A game-based abstraction-refinement framework for probabilistic systems

Marta Kwiatkowska

University of Oxford

Joint work with Mark Kattenbelt, Gethin Norman and Dave Parker

Logic and Algorithms, University of Edinburgh, 2008
Overview

• This talk: combines model checking for probabilistic systems and stochastic games

• Background
  – What is quantitative/probabilistic model checking?
  – Why games?

• How it works
  – Markov decision processes
  – Abstractions for Markov decision processes
  – Approximating probabilistic/expected reachability
  – Abstraction-refinement loop
  – Experimental results

• Where next
  – Current projects
Probabilistic model checking

- **Model checking**
  - Inputs:
    - finite-state transition system + temporal logic specification
  - Outputs:
    - “yes”/“no” + counterexample (e.g. trace to error state)

- **Quantitative/probabilistic model checking**
  - Inputs:
    - finite-state probabilistic model annotated with quantitative rewards, e.g. Markov decision process
    - probabilistic temporal logic specification, e.g. PCTL
  - Outputs:
    - “yes”/”no” + quantitative results/plots
Motivation

• Probability
  – randomised algorithms/protocols
  – systems with component failures

• Probability and nondeterminism
  – concurrency: asynchronous parallel composition of probabilistic components
  – underspecification: unknown probabilities or model parameters
  – abstraction (see later)

• PRISM case studies
  – randomised distributed algorithms (e.g. leader election, self-stabilisation), randomised communication protocols (e.g. FireWire, Bluetooth, Zeroconf), randomised security protocols (e.g. anonymity, contract signing), etc...
Markov decision processes (MDPs)

- **Model both nondeterministic and probabilistic behaviour**
  - extension of discrete-time Markov chains
  - nondeterministic choice between probability distributions

- Formally, an MDP $M$ is a tuple
  - $(S, s_{init}, \text{Steps}, L)$
- where:
  - $S$ is a finite set of states
  - $s_{init} \in S$ is the initial state
  - $\text{Steps} : S \rightarrow 2^{\text{Act} \times \text{Dist}(S)}$ is the transition probability function
    - $\text{Act}$ is a set of actions
    - $\text{Dist}(S)$ is the set of discrete probability distributions over $S$
  - $L : S \rightarrow 2^{\text{AP}}$ is a labelling function with atomic propositions from a set $\text{AP}$
Paths and adversaries

- A (finite or infinite) path through an MDP
  - is a sequence of (connected) states
  - represents an execution of the system
  - resolves both the probabilistic and nondeterministic choices

- An adversary (aka. “scheduler” or “policy”) of an MDP
  - is a resolution of nondeterminism only
  - is (formally) a mapping from finite paths to distributions
  - results in a fully probabilistic model
  - i.e. an (infinite–state) Markov chain over finite paths
  - on which we can define a probability space over infinite paths

- Adversary \( A \) is simple iff: \( A(s_1...s_n) = A(s_n) \) for all \( s_1...s_n \)
  - in this case, resulting model reduces to finite Markov chain
Example adversary

- Fragment of DTMC for adversary which picks b then c in $s_1$
Probabilistic reachability

- Probabilistic reachability (for a set of goal states $F \subseteq S$)
  - $p_s^A(F)$ probability of reaching $F$ from state $s$ under adversary $A$
- Minimum/maximum probabilities over all adversaries
  - $p_s^{\text{min}}(F) = \inf_A p_s^A(F)$
  - $p_s^{\text{max}}(F) = \sup_A p_s^A(F)$

- For probabilistic reachability, simple adversaries suffice
- Two techniques to compute (for all states $s$)
  - linear optimisation problem (polynomial complexity)
  - value iteration (dynamic programming) – simple iterative numerical method
  - in both cases a best/worst simple adversary also generated
- Also use graph-based algorithms as precomputation steps
  - for qualitative verification or to reduce round-off problems
Min/max/average probability that a message is successfully sent by time T

Probability that 10% of gate outputs are erroneous for varying gate failure rates and numbers of stages

Optimum probability of leader election by time T for various coin biases
Costs and rewards

• Augment models with rewards (or, conversely, costs)
  – real-valued quantities assigned to states and/or transitions
  – no distinction between rewards (“good”) and costs (“bad”)
  – simple but flexible, many possible interpretations

