HACMS kickoff meeting: TA2

Technical Area 2: System Software

John Rushby

Computer Science Laboratory
SRI International
Menlo Park, CA

Introduction

- We are teamed with Prof. Grigore Rosu of University of Illinois at Urbana Champaign on this task
- I'll describe our part
- Then hand over to Grigore

Background

- All incidents and accidents in commercial aircraft in which software was a contributory factor implicate the gap between system requirements and software requirements
- None implicate design or coding errors
- Level A software for commercial aircraft costs a lot
- Vulnerabilities in other kinds of vehicles may be different
- FM may reduce costs for aircraft and raise quality elsewehere
- But the gap may still be there
- That's what we (SRI) are focused on

A Conundrum

- Top-level safety requirements are probabilistic (e.g., 10^{-9})
- But software assurance is all about correctness
- JUst do more of it for higher assurance levels
  - 28 objectives at DO178B Level D (10^{-3})
  - 57 objectives at DO178B Level C (10^{-5})
  - 65 objectives at DO178B Level B (10^{-7})
  - 66 objectives at DO178B Level A (10^{-9})
- What's the connection?

A Simple Theorem

- Software assurance establishes a possibility of perfection
  - Will never suffer a failure, wrt. system requirements
- Quantify that as (subjective) probability of (im)perfection
  - An idea due to Bev Littlewood and Lorenzo Strigini
- p_{np} probability the software is imperfect
- p_{fnp} probability that it fails, if it is imperfect
- Then $P(\text{software fails}) \leq p_{np} \times p_{fnp}$
- Traditionally, nuclear protection assumes p_{np} is 1, measures p_{fnp} by massive random testing
- And aircraft certification assumes p_{fnp} is 1, try to justify small p_{np} by massive assurance

A Second Theorem

- Many safety-critical systems have two (or more) diverse "channels" arranged as primary/monitor architectures
- Cannot simply multiply the pfds (probabilities of failure) of the two channels to get pfd for the system
  - Failures are unlikely to be independent
  - E.g., failure of one channel suggests this is a difficult case, so failure of the other is more likely
  - Infeasible to measure amount of dependence
- But the probability of imperfection of one channel is conditionally independent of the pfd of the other
- So you can multiply these together to get system pfd

Putting It Together

- Formally synthesize or verify monitors for system requirements

- Monitors can be simple, as well as formally assured

- Thus, feasible to claim small probability of imperfection

- Hence, multiplicative increase in system reliability

- Though you do need to account for Type 2 monitor failures

- Monitored architecture risk per unit time

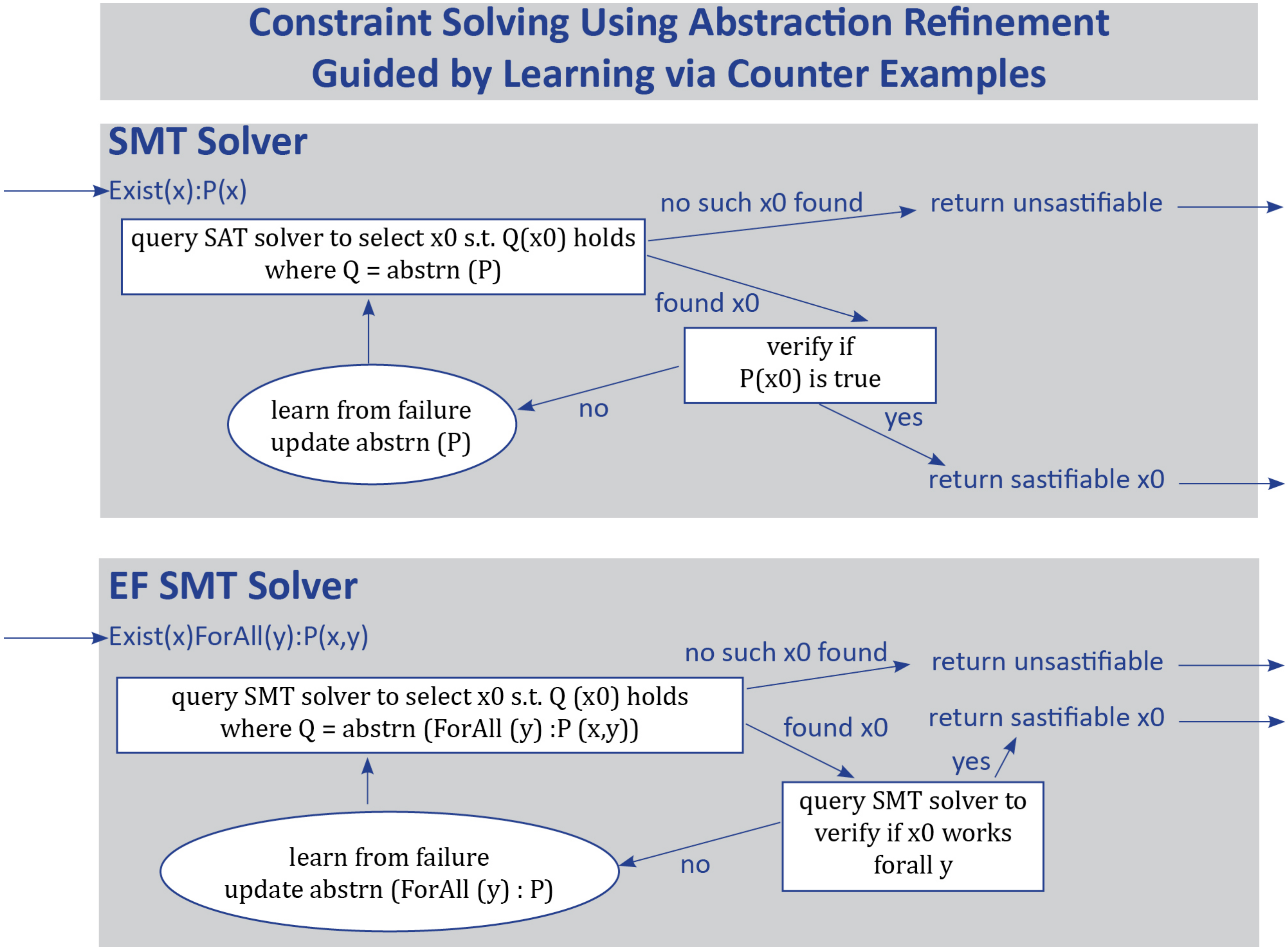
  $$\leq c_1 \times (M_1 + F_A \times P_{B1}) + c_2 \times (M_2 + F_{B2|np} \times P_{B2})$$

  where the Ms are due to mechanism shared between channels

Mechanization

- Biggest breakthrough in FM over last 20 years was development of high-performance SMT solvers
- These solve Forall (UNSAT) and Exists (SAT) problems
- They automate verification problems very effectively
- But for synthesis need to solve Exists-Forall (EF) problems
- Example: template based invariant synthesis
    - $\exists A, B, C : \forall x, y : A \times x + B \times y < C$
    - Many template- or sketch-driven approaches to synthesis can be cast in this form
- So we plan to synthesize monitors with an EF-SMT solver

EF SMT Solver Architecture



Plan

- Develop EF-SMT solver
  - Bruno Dutertre

- Use to synthesize monitors and wrappers for systems software

- Share languages, methods, tools with Grigore Rosu of UIUC
  - Who develops complementary approaches to monitoring