
Foster


Hartmann


Lafortune


Kavraki


Kress-Gazit


Loo


Madhusudan


Bodik

ExCAPE

Expeditions in Computer Augmented Program Engineering


Martin


Alur

Expeditions in Computer Augmented Program Engineering

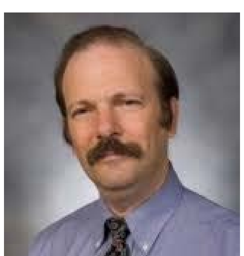
http://excape.cis.upenn.edu/


Pappas

Cornell, Maryland, Michigan, MIT, Penn, Rice, UC Berkeley, UCLA, UIUC

NSF Expeditions PI Meeting, May 2013


Zdancewic


Vardi


Tripakis

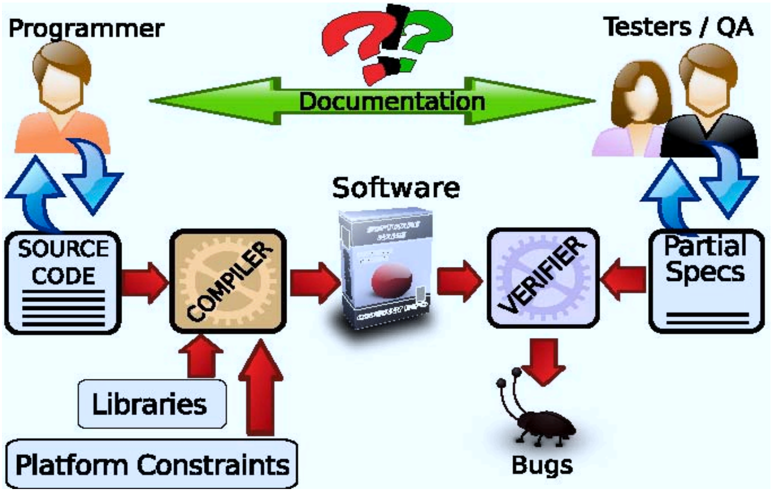

Tabuada


Solar-Lezama


Seshia


Sangiovanni

Software Design Methodology



- What has changed:
  - Programming languages
  - Libraries
  - Verification technology
- What has not changed:
  - Programming is done by experts
  - Fully specified by conventional programming
  - Verification phase is distinct from design

Can we leverage modern analysis tools and increased computing power to revolutionize the task of programming?

Synthesis: A Plausible Solution ?

- Classical: Mapping a high-level (e.g. logical) specification to an executable implementation
  - Theoretical foundations: Church (1960s)
  - Derivation of programs from constructive proofs (e.g. Kestrel)
  - Synthesis from temporal logic specs: Clarke/Emerson (1980s)
  - Refinement in model-based design
  - Ongoing progress, but many challenges remain…
- Recent shift in focus: Integrating different styles of specifications in a consistent executable

Sketch: Program completion

Ref: Solar-Lezama et al (PLDI 2010)

```
Err = 0.0;
for(t = 0; t<T; t+=dT){
  if(stage==STRAIGHT){
    if(t > ??) stage= INTURN;
  }
  if(stage==INTURN){
    car.ang = car.ang - ??;
    if(t > ??) stage= OUTTURN;
  }
  if(stage==OUTTURN){
    car.ang = car.ang + ??;
    if(t > ??) break;
  }
  simulate_car(car);
  Err += check_collision(car);
}
Err += check_destination(car);
```

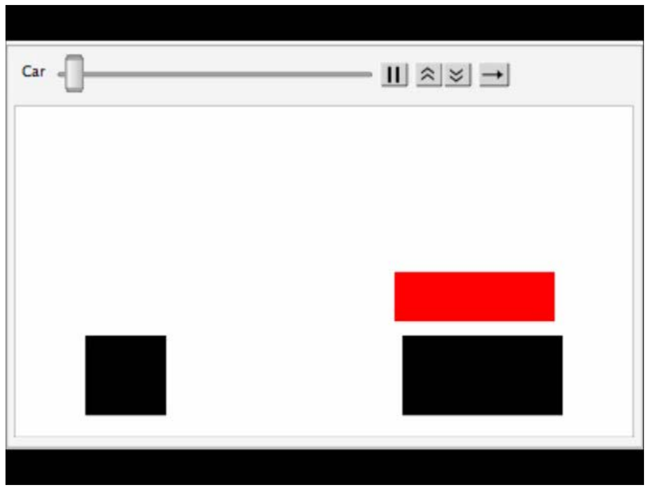
When to start turning?

Backup straight

How much to turn?

