Quantitative verification techniques for probabilistic software

Marta Kwiatkowska

Oxford University Computing Laboratory

Summer School on Model Checking, Beijing, October 2010

Course overview

- 3 sessions (Mon/Tue/Thur): 6 × 50 minute lectures
  - 1: Markov decision processes (MDPs)
  - 2: Probabilistic LTL model checking
  - 3: Compositional probabilistic verification
  - 4: Abstraction, refinement and probabilistic software
  - 5: Probabilistic timed automata (PTAs)
  - 6: Software with time and probabilities
- For additional background material
  - and an accompanying list of references
  - see: http://www.prismmodelchecker.org/lectures/

Part 4

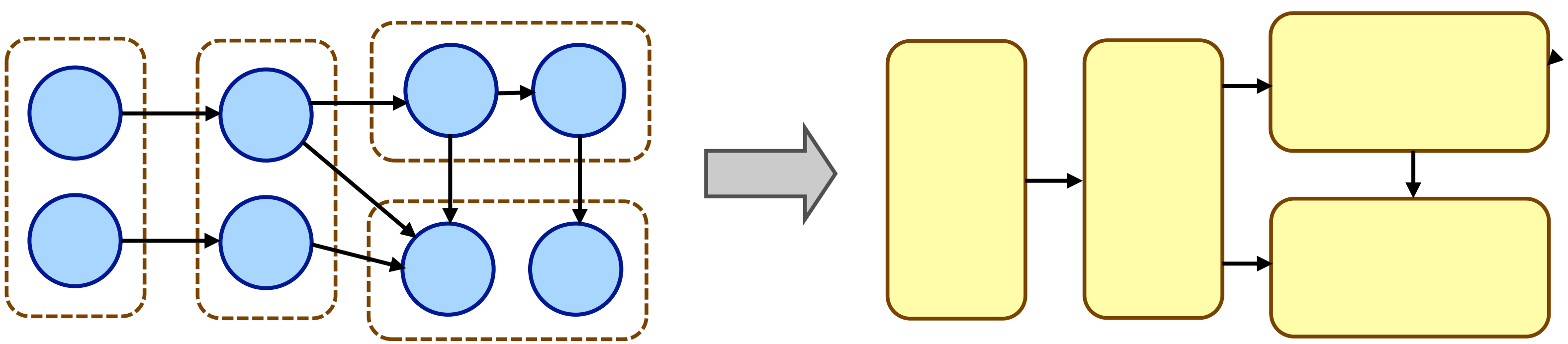
Abstraction, refinement and probabilistic software

Overview (Part 4)

- Abstraction & refinement (CEGAR)
- Abstraction of MDPs using stochastic games
- Quantitative abstraction refinement
- Probabilistic software verification

