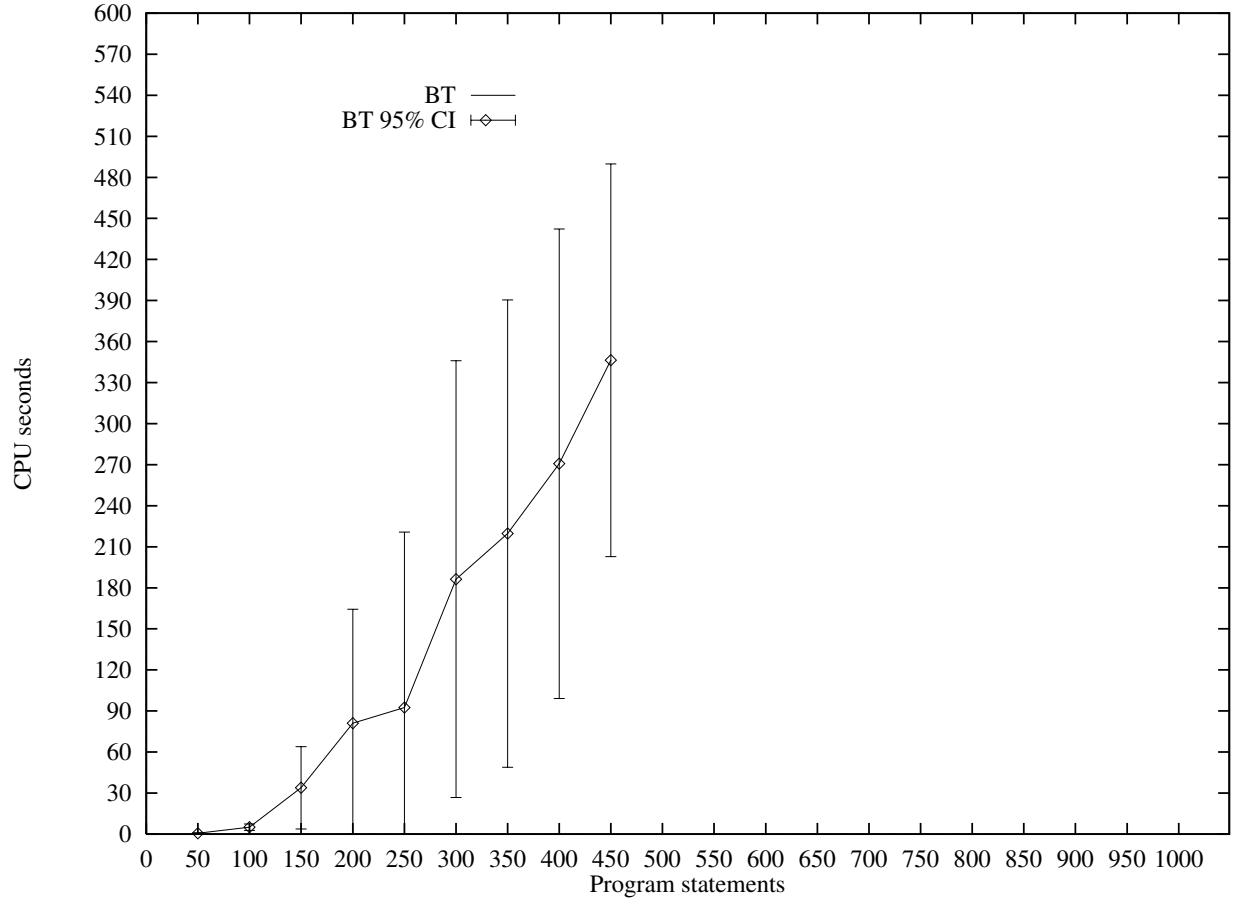



Figure 5: Standard Backtracking (cpu seconds).