• Some examples:
  – elapsed time, power consumption, size of buffer, number of messages successfully delivered, net profit, …

• Analyse (min/max) expected value of these costs/rewards both with instantaneous and cumulative interpretation e.g.:
  – maximum expected message queue size at time t?
  – maximum expected power consumption for the duration of the protocol?
  – minimum expected number of correctly delivered packets within 100 clock-ticks?
**Firewire:**
Maximum expected time for leader election for various coin biases

**Self-stabilisation:**
Worst-case expected number of steps to stabilise for initial configurations with K tokens amongst N processes

**Bluetooth:**
Distribution of expected time for two replies to be received, over all possible initial configurations of sender/receiver (1.7x10^{10} states)
Abstraction

• Very successful in (non-probabilistic) model checking
• Construct abstract model $M'$ of concrete model $M$
  – details not relevant to some property of interest removed
  – merge states according to a given partition of state space
  – e.g. from set of predicates (predicate abstraction)
• Non-probabilistic case: existential abstraction
  – conservative: satisfaction in $M'$ implies satisfaction in $M$
  – converse does not hold, but...
  – information from model checking process (counterexample) can be used to refine the abstraction or validate the property
    • i.e. check if counterexample is spurious (there is no equivalent concrete trace)
  – CEGAR: counterexample–guided abstraction and refinement
Abstraction of MDPs

- Abstraction increases degree of nondeterminism
  - i.e. minimum probabilities are lower and maximums higher
- But what form does the abstract MDP take?
  - MDPs with more nondeterministic choices? [D'Argenio et al.]
  - MDPs with probability intervals? (like [Fecher et al.] for DTMCs)
  - here: two-player stochastic games [QEST'06]
- Key idea: separate two forms of nondeterminism
  - (a) from abstraction and (b) from original MDP
  - then generate separate lower/upper bounds for min/max

- gives quantitative measure of utility of abstraction
- basis of a CEGAR framework
Stochastic two-player games

- **Simple stochastic games** [Condon]
- **Game** $G = ((V,E), v_{init}, (V_1,V_2,V_P), \delta)$
  - $(V,E)$ is a finite directed graph
  - $v_{init}$ is the initial vertex
  - $(V_1,V_2,V_P)$ is a partition of $V$: 'player 1', 'player 2' and 'probabilistic'
  - $\delta : V_P \rightarrow \text{Dist}(V)$ is a probabilistic transition function
- **Execution of $G$**: successor vertex chosen:
  - by player 1/2 for $V_1/V_2$ vertices
  - at random ($\delta$) for $V_P$ vertices

- MDPs can be thought of as stochastic two-player games with no $V_2$ vertices and strict alternation between $V_1/V_P$
Properties of stochastic games

- **Resolution of nondeterminism in a stochastic game**
  - is done by a pair of strategies for players 1 and 2: $(\sigma_1, \sigma_2)$
  - under which the behaviour of the game is fully probabilistic

- **Probabilistic reachability of vertex goal set $F$**
  - $p_v^{\sigma_1, \sigma_2}(F)$ probability of reaching $F$ from vertex $v$ under $(\sigma_1, \sigma_2)$

- **Optimal probabilities for player 1 and player 2**
  - $\sup_{\sigma_1} \inf_{\sigma_2} p_v^{\sigma_1, \sigma_2}(F)$ and $\sup_{\sigma_2} \inf_{\sigma_1} p_v^{\sigma_1, \sigma_2}(F)$
  - computable via simple iterative methods, similar to MDPs
Abstract MDP

- **Abstract MDP** is a two-player stochastic game
  - based on a partition $P_S$ of MDP state space $S$
  - $V_1$ vertices are elements of $P_S$ (subsets of $S$)
  - $V_2$ vertices are sets of prob. distributions (“states of MDP”)
  - $V_P$ vertices are single probability distributions (over $V_1$)
  - strict alternation between $V_1$, $V_2$, $V_P$ vertices
- **Player 1 controls nondeterminism from abstraction**
  - selects a state of the original MDP from a subset of $S$ (in $P_S$)
- **Player 2 controls nondeterminism from original MDP**
  - selects a single probability distribution from a set
MDP \rightarrow \text{Abstract MDP}

