CS294 Tentative Schedule

Tuesday, August 14, 2012 8:41 AM

Introduction. What is program synthesis? What is synthesis with constraint solvers? Constraint solver as program evaluator and invertor. Why synthesis now? Example synthesizers. What artifacts might be synthesizable? Course format. Overview of course project.	HW1: suggest uses for synthesis in your setting
Constraint solvers can help you write programs. Four programming problems solvable when a program is translated into a logical constraint: verification, fault-localization, angelic programming, and synthesis. Example of a program encoding with an SMT formula (Experimenting with Z3).	
Introduction to Racket (the new face of Scheme). Encode a more realistic problem with a Racket formula generator. **Project 0** Project format Problem suggestions Open problems how to synthesize Challenge problems what to synthesize). Review of HW1.	
Satisfiability solvers. Solver diagnostics. Additional solvers and their logics. Example of several encodings for the SIMD matrix transpose problem.	
Encoding program constructs with logical formulas. Arrays. Loops. Heaps. Single-static assignment. (Manually translate a SIMD matrix transpose into relational logic.) Tutorial on the Kodkod solver.	HW2: encode a programming problem as a logical formula
Why are small languages useful. Increase the level of programming abstraction. Language properties desired in general and for synthesis in particular. Examples of small languages (geometry construction, relational data structures, automata protocols).	
Project 1: The problem statement (what we want to synthesize). Students present their project proposals.	
Language implementation I: Introduction to the Racket meta-programming constructs (macros).	
Show and tell: lessons learnt in HW2. **Synthesis vs. constraint solving.** Angelic programming. Interfaces to solvers (Kaplan). Constraint Programming?	
Language implementation II: Shallow embedding. Deep embedding. Compilation. Open problem: how to define the language once and obtain an interpreter as well as formula translator.	may overflow to previous lecture
Project 2: what's our DSL (how to say it)	
Specifications. Functional vs. structural specs. Executable vs. declarative specs. Demonstrations and trace assertions. Test cases. Stepwise refinement.	Leino on Specifications
Case studies on specifications. Inversion of image format normalizer. Refinement (TBD)	
Automatic translation of programs to formulas. Synthesizer compilation strategies for finitization and partial evaluation. Static analysis in a constraints-based compiler. (Case studies: CBMC, Sketch, Rosette, Cell).	
Project 3: What's our specs (what to say). Spec clinic.	
Synthesis algorithms. Counterexample-guided inductive synthesis (CEGIS). Synthesizer that searches for input witnesses ambiguity in the specification. (Case study: CEGIS implemented in Rosette.)	HW3: Implement Disambiguating Solver Rosette
Synthesis of concurrent and distributed programs. Encoding thread interleaving. Specifications for concurrent programs. 3QBF synthesis.	
Project 4: My encoding (how to encode our programs as constraint system).	
Synthesis with version space algebra. Programming by demonstration. (SmartEdit, QuickCode).	
Rewrite synthesizers. (Spiral, Denali, AutoBayes)	
Backtracking and exhaustive exploration. Strategies for pruning search (Skalch and Geometry Construction)	
Project 5: Synthesis algorithm used in my project (search of program space as constraint solving).	
Programming domain knowledge: Revealing redundant constraints to the synthesizer. Encoding a problem domain with a partial program (dynamic programming).	
Scalability and reuse. Partial programs as grammars. Reusable partial programs. Symmetry reduction in partial programs.	
Tricks: Using angelic non-determinsim (constraint solving) in a language interpreter (to schedule dataflow programs). Reduce formula size by avoiding subformula duplication.	up
Project 6: scaling up with domain knowledge (reveal to solver additional knowledge)	
Synthesis-based compilers. Synthesis at abstract semantic level (FTL, L3 synthesizers).	

<u>Project 7</u>: project presentations (demos and timeless lessons)	