Turn

Straighten

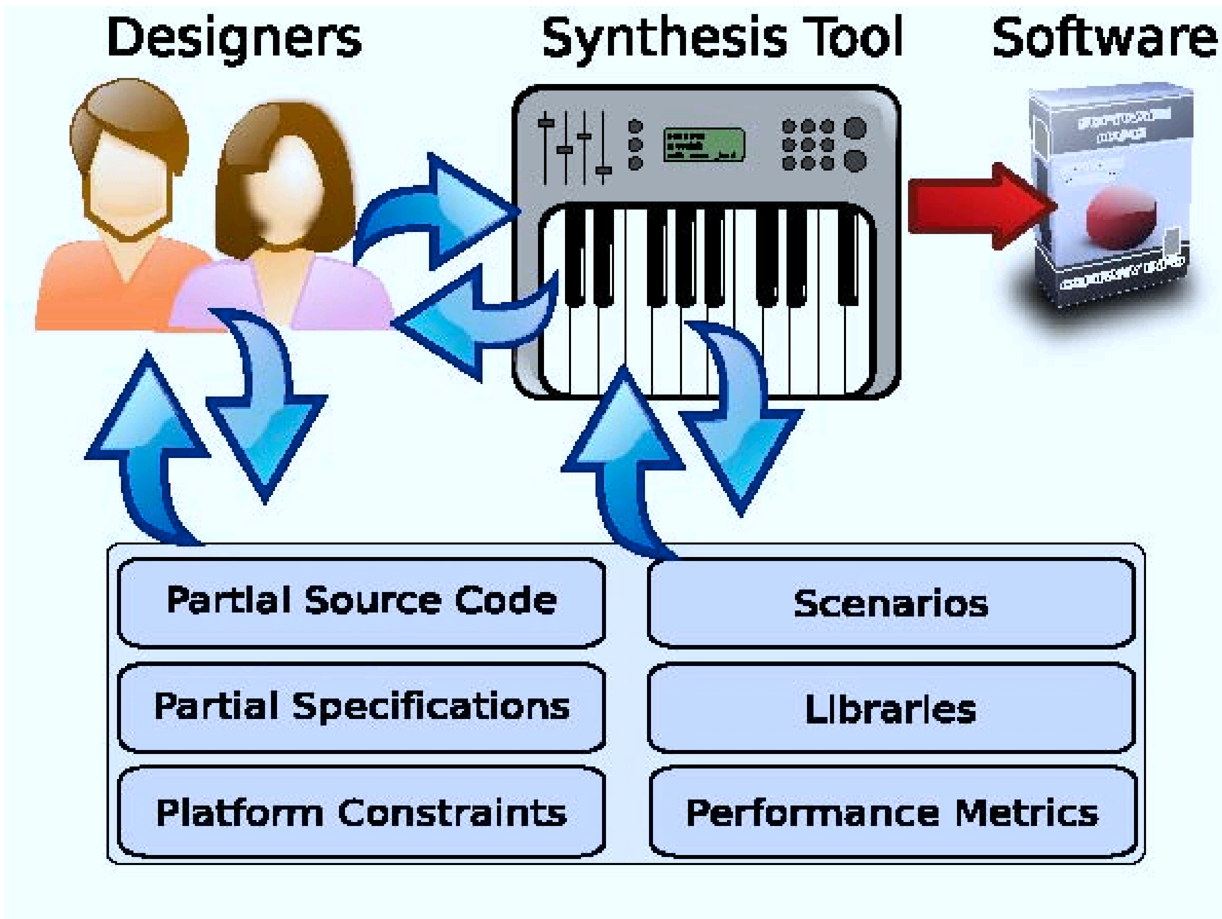


Enables programmers to focus on high-level solution strategy

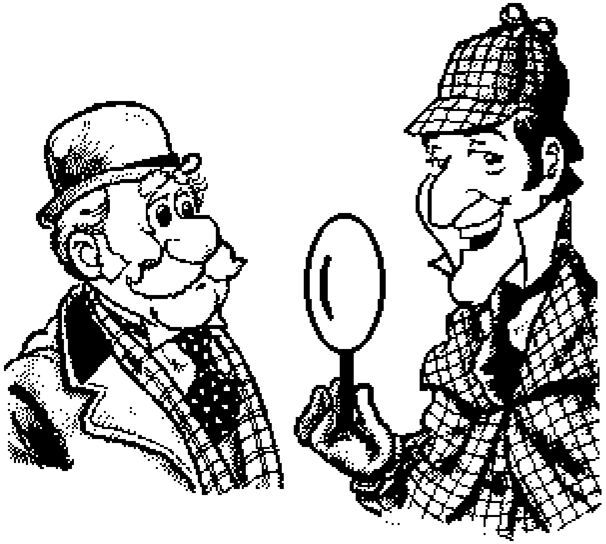
ExCAPE Vision

Harnessing computation to transform programming:

Programming made easier, faster, cheaper

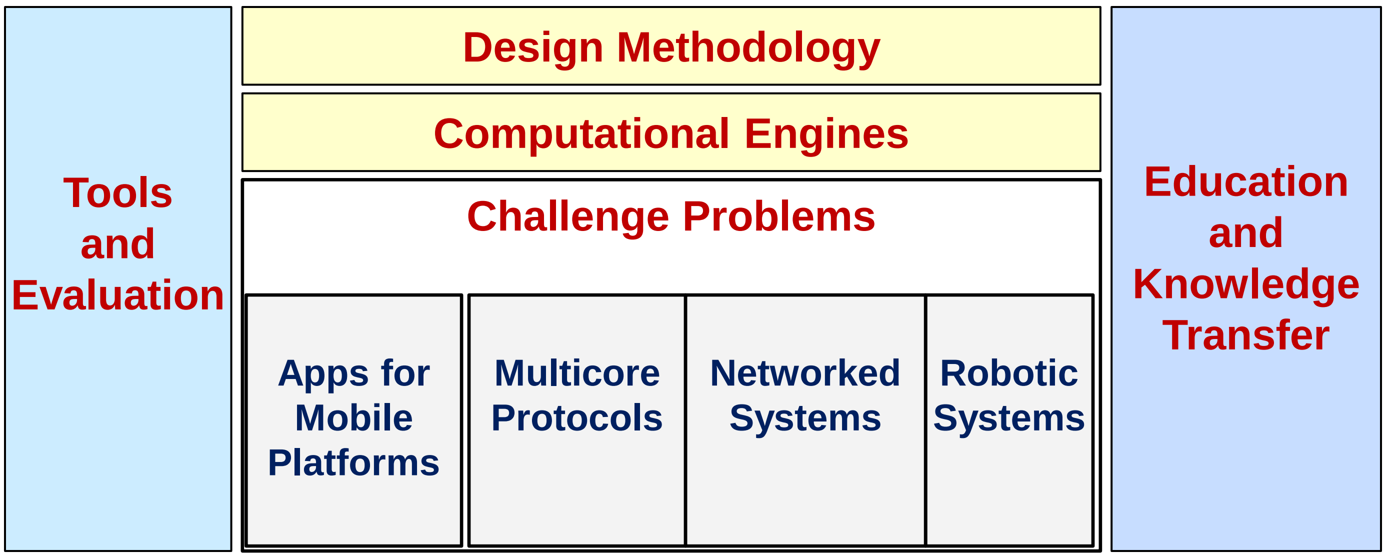


Synthesis Tool:
Intelligent Assistance



- Designer expresses "what", possibly using multiple input formats
- Synthesizer discovers new artifacts via integration and completion
- Synthesizer solves computationally demanding problems using advanced analysis tools
- Interactive iterative design
- Integrated formal verification

Research Organization



Talk Outline

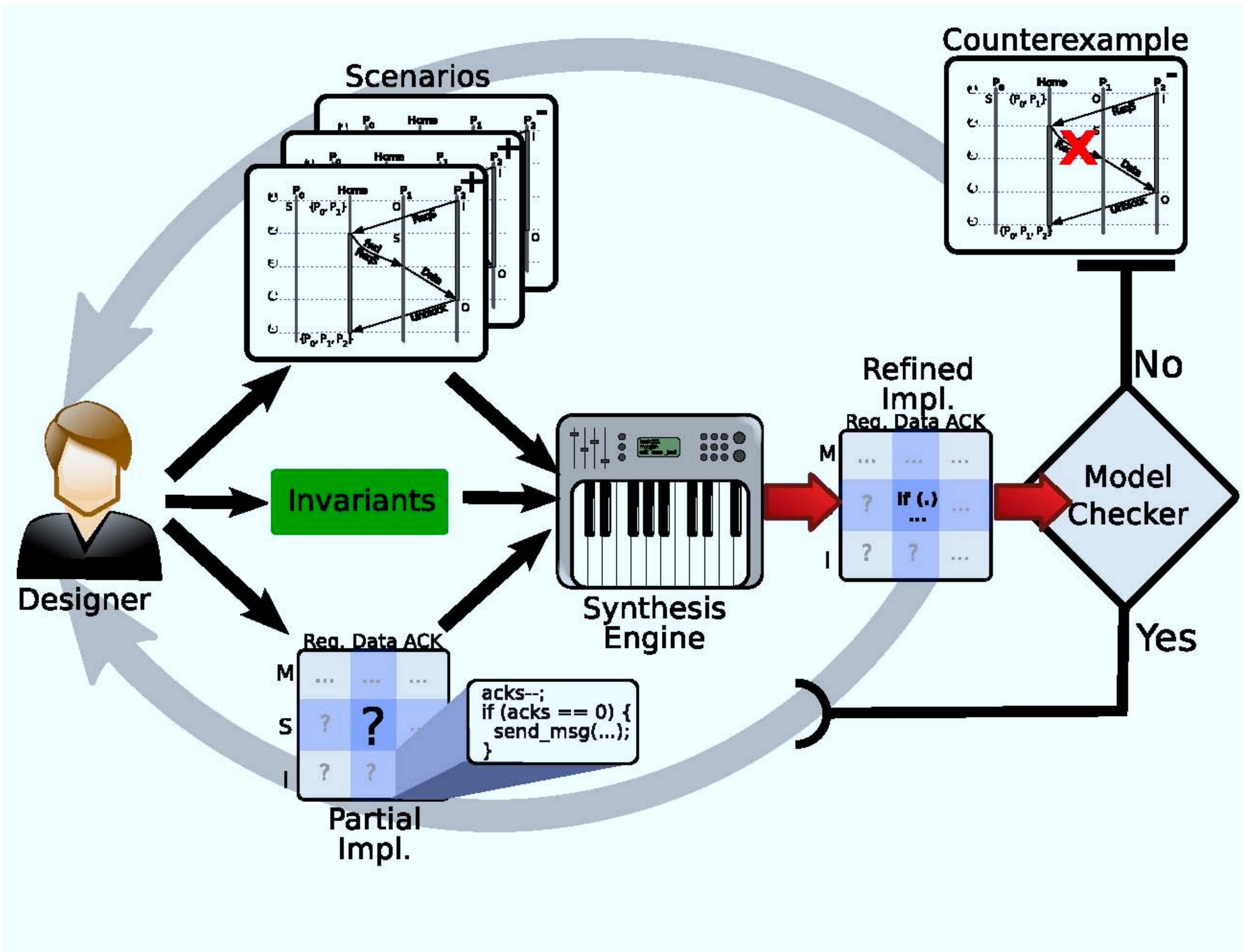
- Design tool for distributed protocols
- Synthesis for programming robots
- Synthesis to support online education
- Summary of ExCAPE activities