# Empirical Results : Heuristic Strategy

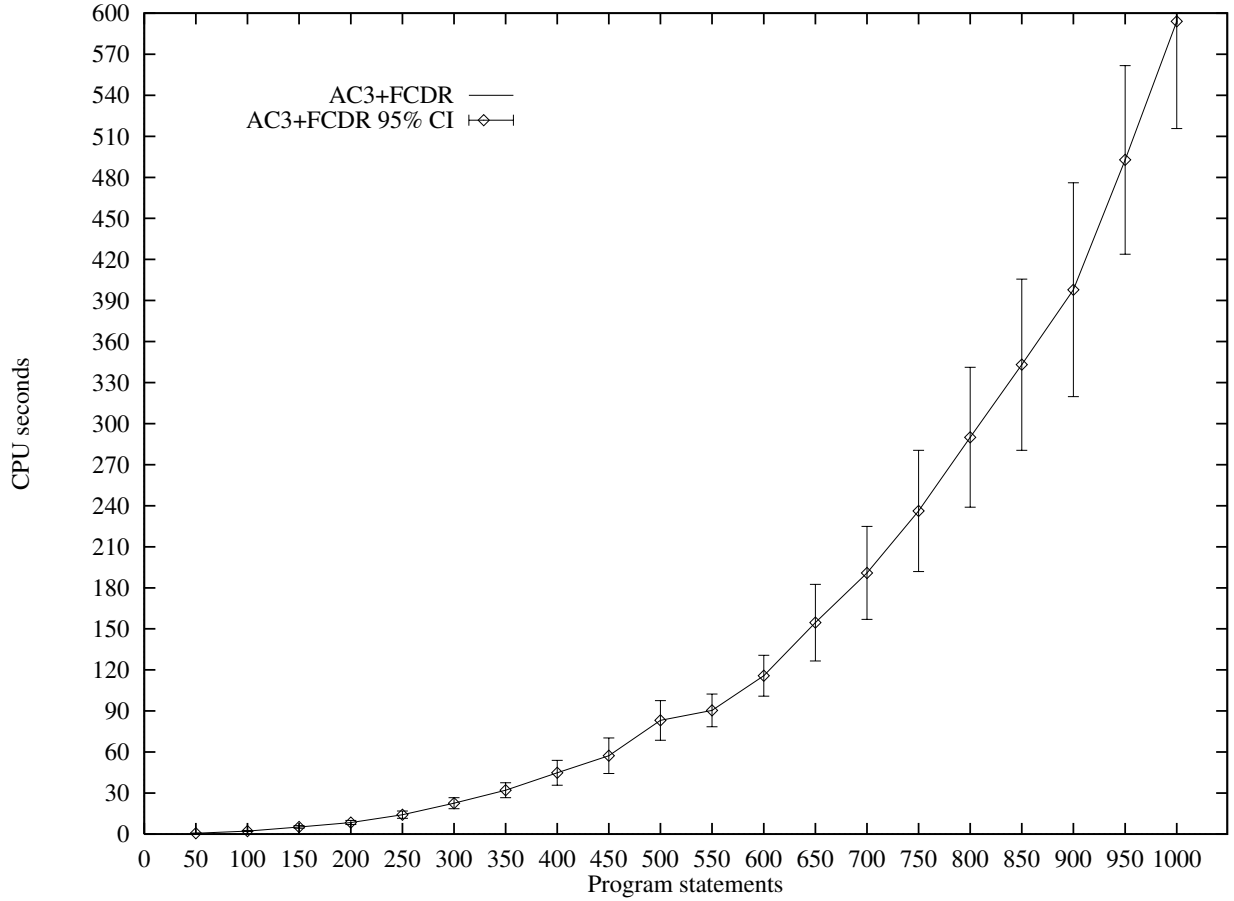



Figure 6: AC-3 with FC, DR (cpu seconds).