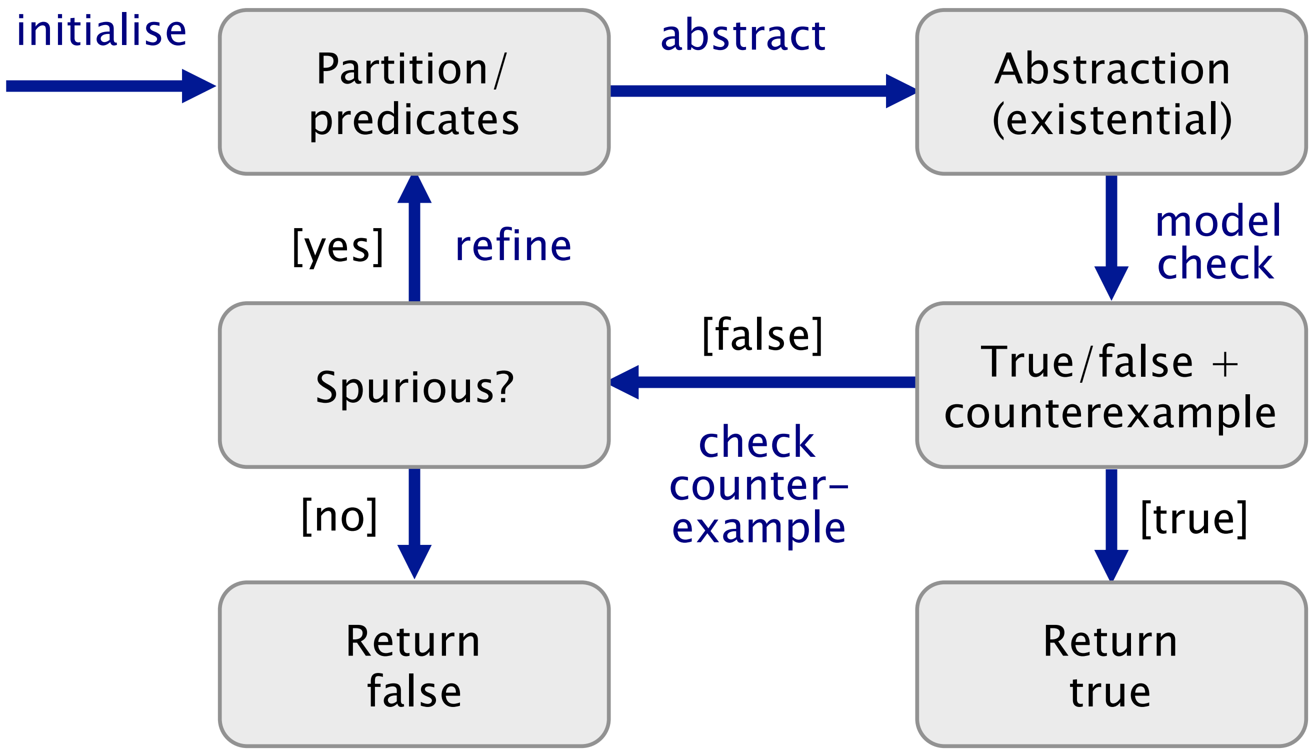
Abstraction

- Very successful in (non-probabilistic) formal methods
  - essential for verification of large/infinite-state systems
  - hide details irrelevant to the property of interest
  - yields smaller/finite model which is easier/feasible to verify
  - loss of precision: verification can return “don’t know”
- Construct abstract model of a concrete system
  - e.g. based on a partition of the concrete state space
  - an abstract state represents a set of concrete states



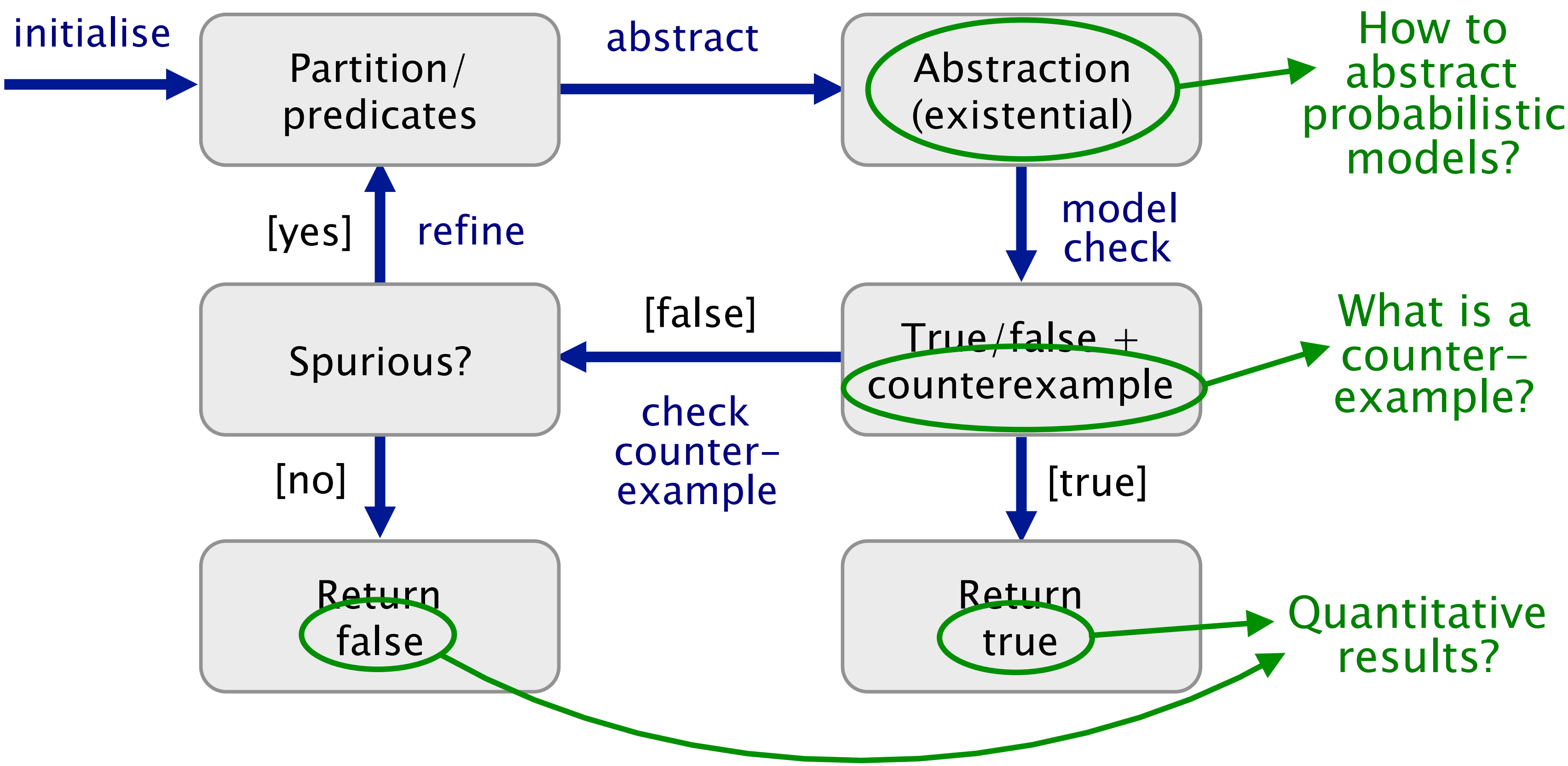
Abstraction refinement (CEGAR)

- Counterexample-guided abstraction refinement
  - (non-probabilistic) model checking of reachability properties



Abstraction refinement (CEGAR)

- Counterexample-guided abstraction refinement
  - (~~non~~-probabilistic) model checking of reachability properties