- Player 1 vertices are partition elements (abstract states)
MDP $\rightarrow$ Abstract MDP

- (Sets of) distributions are lifted to the abstract state space
MDP → Abstract MDP

- States with same (sets of) choices form player vertices
MDP → Abstract MDP

• Complete transformation:
Analysis of abstract MDP

- For a stochastic game built from an MDP and partition $P_S$
- Let $s \in S$ be an MDP state, $v \in V$ the corresponding game vertex (i.e. $s \in v$) and $F \in P_S$ a set of goal states
- Analysis of game yields lower/upper bounds for MDP:

$$\inf_{\sigma_1, \sigma_2} p_v^{\sigma_1, \sigma_2}(F) \leq p_s^{\min}(F) \leq \sup_{\sigma_1} \inf_{\sigma_2} p_v^{\sigma_1, \sigma_2}(F)$$

$$\sup_{\sigma_2} \inf_{\sigma_1} p_v^{\sigma_1, \sigma_2}(F) \leq p_s^{\max}(F) \leq \sup_{\sigma_1, \sigma_2} p_v^{\sigma_1, \sigma_2}(F)$$
Analysis of abstract MDP

• For a stochastic game built from an MDP and partition $P_S$
• Let $s \in S$ be an MDP state, $v \in V$ the corresponding game vertex (i.e. $s \in v$) and $F \in P_S$ a set of goal states
• Analysis of game yields lower/upper bounds for MDP:

\[
\begin{align*}
\inf_{\sigma_1, \sigma_2} p_{v}^{{\sigma_1}, {\sigma_2}}(F) & \leq p_{s}^{\min}(F) \leq \sup_{\sigma_1} \inf_{\sigma_2} p_{v}^{\sigma_1, \sigma_2}(F) \\
\sup_{\sigma_2} \inf_{\sigma_1} p_{v}^{\sigma_1, \sigma_2}(F) & \leq p_{s}^{\max}(F) \leq \sup_{\sigma_1, \sigma_2} p_{v}^{\sigma_1, \sigma_2}(F)
\end{align*}
\]

min/max reachability probabilities for original MDP
Analysis of abstract MDP

- For a stochastic game built from an MDP and partition $P_S$
- Let $s \in S$ be an MDP state, $v \in V$ the corresponding game vertex (i.e. $s \in v$) and $F \in P_S$ a set of goal states
- Analysis of game yields lower/upper bounds for MDP:

$$\inf_{\sigma_1, \sigma_2} p_{v}^{\sigma_1, \sigma_2}(F) \leq p_s^{\min}(F) \leq \sup_{\sigma_1} \inf_{\sigma_2} p_{v}^{\sigma_1, \sigma_2}(F)$$

$$\sup_{\sigma_2} \inf_{\sigma_1} p_{v}^{\sigma_1, \sigma_2}(F) \leq p_s^{\max}(F) \leq \sup_{\sigma_1, \sigma_2} p_{v}^{\sigma_1, \sigma_2}(F)$$

optimal probabilities for player 1, player 2 in abstract MDP
Analysis of abstract MDP

- For a stochastic game built from an MDP and partition $P_S$
- Let $s \in S$ be an MDP state, $v \in V$ the corresponding game vertex (i.e. $s \in v$) and $F \in P_S$ a set of goal states
- Analysis of game yields lower/upper bounds for MDP:

$$\inf_{\sigma_1, \sigma_2} p_v^{\sigma_1, \sigma_2}(F) \leq p_s^{\text{min}}(F) \leq \sup_{\sigma_1} \inf_{\sigma_2} p_v^{\sigma_1, \sigma_2}(F)$$

$$\sup_{\sigma_2} \inf_{\sigma_1} p_v^{\sigma_1, \sigma_2}(F) \leq p_s^{\text{max}}(F) \leq \sup_{\sigma_1, \sigma_2} p_v^{\sigma_1, \sigma_2}(F)$$

like min/max reachability probabilities on MDPs
(but performed on abstract MDP)
Generating the abstraction

• How to efficiently construct the abstraction [QAPL’08]

- Bottleneck: construction of concrete system (MDP)
Generating the abstraction

- How to efficiently construct the abstraction [QAPL’08]
Generating the abstraction

• **Language level abstraction**
  – based on **predicate abstraction**
  – applied **compositionally** (can cause loss of precision when predicates refer to variables of different components)