Goal: Simplify Protocol Design

- Design challenging due to asynchronous model of communication
- Cache coherence protocols, Distributed coordination algorithms
- Successful application domain for formal verification / model checking
- Correctness involves both safety and liveness properties
- Proposed solution: Allow programmers flexibility

Protocol = **Skeleton based on Extended-Finite-State-Machines**
+ High-level requirements
+ Example behaviors

TRANSIT for Distributed Protocol Design



Computational Problem

- Inputs:
  - Variable types and corresponding expression grammar
  - For each process,
    1. Control states of EFSM
    2. List of all variables , input/output messages
    3. Set of concrete examples + symbolic constraints
  - High-level requirements (invariants and temporal logic formulas)

- Solution strategy:
  - Expression Inference: For each EFSM transition, generate expressions for guards and updates. Solution uses Counter-example-guided-inductive synthesis using SMT solver Z3
  - Check if resulting protocol meets all requirements, using a model checker (Murphi) and if not, report a counter-example

Challenging Case Study: SGI Origin protocol

- Source: Laudon and Lenoski; The SGI Origin: A CCNUMA highly scalable server; ISCA 1997

- Directory-based MESI protocol that handles multiple concurrent requests to same requests over unordered network

- Textual description directly leads to protocol skeleton, and symbolic (incomplete) descriptions of most of the transitions

- During debugging, programmer focuses on local fixes of counteexamples and adds concrete examples

- Final iteration required 30 min synthesis time (with 5 Million states explored by Murphi)

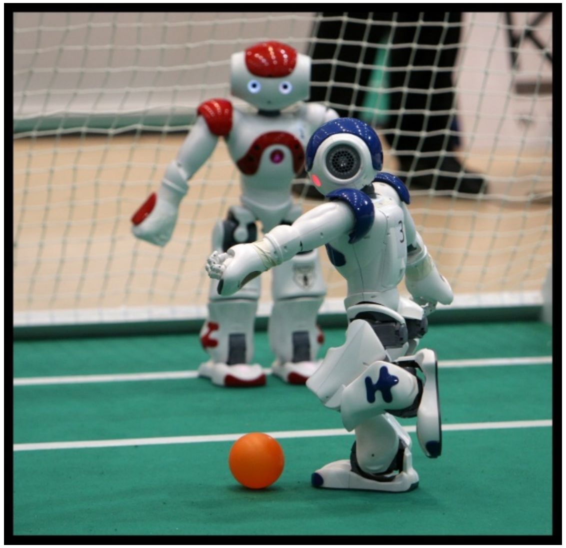
- SMT solver / model checker in the loop is feasible for programming

Synthesis for Robot Programming

Goal: Allow end-users to program robotic behaviors

Automatically

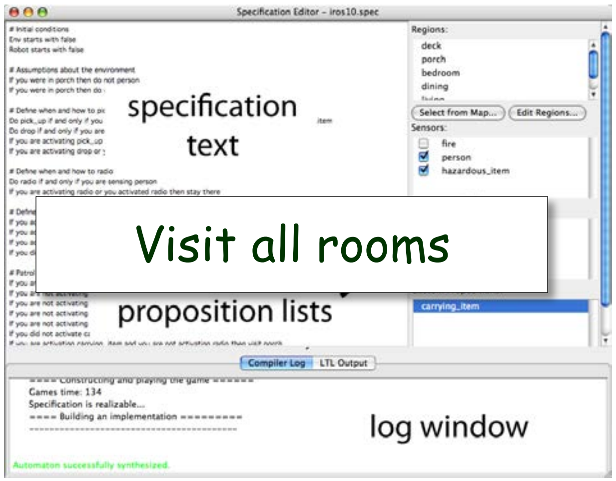
(Provably Correct)



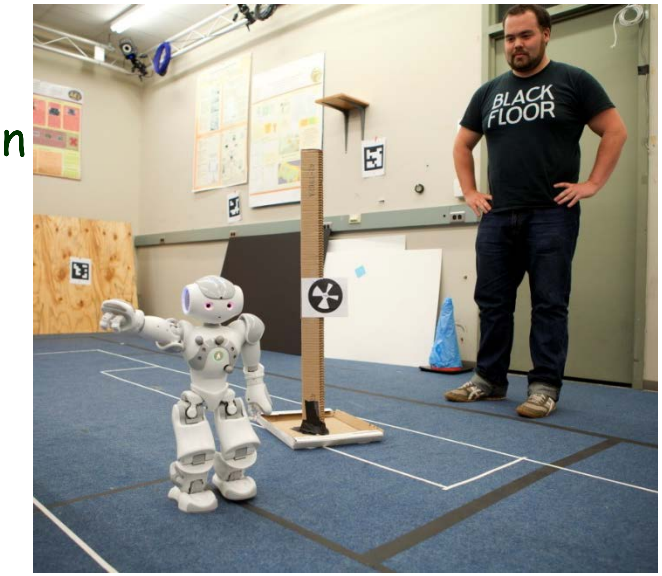
Robotic controllers: Research Challenges

- How to consistently integrate physical constraints, sample trajectories, safety rules, and language/temporal-logic requirements?
- How to explain infeasible requirements? How to suggest potential fixes?
- How to program a synthesis engine with completion strategies that take into account the physical and continuous nature of robotics (power, safety, environment traversability)?
- How to address optimality and performance?
- How to evaluate human-robot interaction?
- How to generate control that ports across different robots (different dynamics, control capabilities, safety considerations)?