# Empirical Results : Backtrack vs Heuristic

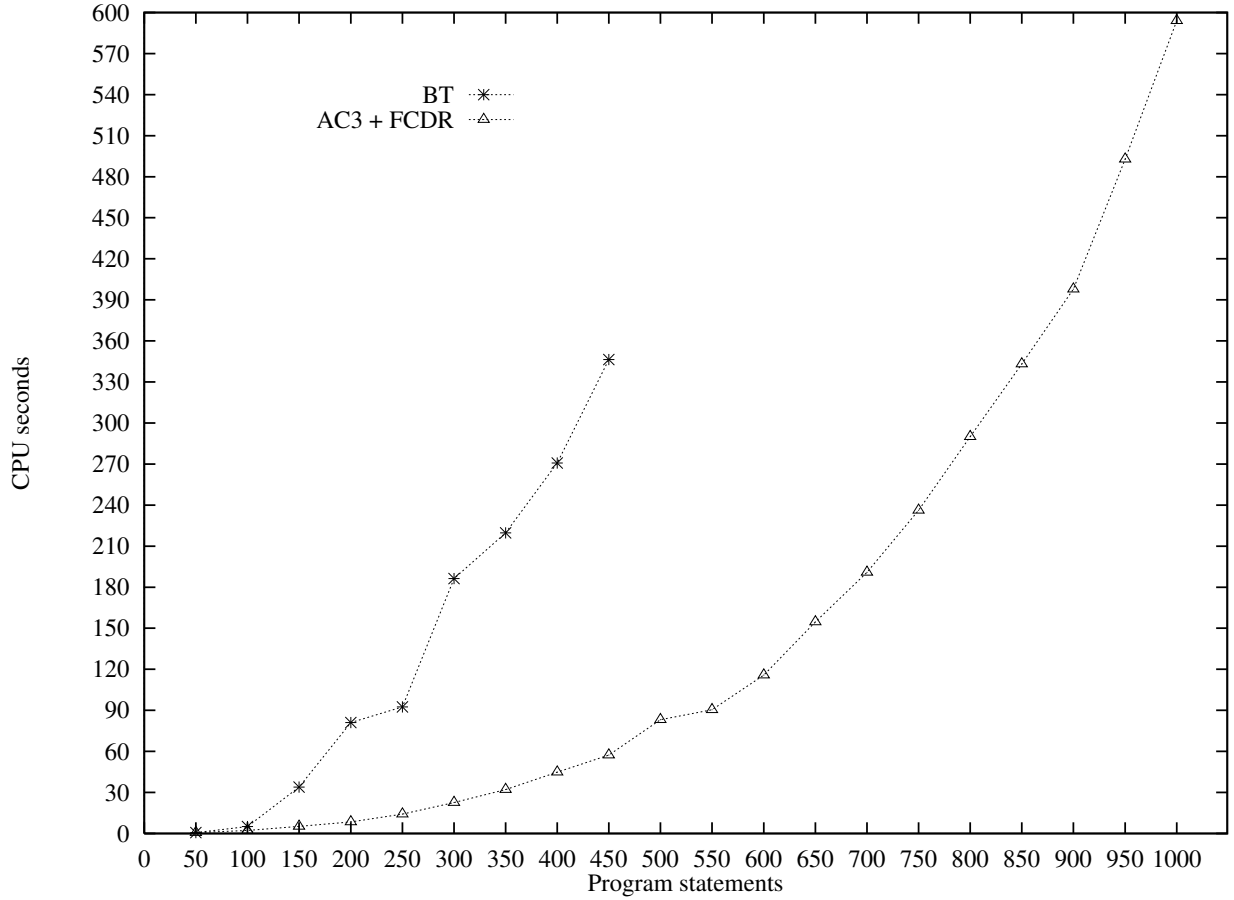



Figure 7: Standard BT vs AC-3 with FC, DR (cpu seconds).