• **Prototype implemented in PRISM**
  – uses SMT ALL-SAT procedures
  – and symbolic data structures (**BDDs** and **MTBDDs**)  

• **Results demonstrate that:**
  – larger models can be verified
  – models can be verified faster
  – compositional approach often vital
Abstraction: Results

- Israeli & Jalfon’s Self Stabilisation [IJ90]

- Protocol for obtaining a stable state in a token ring
- Minimum probability of reaching a stable state by $T$
Abstraction: Results

- IPv4 Zeroconf [CAG02]

- Protocol for obtaining an IP address for a new host
- Maximum probability the new host not configured by T
Abstraction: Results

- Sliding Window Protocol

- Protocol for sending data over an insecure medium
- Maximum probability of K timeouts
Abstraction refinement loop for MDPs

- Quantitative analogue of CEGAR using stochastic games
  - difference between lower/upper bounds gives “error”
  - i.e. a quantitative measure of the preciseness of the abstraction
Partition refinement

- **Refine when error above some threshold**
  - error equals the difference between lower and upper bound
  - aim is to reduce the difference between the bounds (make the “error” smaller)

- **Partition refinement**
  - replace the partition of the state space used to construct the abstract MDP with a finer partition
  - a finer partition yields a more precise abstraction

- **Consider two different partition refinement schemes**
  - strategy-based
  - value-based
Partition refinement

• **Strategy-based**
  - consider a strategy pair that obtains the lower bound
  - consider a strategy pair that obtains the upper bound
  - since actual value lies between these bounds one of these strategy pairs must be *spurious*
    - *spurious*: make choices that are not valid in the concrete system
  - therefore refine so that choices are eliminated
    - split elements of the partition where the strategy pairs make different choices

• **Value-based**
  - similar but do not look at single a strategy pair for the lower or upper bound
  - look at all strategies that obtain the upper/lower bound and base the refinement on the difference between these sets
Partition refinement – Example

- Expected reward of reaching “E” from initial state (A)
  - all transitions have reward 1
Partition refinement – Example

- **Expected reward of reaching “E” from initial state (A)**
  - Initially partition state space into target set and remaining states ({A,B,C,D} and {E})
Partition refinement – Example

- Expected reward of reaching “E” from initial state (A)
  - lower bound is 2 (choose player 2 state “D” in \{A,B,C,D\})
  - upper bound is \(\infty\) (choose player 2 state “A,B,C” in \{A,B,C,D\})
Partition refinement – Example

- Expected reward of reaching “E” from initial state (A)
  - lower bound is 2 (choose player 2 state “D” in \{A,B,C,D\})
  - upper bound is \(\infty\) (choose player 2 state “A,B,C” in \{A,B,C,D\})

Difference between upper and lower bound strategies:

in player 1 state \{A,B,C,D\}:
- lower bound choose “D”
- upper bound choose “A,B,C”

Therefore under either refinement strategy split \{A,B,C,D\} into \{A,B,C\} and \{D\}
Partition refinement – Example

• Expected reward of reaching “E” from initial state (A)
  – after refinement new partition is \{A,B,C\}, \{D\} and \{E\}
  – and new abstract MDP is given by
Partition refinement – Example

- Expected reward of reaching “E” from initial state (A)
  - now \{A,B,C\} yields three distinct player 2 vertices
Partition refinement – Example

- Expected reward of reaching “E” from initial state (A)
  - lower bound is 4 (choose player 2 state “C” in \{A,B,C\})
  - upper bound is $\infty$ (choose player 2 state “A” or “B” in \{A,B,C\})
Partition refinement – Example

- Expected reward of reaching “E” from initial state (A)
  - lower bound is 4 (choose player 2 state “C” in \{A,B,C\})
  - upper bound is \(\infty\) (choose player 2 state “A” or “B” in \{A,B,C\})

under strategy-based refinement split based on difference between one lower and one upper bound strategy

two possibilities:
- split due to choices “C” and “A” in \{A,B,C\}
- split due to choices “C” and “B” in \{A,B,C\}

in either case split \{A,B,C\} into \{A\}, \{B\} and \{C\}
Partition refinement – Example

