Advanced C Programming

Exam, Competition, Code Review

Sebastian Hack
hack@cs.uni-sb.de

Christoph Weidenbach
weidenbach@mpi-inf.mpg.de

Winter Term 2008/09

Final Exam

Assignemnts

1. Declarations, Bindings, ... (multiple choice)
2. Find Errors in Code Fragments
3. Improve Code Fragments
4. Profiling, Makefiles, ...
5. Bit Operations
6. Know the Compiler
7. Code/Data Structure Design (possibly related to SAT)

Out of Scope

1. Open MP
2. Syntax Checks

Advanced C SAT Competition

Problems

1. 2bitmax_6.cnf, 252 vars, 767 clauses, satisfiable
2. qg2-08.cnf, 512 vars, 148957 clauses, satisfiable
3. qg3-09.cnf, 729 vars, 16732 clauses, satisfiable
4. uf250-01.cnf, 250 vars, 1065 clauses, satisfiable
5. uf250-013.cnf, 250 vars, 1065 clauses, satisfiable
6. uuf250-01.cnf, 250 vars, 1065 clauses, unsatisfiable
7. uuf250-013.cnf, 250 vars, 1065 clauses, unsatisfiable

Out of Scope

- Medium hard unsatisfiable problems.
- Competition problems from 2008.

Advanced C SAT Competition: Setup

Parameters

- 300 sec per problem
- 41 SAT programs entered
- all programs compiled with -O3
- most recent version taken
- hardware: 3.16GHz Xeon, 6MB Cache, 16GB RAM, 4 CPUs, 2 Jobs

Advanced C SAT Competition: Results

Statistics

- 16 programs crashed on at least one example
- 3 programs produced wrong results
- 29 programs could not solve any problem
- 4 programs solved one problem
- 2 programs solved two problems
- 2 programs solved three problems
- 1 program solved four problems
- 1 program solved five problems
- 1 program solved six problems
- 1 program solved seven problems

Advanced C SAT Competition: Comparison

Timing

Program	2bitmax	qg2	qg3	uf-1	uf-13	uuf-1	uuf-13
SAT	44.83	126.10	3.95	17.07	32.23	88.62	45.71
PROP	7.14	2.37	14.71	37.99	tout	tout	tout
Mini	0.00	3.23	2.99	0.05	1.71	1.99	2.11

Merging Replacement Resolution

Tricks

1. link complementary literals
2. consider clause length
3. sort literals
4. do fingerprint of first n-atoms

Example1: Queues

```
typedef struct QUEUE_HELP
{
  int first;     /* first element */
  int last;      /* if last = -1 the queue is empty */
  int step;      /* amount to grow */
  int size;      /* current array size */
  void** queue; /* cyclic array of void * */
}
QUEUE_NODE;

typedef QUEUE_NODE *QUEUE;
```

Queues Continued

```
void*  queue_Get(QUEUE q)
/*********************************************
INPUT:   A queue.
RETURNS: The first element is removed from
         the queue a returned.
NOTE:    Should only be called on a nonempty queue.
*********************************************/
{
  void* res = q->queue[q->first];

  ASSERT(!queue_IsEmpty(q));

  if (q->first == q->last) { /*last element*/
    q->last = -1;            /*we are empty */
  } else {
    q->first = (q->first+1) % q->size;
  }

  return res;
}
```

Example2: DPLL

```
int solver_Solve(SOLVER sol)
{ int unit_literal;
  if (sol->contradiction) {
    sol->contradiction = 0;
    return 0;}
  if (clause_AllTrue(sol->set)) return 1;
  else if (clause_SomeFalse(sol->set)) return 0;
  else {
    unit_literal = clause_FindUnitLiteral(sol->set);
    if (unit_literal) {
        solver_Decide(sol, unit_literal);
        return solver_Solve(sol);
    } else { int var, pos;
        var = solver_UndefinedVar(sol);
        pos = solver_Decide(sol, -var);
        if (solver_Solve(sol)) return 1;
        else {
            solver_Backtrack(sol, pos);
            solver_Decide(sol, var);
            return solver_Solve(sol);
    }}}
   return 0;
 }
```