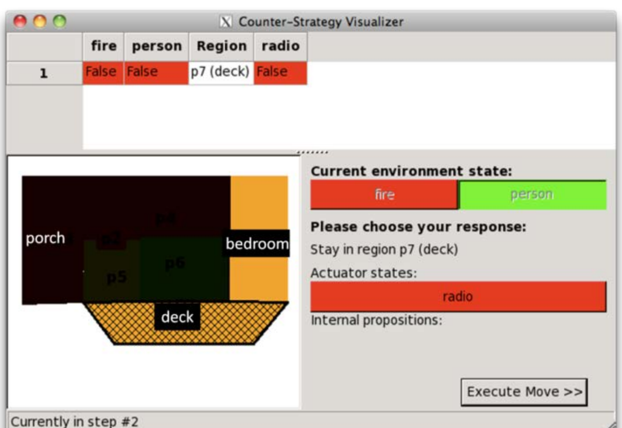
LTLMoP: Robot control from structured English

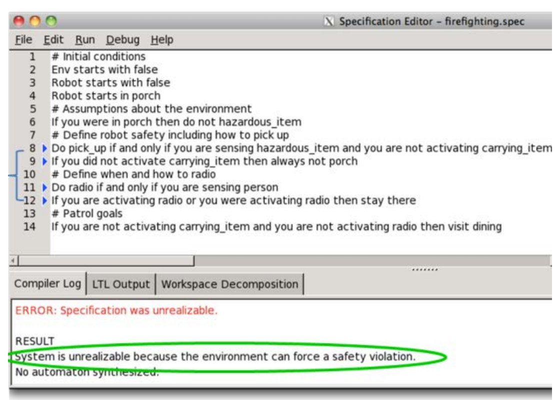


Feasible specification



Unsynthesizable specification



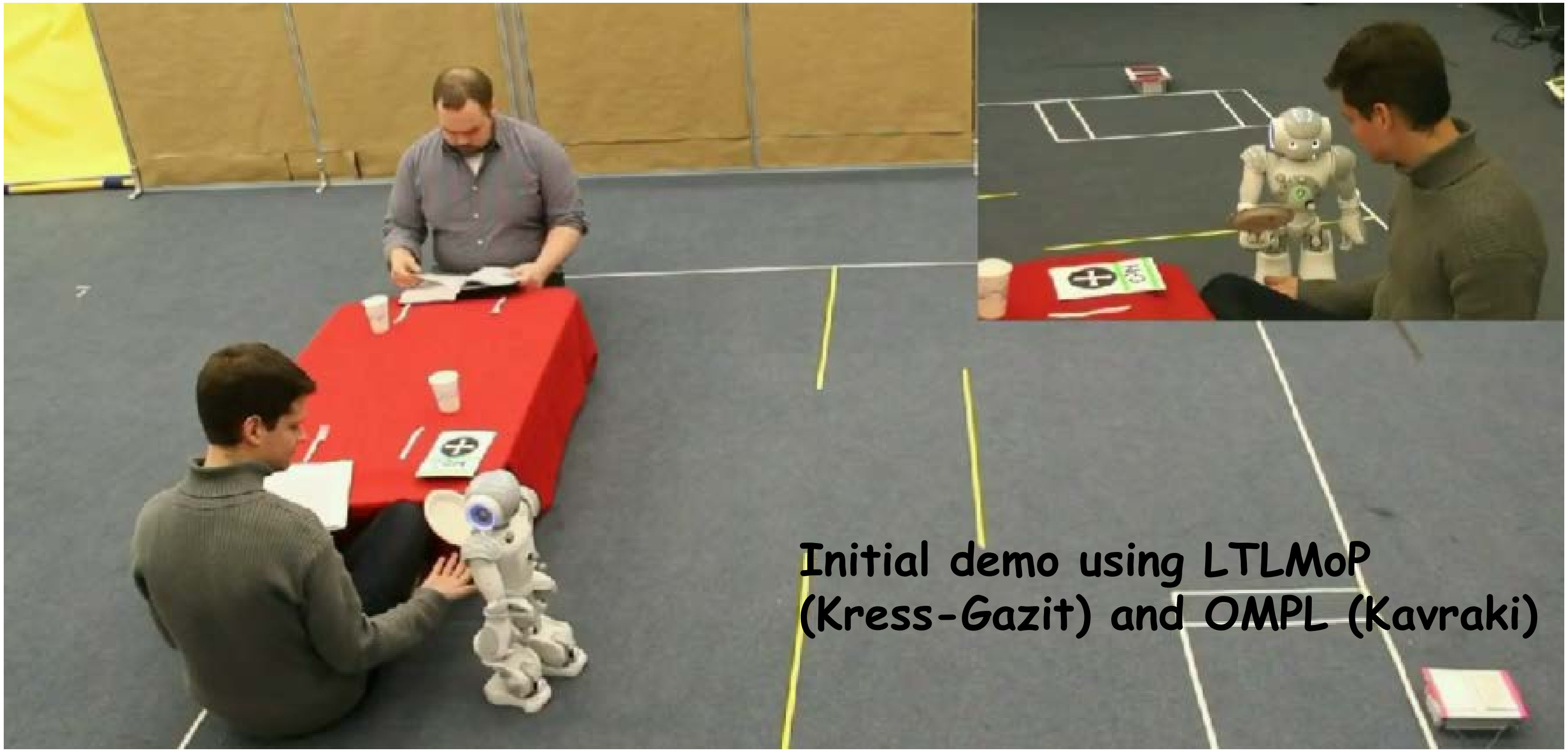


Research Results

- Improving the scalability of core engine for mapping Temporal Logic formulas to Controllers:
  - Synthesis with identifiers (Kress-Gazit, Seshia)
- Synthesis of cost-optimal plans (Kress-Gazit)
- Motion planning in partially unknown environments (Kavraki, Vardi)
- Synthesis of controllers with robust performance in presence of uncertainties
  - Theory of robustness for hybrid systems (Tabuada)
- Accuracy in mapping discrete actions to continuous-time trajectories with durations (Kress-Gazit)
- Automatic generation of environment assumptions (Alur, Topcu)

Ongoing Case Study: Robotic Waiter

- Challenges: Scalability (items, costumers), uncertainty in sensing and actuation, optimality of behavior, fault recovery
- Future plans: exploit symmetries, robust synthesis, task specific abstractions



Synthesis for Online Education

- Emerging opportunity: MOOCs