- **Expected reward of reaching “E” from initial state (A)**
  - lower bound is 4 (choose player 2 state “C” in \{A,B,C\})
  - upper bound is ∞ (choose player 2 state “A” or “B” in \{A,B,C\})

under value-based refinement split based on difference between set of all lower bound strategies and set of all upper bound strategies

- “C” only lower bound choice in \{A,B,C\}
- “A” and “B” both upper bound choices in \{A,B,C\}

therefore split \{A,B,C\} into \{A,B\} and \{C\}
Partition refinement – Convergence of bounds

- **IEEE 802.3 CSMA/CD Ethernet network protocol**
  - used to ensure only one node is transmitting
  - minimum probability a station’s backoff counter reaches 2
Partition refinement – Convergence of bounds

- **IEEE 802.3 CSMA/CD Ethernet network protocol**
  - used to ensure only one node is transmitting
  - minimum probability a station’s backoff counter reaches 4
Partition refinement – Convergence of bounds

- **Randomised consensus shared coin protocol (2 processes)**
  - maximum expected time until termination
Partition refinement – Results

- Randomised consensus shared coin protocol (4 processes)
  - maximum expected time until termination
## Partition refinement – Results

<table>
<thead>
<tr>
<th></th>
<th>MDP states</th>
<th>Abstract MDP states (refinement steps)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Strategy–based</td>
</tr>
<tr>
<td><strong>IEE 802.3</strong></td>
<td>92,978</td>
<td>419 (53)</td>
</tr>
<tr>
<td><strong>CSMA/CD</strong></td>
<td>793,110</td>
<td>1,671 (138)</td>
</tr>
<tr>
<td></td>
<td>2,221,189</td>
<td>3,297 (266)</td>
</tr>
<tr>
<td><strong>Randomised</strong></td>
<td>173,056</td>
<td>1,298 (47)</td>
</tr>
<tr>
<td><strong>consensus</strong></td>
<td>637,696</td>
<td>5,008 (155)</td>
</tr>
<tr>
<td></td>
<td>1,257,216</td>
<td>9,987 (282)</td>
</tr>
<tr>
<td><strong>IPv4</strong></td>
<td>552,097</td>
<td>2,277 (208)</td>
</tr>
<tr>
<td><strong>Zerconf</strong></td>
<td>1,065,567</td>
<td>2,277 (188)</td>
</tr>
<tr>
<td></td>
<td>2,092,513</td>
<td>1,720 (217)</td>
</tr>
</tbody>
</table>
## Partition refinement – Results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>MDP states</th>
<th>Abstract MDP states (refinement steps)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Strategy-based</td>
</tr>
<tr>
<td>IEE 802.3 CSMA/CD</td>
<td>92,978</td>
<td>419 (53)</td>
</tr>
<tr>
<td></td>
<td>793,110</td>
<td>1,671 (138)</td>
</tr>
<tr>
<td></td>
<td>2,221,189</td>
<td>3,297 (266)</td>
</tr>
<tr>
<td>Randomised consensus</td>
<td>173,056</td>
<td>1,298 (47)</td>
</tr>
<tr>
<td></td>
<td>637,696</td>
<td>5,008 (155)</td>
</tr>
<tr>
<td></td>
<td>1,257,216</td>
<td>9,987 (282)</td>
</tr>
<tr>
<td>IPv4 Zerconf</td>
<td>552,097</td>
<td>2,277 (208)</td>
</tr>
<tr>
<td></td>
<td>1,065,567</td>
<td>2,277 (188)</td>
</tr>
<tr>
<td></td>
<td>2,092,513</td>
<td>1,720 (217)</td>
</tr>
</tbody>
</table>

- Orders of magnitude reduction in the state space
Partition refinement – Results

<table>
<thead>
<tr>
<th></th>
<th>MDP states</th>
<th>Abstract MDP states (refinement steps)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Strategy–based</td>
</tr>
<tr>
<td>IEE 802.3 CSMA/CD</td>
<td>92,978</td>
<td>419 (53)</td>
</tr>
<tr>
<td></td>
<td>793,110</td>
<td>1,671 (138)</td>
</tr>
<tr>
<td></td>
<td>2,221,189</td>
<td>3,297 (266)</td>
</tr>
<tr>
<td>Randomised consensus</td>
<td>173,056</td>
<td>1,298 (47)</td>
</tr>
<tr>
<td></td>
<td>637,696</td>
<td>5,008 (155)</td>
</tr>
<tr>
<td></td>
<td>1,257,216</td>
<td>9,987 (282)</td>
</tr>
<tr>
<td>IPv4 Zerconf</td>
<td>552,097</td>
<td>2,277 (208)</td>
</tr>
<tr>
<td></td>
<td>1,065,567</td>
<td>2,277 (188)</td>
</tr>
<tr>
<td></td>
<td>2,092,513</td>
<td>1,720 (217)</td>
</tr>
</tbody>
</table>