# Empirical Results : BackTrack Quadratic

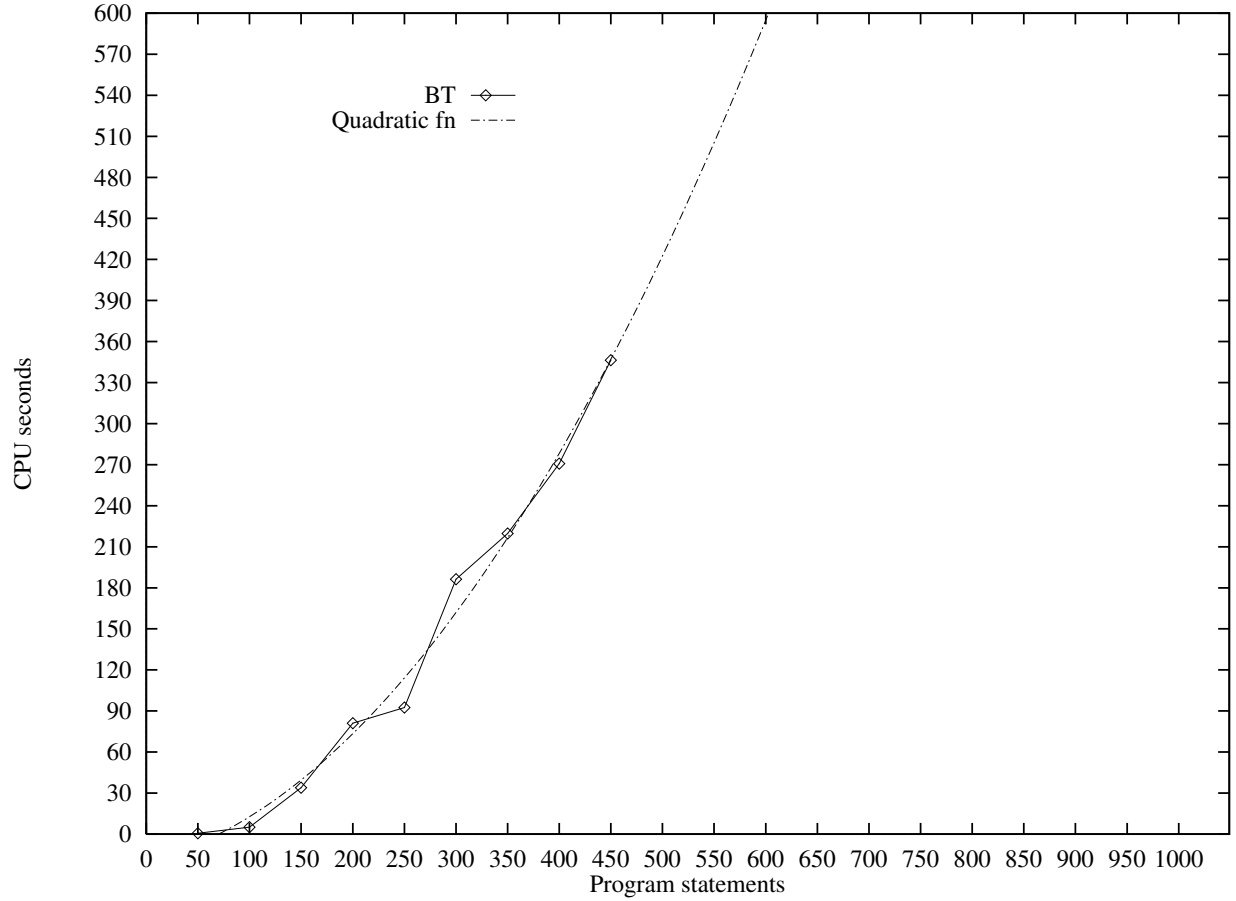



Figure 8: Quadratic fit to Standard BT Data.