- Challenge: Personalized feedback on assignments
  - Manual feedback by TAs (not scalable)
  - Grading by peers (not reliable)
  - Evaluation on test cases (how to translate failed tests to errors?)

- Application for ExCAPE tools for synthesis
  - Introductory programming assignments (Solar-Lezama, MIT)
  - Scheduling problems in Embedded Systems course (Seshia, UC Berkeley)
  - DFA construction in Theory of Computation (with Hartmann, UC Berkeley)

    see automatatutor.com

Sample Problem: Derivative of a Polynomial

```
def computeDeriv(poly):
    result = []
    for i in range(len(poly)):
        result += [i * poly[i]]
    if len(poly) == 1:
        return result # return [0]
    else:
        return result[1:]  # remove the leading 0
```

Autograder Output on a Student Solution

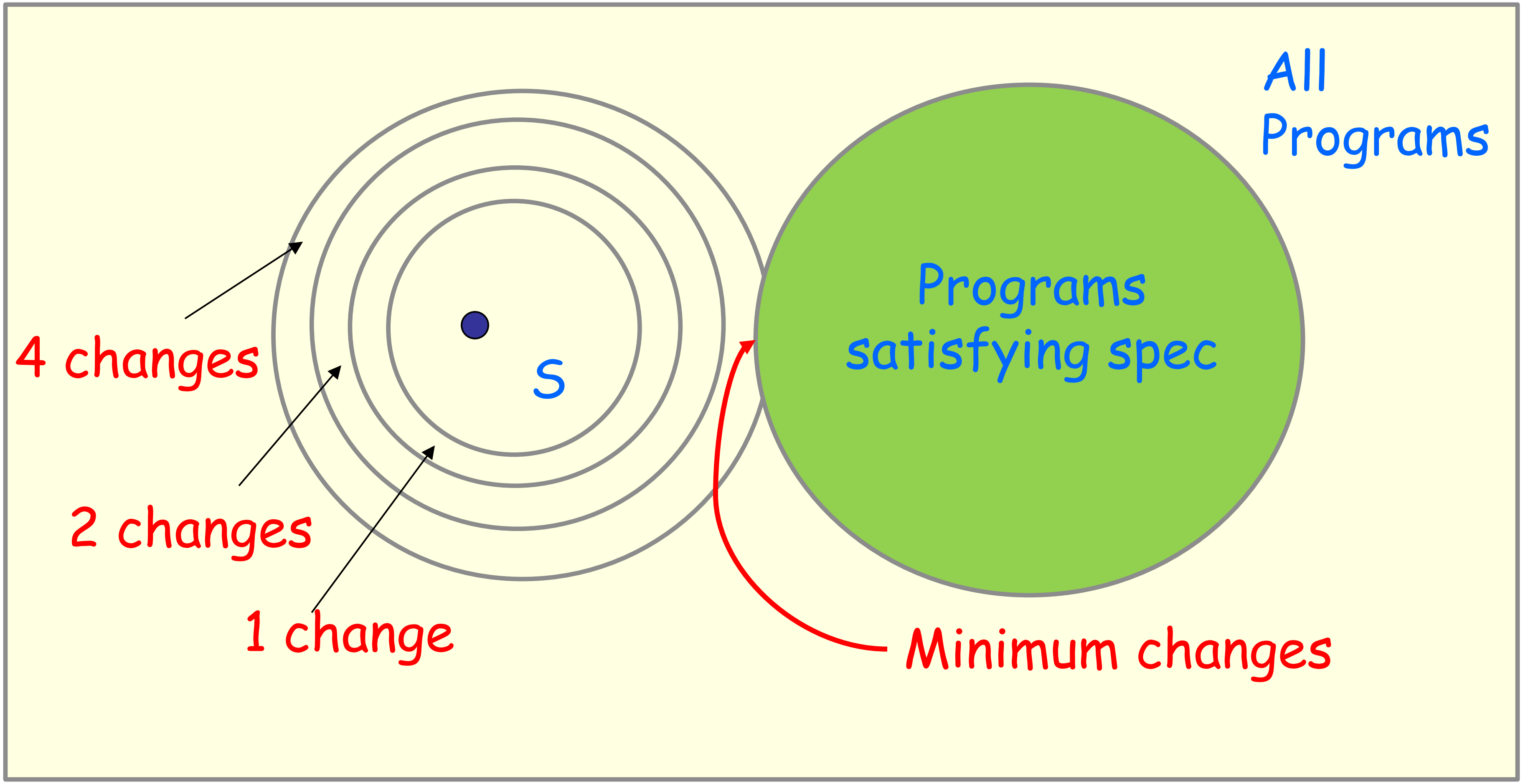
```
1  def computeDeriv(poly):
2      deriv = []
3      zero = 0
4      if (len(poly)==1):
5          return deriv
6      for e in range(0,len(poly)):
7          if(poly[e]==0):
8              zero += 1
9          else:
10             deriv.append(poly[e]*e)
11
12     return deriv
```