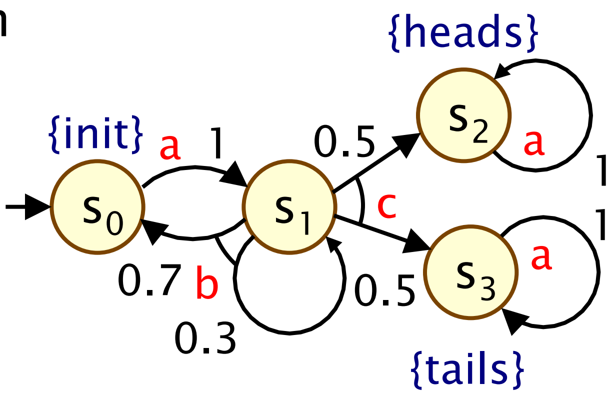


Overview (Part 4)

- Abstraction & refinement (CEGAR)
- Abstraction of MDPs using stochastic games
- Quantitative abstraction refinement
- Probabilistic software verification

Recap: MDPs

- Markov decision processes (MDPs)
  - mix probability and nondeterminism

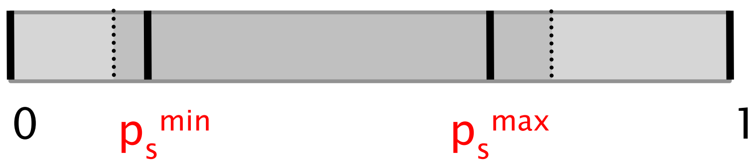


- An adversary σ for an MDP M
  - resolves nondeterministic choices based on history so far
  - induces probability measure $Pr_{M,s}^{\sigma}$ over (infinite) paths $Path_{M,s}^{\sigma}$

- Properties:
  - key property: probabilistic reachability
  - quantify over all possible adversaries
  - $Pr_M^{min}(\Diamond F) = \inf_{\sigma} \{ Pr_{M,s}^{\sigma}(\Diamond F) \}$
  - $Pr_M^{max}(\Diamond F) = \sup_{\sigma} \{ Pr_{M,s}^{\sigma}(\Diamond F) \}$
  - here, we will abbreviate these to $p_s^{\sigma}(F)$, $p_s^{min}(F)$ and $p_s^{max}(F)$

Abstraction of MDPs

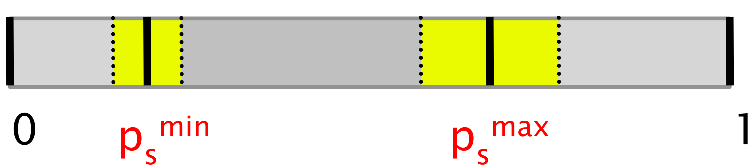
- Abstraction increases degree of nondeterminism
  - i.e. minimum probabilities are lower and maximums higher



- But what form does the abstraction of an MDP take?
- 2 possibilities:

  (i) an MDP [DJJL01]
  - probabilistic simulation relates concrete/abstract models

  (ii) a stochastic two-player game [KNP06]
  - separates nondeterminism from abstraction and from MDP
  - yields separate lower/upper bounds for min/max