# Empirical Results : Heuristic Quadratic

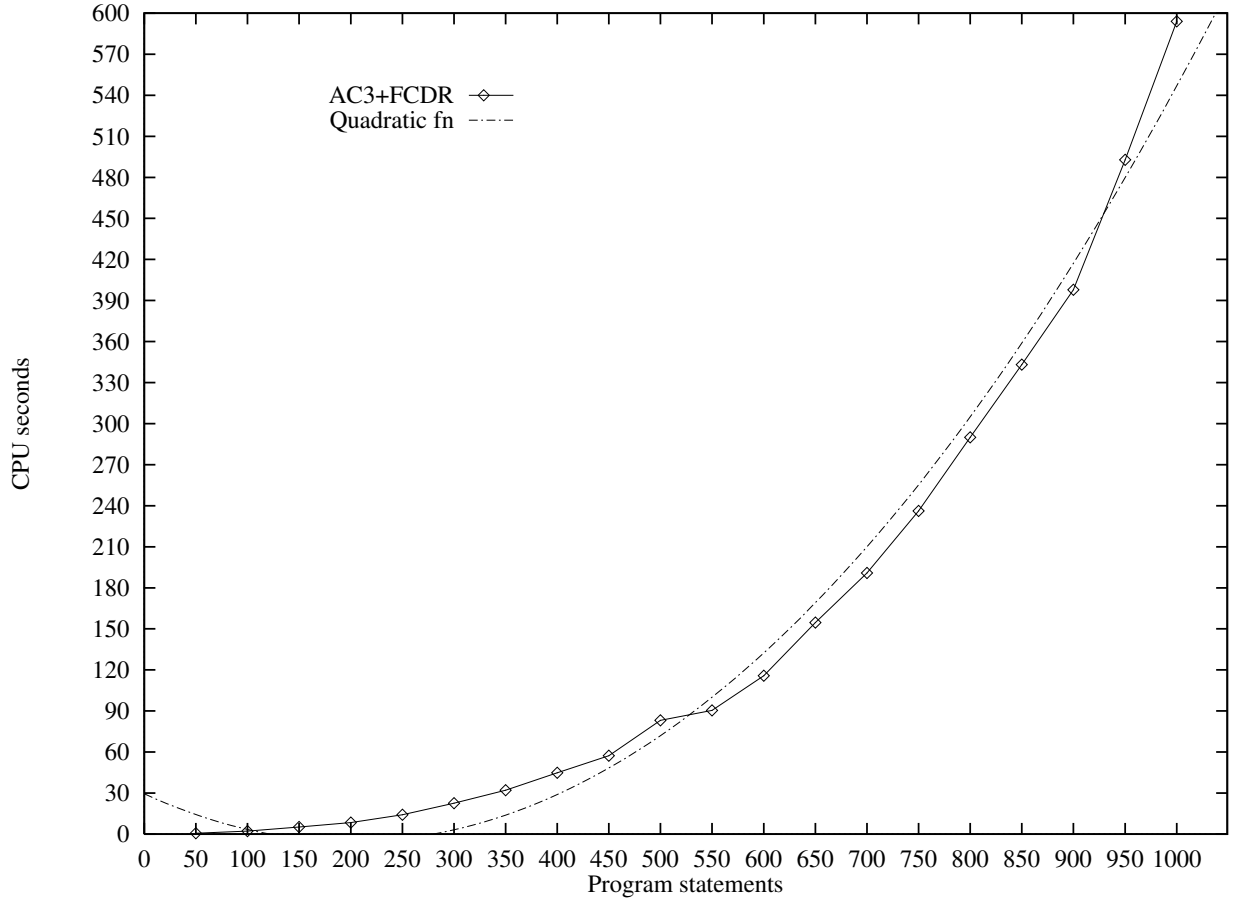
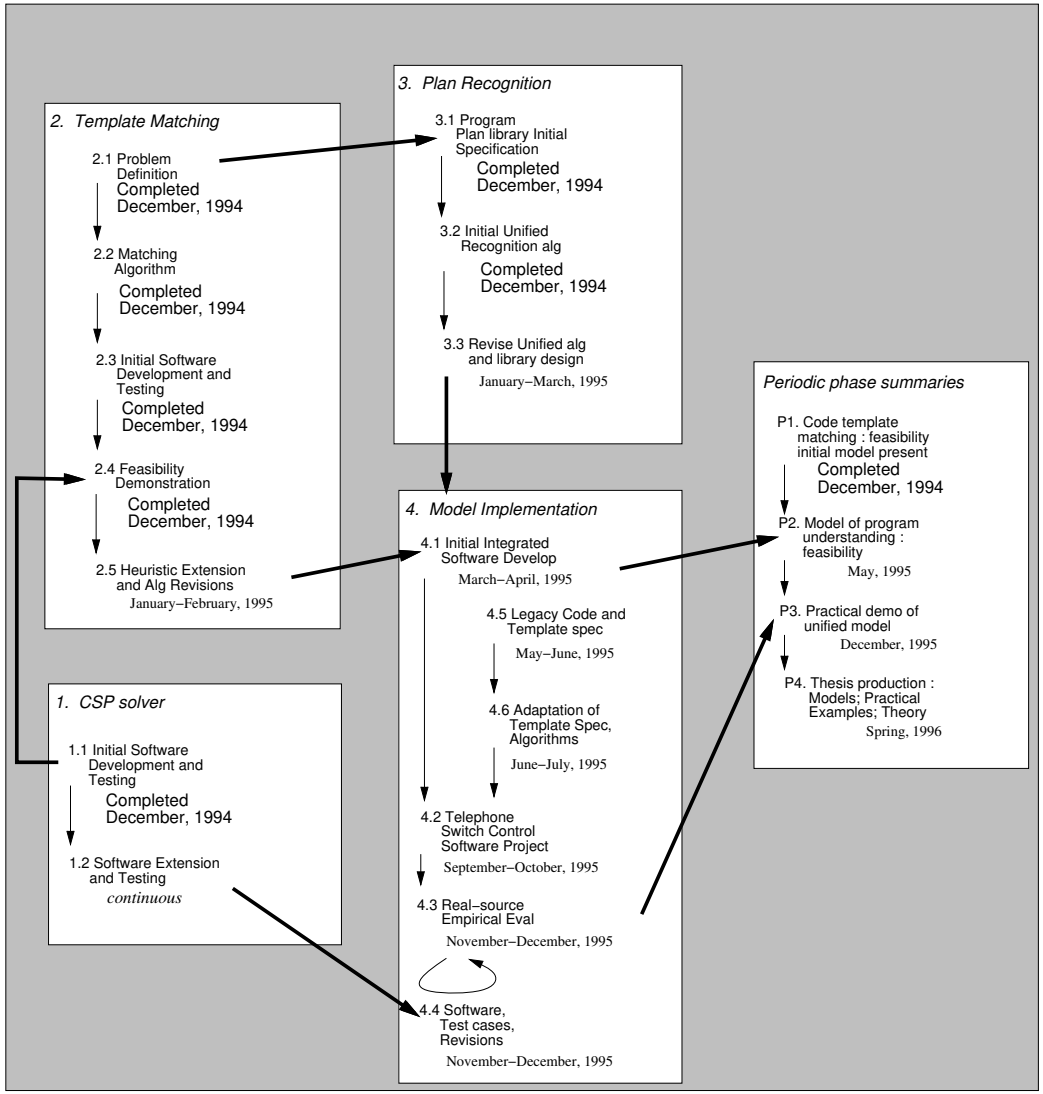



Figure 9: Quadratic fit to AC-3 with FC, DR.

# Research Timeline



## Major Milestones

- **[May, 95]** Represent program plan *knowledge* & legacy source *structure* as constraints in **Understanding CSP**.
- **[May, 95]** Combine **Matching CSP** and **Understanding CSP** $\longrightarrow$ integrated model of understanding including *Plan Under.* and *Rev. Eng.* heuristics
- **[Oct, 95]** Formulate partial library from industrial abstracted plans (BNR cooperation)
- **[Dec, 95]** Empirically demonstrate (partial) understanding of industrial source using **Understanding CSP**
- **[Dec, 95]** Identify knowledge and constraints req'd to give heuristic adequacy in interactive and batch instances