Student Solution
+ Reference Solution
+ Error Model

The program requires **3** changes:

- In the return statement **return deriv** in **line 5**, replace **deriv** by **[0]**.
- In the comparison expression **(poly[e] == 0)** in **line 7**, change **(poly[e] == 0)** to **False**.
- In the expression **range(0, len(poly))** in **line 6**, replace **0** by **1**.

Computational Engine: Sketch Synthesis Tool



S: Student Solution

Autograder Experiments (MIT 6.00)

Benchmark	TestSet	Generated Feedback	Percentage	AvgTime(s)
prodBySum-6.00	268	218	81.34%	2.49
oddTuples-6.00	344	185	53.78%	2.65
compDeriv-6.00	103	88	85.44%	12.95
evalPoly-6.00	13	6	46.15%	3.35
compBal-stdin-6.00	52	17	32.69%	29.57
compDeriv-6.00x	918	753	82.03%	12.42
evalPoly-6.00x	541	167	30.87%	4.78
oddTuples-6.00x	1756	860	48.97%	4.14
iterPower-6.00x	2875	1693	58.89%	3.58
recurPower-6.00x	2938	2271	77.30%	10.59
iterGCD-6.00x	2988	2052	68.67%	17.13
hangman1-6.00x	351	171	48.72%	9.08
hangman2-6.00x	218	98	44.95%	22.09