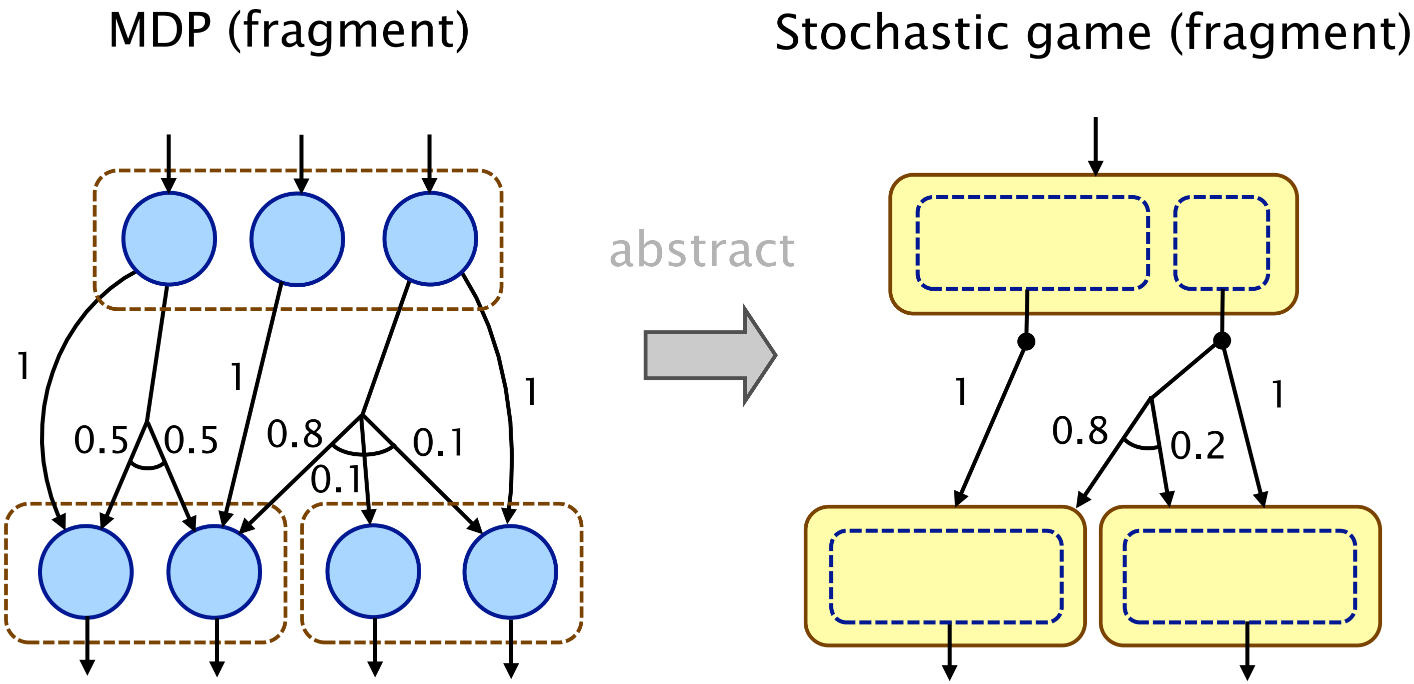


Stochastic two-player games

- Subclass of simple stochastic games [Shapley,Condon]
  - two nondeterministic players (1 and 2) and probabilistic choice
- Resolution of the nondeterminism in a game
  - corresponds to a pair of strategies for players 1 and 2: (σ_1,σ_2)
  - $p_a^{\sigma 1,\sigma 2}(F)$ probability of reaching F from a under (σ_1,σ_2)
  - can compute, e.g. : $\sup_{\sigma 1} \inf_{\sigma 2} p_a^{\sigma 1,\sigma 2}(F)$
  - informally: “the maximum probability of reaching F that player 1 can guarantee no matter what player 2 does”
- Abstraction of an MDP as a stochastic two-player game:
  - player 1 controls the nondeterminism of the abstraction
  - player 2 controls the nondeterminism of the MDP

Game abstraction (by example)

- Player 1 vertices (▭) are abstract states
- (Sets of) distributions are lifted to the abstract state space
- Player 2 vertices (⅄) are states with same (sets of) choices



Properties of the abstraction

- Analysis of game yields lower/upper bounds:
  - for target $F \in A$, $s \in S$ and $a \in A$ with $s \in a$

$$\inf_{\sigma 1,\sigma 2} p_a^{\sigma 1,\sigma 2}(F) \;\le\; p_s^{min}(F) \;\le\; \sup_{\sigma 1} \inf_{\sigma 2} p_a^{\sigma 1,\sigma 2}(F)$$

$$\inf_{\sigma 1} \sup_{\sigma 2} p_a^{\sigma 1,\sigma 2}(F) \;\le\; p_s^{max}(F) \;\le\; \sup_{\sigma 1,\sigma 2} p_a^{\sigma 1,\sigma 2}(F)$$

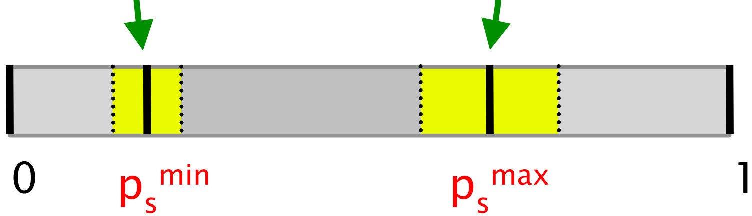
Properties of the abstraction

- Analysis of game yields lower/upper bounds:
  - for target $F \in A$, $s \in S$ and $a \in A$ with $s \in a$

$$\inf_{\sigma1,\sigma2} p_a^{\sigma1,\sigma2}(F) \leq p_s^{min}(F) \leq \sup_{\sigma1} \inf_{\sigma2} p_a^{\sigma1,\sigma2}(F)$$

$$\inf_{\sigma1} \sup_{\sigma2} p_a^{\sigma1,\sigma2}(F) \leq p_s^{max}(F) \leq \sup_{\sigma1,\sigma2} p_a^{\sigma1,\sigma2}(F)$$

min/max reachability probabilities for original MDP



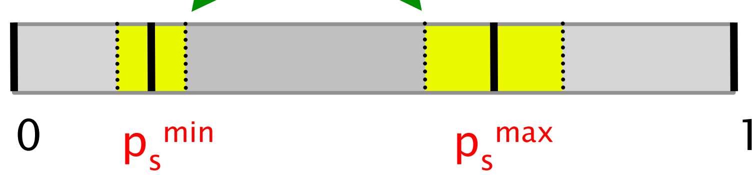
Properties of the abstraction

- Analysis of game yields lower/upper bounds:
  - for target $F \in A$, $s \in S$ and $a \in A$ with $s \in a$

$$\inf_{\sigma 1,\sigma 2} p_a^{\sigma 1,\sigma 2}(F) \;\le\; p_s^{min}(F) \;\le\; \sup_{\sigma 1} \inf_{\sigma 2} p_a^{\sigma 1,\sigma 2}(F)$$

$$\inf_{\sigma 1} \sup_{\sigma 2} p_a^{\sigma 1,\sigma 2}(F) \;\le\; p_s^{max}(F) \;\le\; \sup_{\sigma 1,\sigma 2} p_a^{\sigma 1,\sigma 2}(F)$$