- Often in a small number of refinement steps
Improved numerical computation

- **Numerical computation is expensive therefore**
  - do not want to *repeat numerical computation*
  - do not want to *throw away previous computations*

- **Improved algorithm re-uses previous computation**
  - if bounds agree, then value will not change, and therefore do not need to compute the values for these abstract states again
  - lower bounds from current abstraction can be used as initial approximation when computing upper bounds
  - lower bounds for previous abstraction can be used as initial approximation when computing upper bounds
Improved numerical computation

<table>
<thead>
<tr>
<th>Scenario</th>
<th>PRISM time (seconds)</th>
<th>Refinement time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>total (build)</td>
</tr>
<tr>
<td>IEE 802.3 CSMA/CD</td>
<td>41.39 134.3</td>
<td>654.3 (616.3)</td>
</tr>
<tr>
<td>Randomised consensus</td>
<td>2,379 14,956</td>
<td>7,368 (5,637)</td>
</tr>
<tr>
<td>IPv4 Zerconf</td>
<td>149.5 269.8</td>
<td>1,584 (1,574)</td>
</tr>
</tbody>
</table>

- Prototype slower than PRISM
  - due to time to build abstraction (at each step first requires the construction of the concrete system)
### Improved numerical computation

<table>
<thead>
<tr>
<th></th>
<th>PRISM time (seconds)</th>
<th>Refinement time (seconds)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>total (build)</td>
<td>optimisation saving</td>
</tr>
<tr>
<td>IEE 802.3 CSMA/CD</td>
<td>41.39</td>
<td>654.3 (616.3)</td>
<td>61.79</td>
</tr>
<tr>
<td></td>
<td>134.3</td>
<td>3,580 (3,322)</td>
<td>394.6</td>
</tr>
<tr>
<td>Randomised consensus</td>
<td>2,379</td>
<td>7,368 (5,637)</td>
<td>4,027</td>
</tr>
<tr>
<td></td>
<td>14,956</td>
<td>34,133 (21,837)</td>
<td>41,878</td>
</tr>
<tr>
<td>IPv4 Zerconf</td>
<td>149.5</td>
<td>1,584 (1,574)</td>
<td>18.83</td>
</tr>
<tr>
<td></td>
<td>269.8</td>
<td>3,300 (3,287)</td>
<td>52.47</td>
</tr>
</tbody>
</table>

- Optimisation of numerical algorithm yields substantial saving
Conclusions

• Introduced novel abstraction approach for MDPs using two player games
  – separation of nondeterminism from MDP/abstraction
  – both lower/upper bounds for min/max probabilities/rewards
  – quantitative measure of the utility of abstraction
  – combined with predicate abstraction to obtain a language level abstraction

• Promising results on model-level refinement
  – refinements yields small abstract models
  – limitation: requires construction of concrete model (partition refinement)
Related work

• Abstraction for MDPs
  – D’Argenio, Larsen, Katoen, Fecher, Wolf, ...
  – Use partitions, probability intervals, etc

• Counter-examples for probabilistic systems
  – Katoen&Han
  – Fully probabilistic systems (no MDPs)

• CEGAR for probabilistic systems
  – Hermanns et al
  – Language-level abstraction
  – One sided bounds only

• Key difference: here stochastic games
  – Two sided bounds, better precision
Current work

• Work in progress: quantitative software verification
  – ANSI–C programs
    · goto–cc, SATABS
  – combine predicate abstraction and refinement schemes
  – refinement: predicate discovery
  – consider nondeterminism and probabilistic behaviour
    · probability from random assignment
    · nondeterminism from input
  – consider rewards
    · Best/worst case time, power usage, buffer sizes, …

• Handling C programs introduces new challenges…