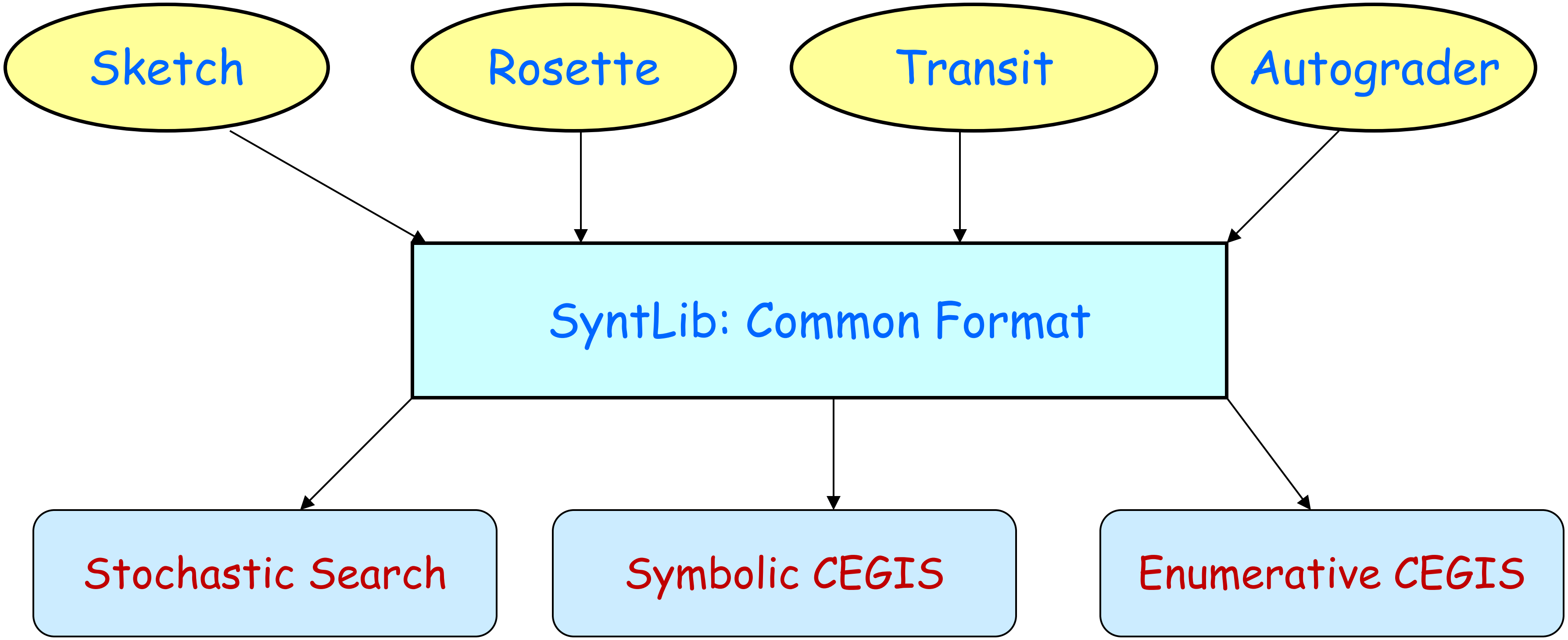
64%

9.9s

Theory / Methodology / Tools / Application domains

- Rosette: Framework for developing synthesis-enhanced DSLs (Bodik)
- Enhancements to Sketch to support modularity (Solar-Lezama)
- Bridging the gap between reactive synthesis and supervisory control (Lafortune, Tripakis, Vardi)
- Verified LLVM Infrastructure (Martin, Zdancewic)
- Platform-based design for software synthesis (Sangiovanni-Vincetelli)
- Synthesis of logic for avoiding concurrency bugs (Lafortune)
- Component-based synthesis for probabilistic systems (Vardi)
- Theory of regular functions for quantitative analysis (Alur)
- Automated cloud configuration using synthesis(Alur, Loo, Parthasarathy)
- Route Shepherd for configuration of routing protocols (Loo)
- Synthesis of control + scheduling for wireless control networks (Pappas)
- Programming for mobile platforms (Foster, Solar-Lezama)
- User studies for improving programming notations (Hartmann)

Template-based Synthesis Modulo Theories



Based on input format for SMTLib 2

Problem: Given a formula ϕ in an SMT theory with an extra function symbol f, and context-free language L for templates, find an expression e in L such that $\phi[f/e]$ is valid

Basis for synthesis competition (to be held at CAV 2014)

Education and Outreach

- ExCAPE Summer School: June 13—16, Berkeley; 125 registrants
  - Tutorials: Reactive synthesis (Vardi)
    - Constraint-based program synthesis (Bodik)
    - Synthesis for cyber-physical systems (Tabuada)
  - + Talks
- ExCAPE Webinar: Monthly talks on diverse topics
- Sponsored workshops
  - SYNT (at CAV 2013, by Solar-Lezama)
  - Synthesis for robotics (at RSS 2013, by Kavraki and Kress-Gazit)
  - Special session on synthesis at ACC 2013 (by Lafortune)
- Graduate course at Berkeley: Program synthesis for everyone
- K-12 programs: CURIE @ Cornell

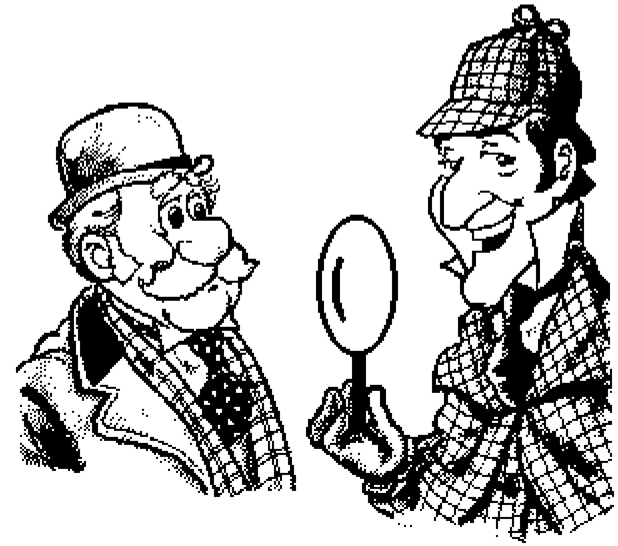
Rotating Postdoc Program

- Each ExCAPE postdoc has two mentors, at two different institutions

- Year 2012-13:
  - Ruediger Ehlers (Robotics)
    - Mentors: Kress-Gazit (Cornell), Seshia (UC Berkeley)

- For the upcoming year:
  - Xiaokang Qiu (UIUC), Apps for mobile platforms
    - Mentors: Foster (Maryland), Solar-Lezama (MIT)
  - Indranil Saha (UCLA), Robotics
    - Mentors: Pappas (Penn), Seshia (UC Berkeley)
  - Christos Stergiou (UC Berkeley), Multicore protocols
    - Mentors: Martin (Penn), Tripakis (UC Berkeley)
  - Damian Zufferey (IST Austria), Networked systems
    - Mentors: Loo (Penn), Parthasarathy (UIUC)

ExCAPE
Expeditions in Computer Augmented
Program Engineering



- Paradigm shift in synthesis:
  - Old: Allow more concise, high-level description
  - New: Designer uses multiple, natural formats,
    - Synthesis tool assists in discovering tricky logic

- Paradigm shift in design tools:
  - Old : Any compiler transformation must be polynomial-time
  - New: Computational intractability not a show-stopper

- Common theme: Guided search in a space of programs to find one that meets multiple design goals
  - A bit like model checking, but can be interactive!