optimal probabilities for player 1, player 2 in game



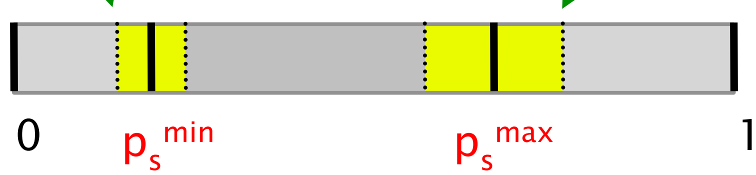
Properties of the abstraction

- Analysis of game yields lower/upper bounds:
  - for target $F \in A$, $s \in S$ and $a \in A$ with $s \in a$

$$\inf_{\sigma1,\sigma2} p_a^{\sigma1,\sigma2}(F) \leq p_s^{min}(F) \leq \sup_{\sigma1} \inf_{\sigma2} p_a^{\sigma1,\sigma2}(F)$$

$$\inf_{\sigma1} \sup_{\sigma2} p_a^{\sigma1,\sigma2}(F) \leq p_s^{max}(F) \leq \sup_{\sigma1,\sigma2} p_a^{\sigma1,\sigma2}(F)$$

min/max reachability probabilities, treating game as MDP
(i.e. assuming that players 1 and 2 cooperate)

