Proposal Outline: Acquiring Domain-Specific Planners from Example Plans

June 11, 2002

1. Introduction
   (a) Motivation
      i. Given plans that demonstrate how to solve a domain, learn program
      ii. (Agent modelling)
      iii. G-P planning: solvability horizon, speed
      iv. Learned control too big, therefore too slow (utility prob)
      v. Handwritten control must be... hand written!
      vi. Can often get "free" observed solutions, at least for simple probs
      vii. Use to learn d-s planner–they're faster
      viii. Discuss sound, complete, optimal tradeoffs, etc
   (b) Approach
      i. Use Observed Problems/Solutions as Source for Learning
      ii. Analyze Examples
      iii. Extract "Program" (/Template/Domain-Specific Planner) from Examples
      iv. Use D-S Planner to Solve New Problems
   (c) Expected Contributions
      i. Domain-independent algorithm for extracting rationale from example plans
      ii. Algorithm for learning domain-specific planners from example plans
2. Background
   (a) Plan Analysis
      i. Triangle Tables
      ii. Validation Structures
      iii. Derivational Analogy
      iv. Partially Ordering Total Orders
      v. Partial-Order Planning
   (b) Domain Knowledge for Planning
      i. Control Rules
      ii. Macro-Operators
      iii. Skeletal Plans & Abstraction Hierarchies
      iv. Case-Based & Analogical Planning
      v. Domain-Specific Planners
   (c) Program Learning
      i. Iterative and Recursive Macro-Operators

        ii. Program Learning from Specifications
        iii. Inductive Program Synthesis

3. Previous Work

    (a) Sprawl

    (b) Distill

4. Proposed Work

    (a) Definitely Will Do
        i. Discuss sound, complete, optimal tradeoffs, etc
        ii. Algorithm to acquire rich templates from example plans
            A. Extract Loops from Nice Examples
            B. Extracting Loops from Slightly-Less-Nice Examples (simplification)
            C. Merging Templates with Loops
        iii. Use and maintenance of the programs
            A. Current method
            B. Repairing Incorrect Templates
            C. Active Learning

    (b) Will Also Dabble In
        i. Agent Modelling (undersimplifying)
        ii. Examples of domains that can be covered
        iii. When template doesn't solve whole problem, can we use parts?

    (c) Evaluation
        i. Test on existing domains + more
            A. Generate domains with choices to demonstrate modelling
            B. Look for other domains for which it'd be good?
        ii. Test for:
            A. Possible to cover whole domain?
            B. Number of examples to cover whole domain
            C. Restrictions on examples for learning
            D. Ability to learn program from examples produced by the program
            E. Efficacy of learning on examples produced by various planners
            F. Efficacy for agent modelling (Ability to capture *preferences*)
            G. Speed vs generative planning
            H. Max problem size handled vs generative planning
            I. Efficiency of plans vs generative planning