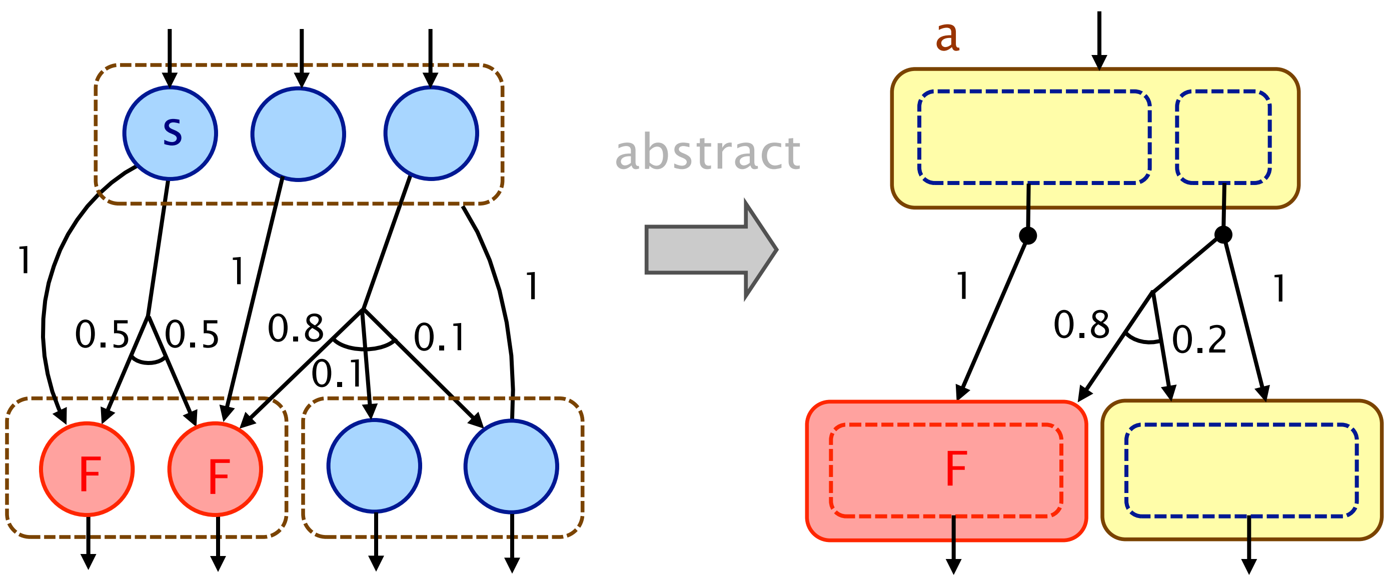


Example – Abstraction

$p_s^{max}(F) = 1 \in [0.8, 1]$

$p_a^{lb,max}(F) = 0.8$

$p_a^{ub,max}(F) = 1$

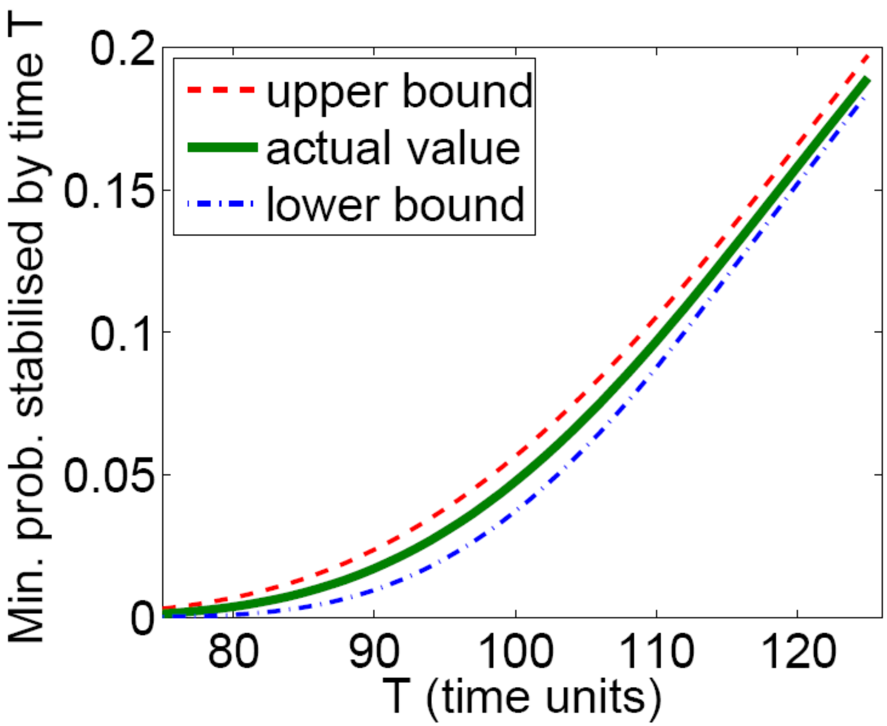


where $p_a^{lb,max}(F)$ denotes $\inf_{\sigma 1} \sup_{\sigma 2} p_a^{\sigma 1, \sigma 2}(F)$

and where $p_a^{ub,max}(F)$ denotes $\sup_{\sigma 1, \sigma 2} p_a^{\sigma 1, \sigma 2}(F)$

Experimental results

- Israeli & Jalfon's Self Stabilisation
  - protocol for obtaining a stable state in a token ring
  - minimum probability of reaching a stable state by time T

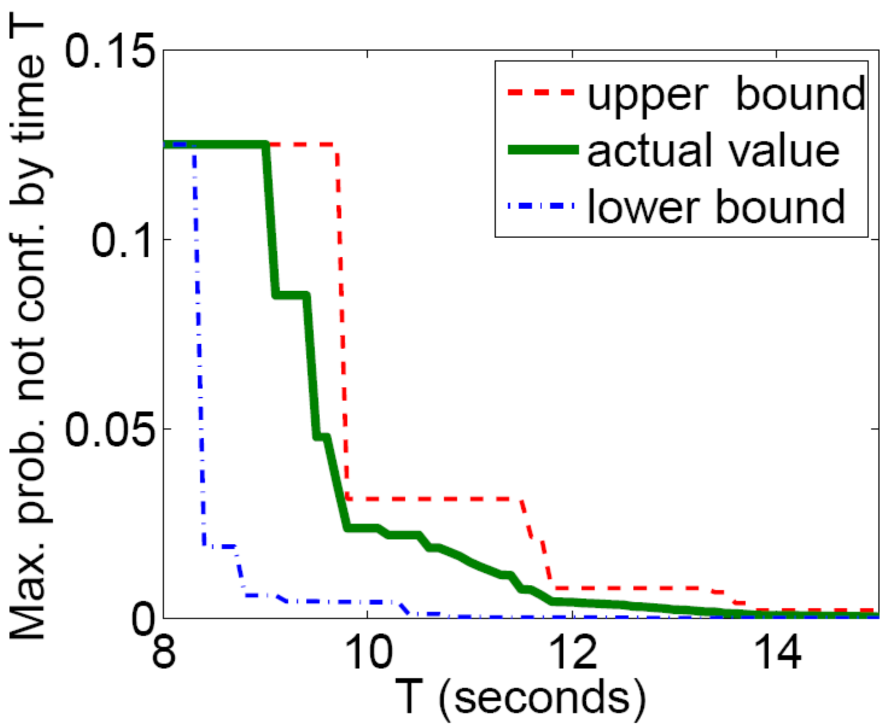


concrete states: 1,048,575

abstract states: 627

Experimental results

- IPv4 Zeroconf
  - protocol for obtaining an IP address for a new host
  - maximum probability the new host not configured by T



concrete states: 838,905

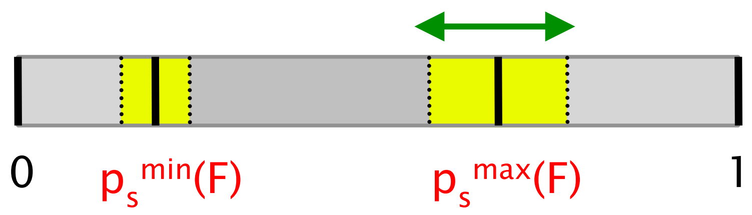
abstract states: 881

Overview (Part 4)

- Abstraction & refinement (CEGAR)
- Abstraction of MDPs using stochastic games
- Quantitative abstraction refinement
- Probabilistic software verification

Abstraction refinement

- Consider (max) difference between lower/upper bounds
  - gives a quantitative measure of the abstraction's precision



- If the difference ("error") is too great, refine the abstraction
  - a finer partition yields a more precise abstraction
  - lower/upper bounds can tell us where to refine (which states)
  - (memoryless) strategies can tell us how to refine

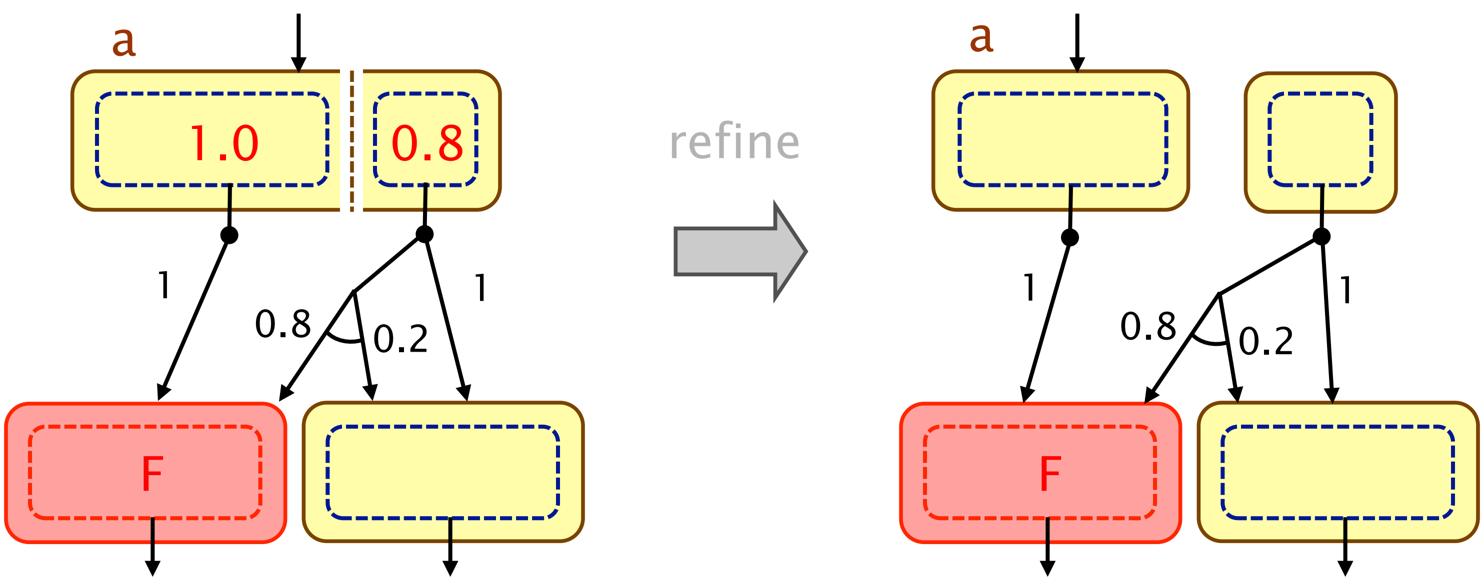
Example – Refinement

$p_s^{max}(F) = 1 \in [0.8,1]$

"error" = 0.2

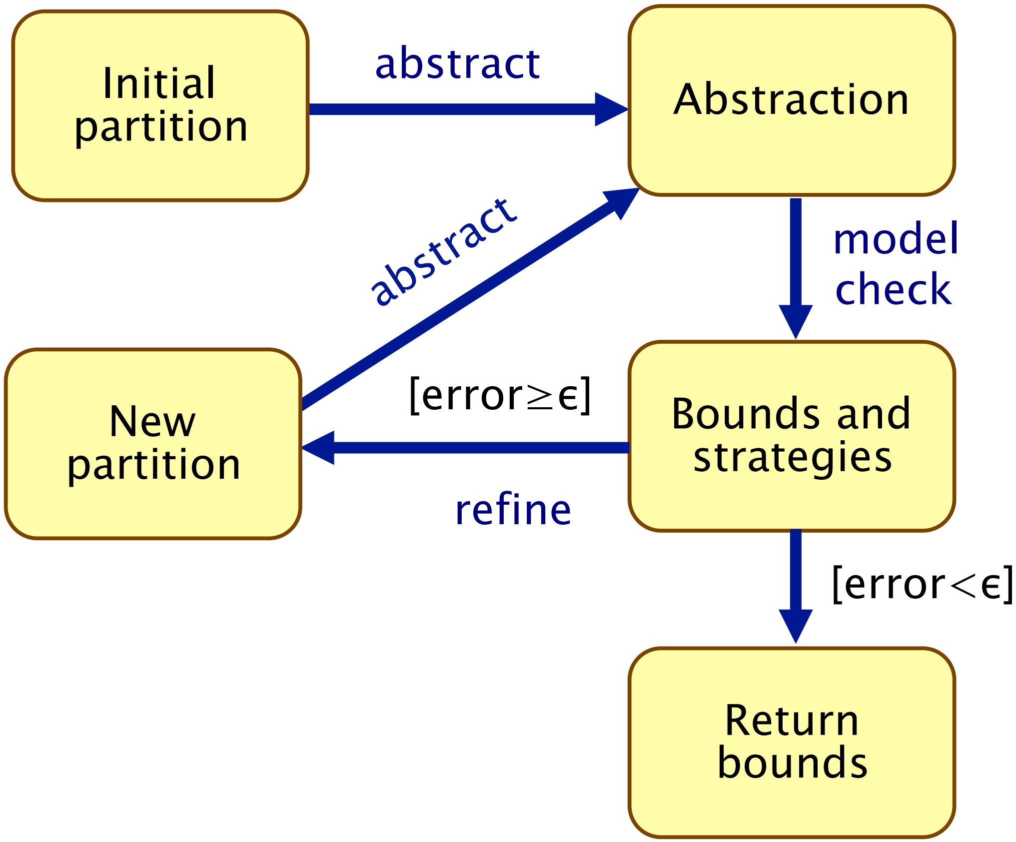
$p_s^{max}(F) = 1 \in [1,1]$

"error" = 0



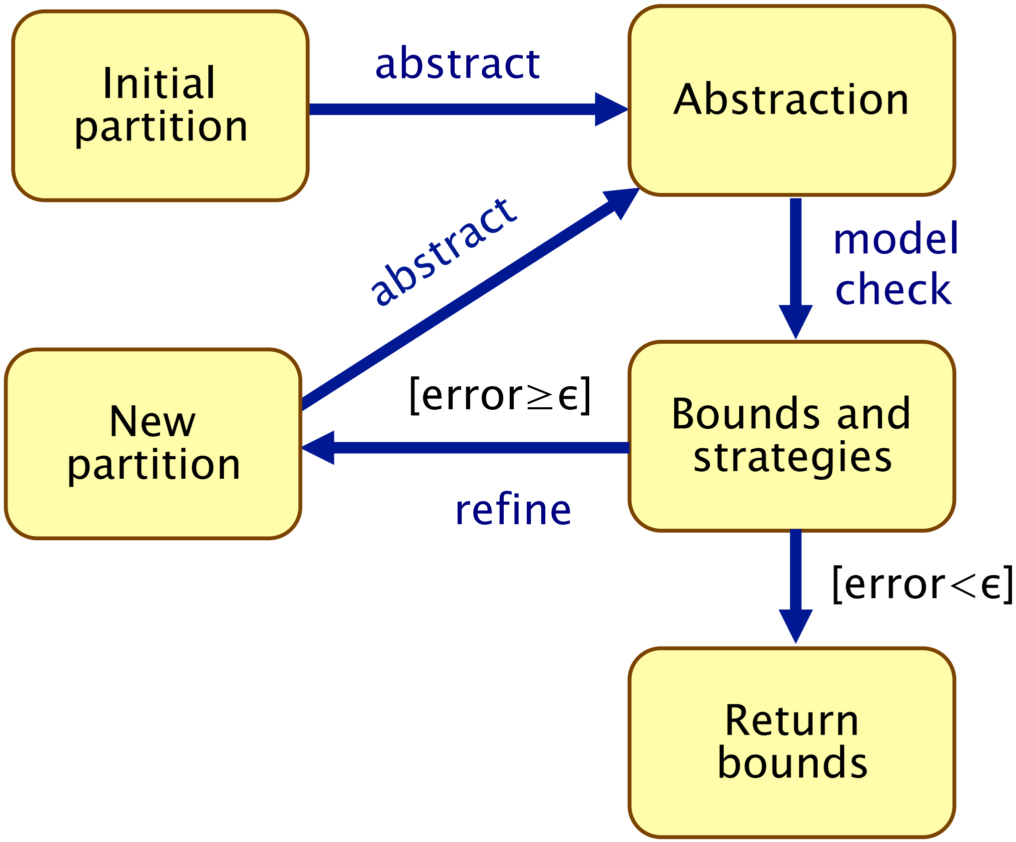
Abstraction-refinement loop

- Quantitative abstraction-refinement loop for MDPs



Abstraction-refinement loop

- Quantitative abstraction-refinement loop for MDPs



- Refinements yield strictly finer partition
- Guaranteed to converge for finite models
- Guaranteed to converge for infinite models with finite bisimulation

Abstraction-refinement loop

- Implementations of quantitative abstraction refinement…
- Verification of probabilistic timed automata [KNP09c]
  - zone-based abstraction/refinement using DBMs
  - implemented in (development release of) PRISM
  - outperforms existing PTA verification techniques
- Verification of probabilistic software [KKNP09]
  - predicate abstraction/refinement using SAT solvers
  - implemented in tool qprover: components of PRISM, SATABS
  - analysed real network utilities (ping, tftp) - approx 1KLOC
- Verification of concurrent PRISM models [WZ10]
  - implemented in tool PASS; infinite-state PRISM models

Overview (Part 4)

- Abstraction & refinement (CEGAR)
- Abstraction of MDPs using stochastic games
- Quantitative abstraction refinement
- Probabilistic software verification

Probabilistic software

- Consider sequential ANSI C programs
  - support functions, pointers, arrays, but not dynamic memory allocation, unbounded recursion, floating point op.s
- Add function bool coin(double p) for probabilistic choice
  - for modelling e.g. failures, randomisation
- Add function int ndet(int n) for nondeterministic choice
  - for modelling e.g. user input, unspecified function calls
- Focus on software where failure is unavoidable
  - e.g. network protocols/utilities, esp. wireless
- Quantitative properties based on probabilistic reachability
  - e.g. maximum probabilistic of unsuccessful data transmission
  - e.g. minimum expected number of packets sent

Example – sample target program

```
bool fail = false;
int c = 0;
int main()
{
    // nondeterministic
    c = num_to_send();
    while (! fail && c > 0)
    {
        // probabilistic
        fail = send_msg();
        c--;
    }
}
```

Program:

- Loop that tries to send c messages
- c is obtained from num_to_send() (returns 0/1/2 nondeterministically)
- send_msg() fails with probability 0.1
- Any failure causes loop to terminate

Property:

- "what is the minimum/maximum probability of the program terminating with fail being true?"

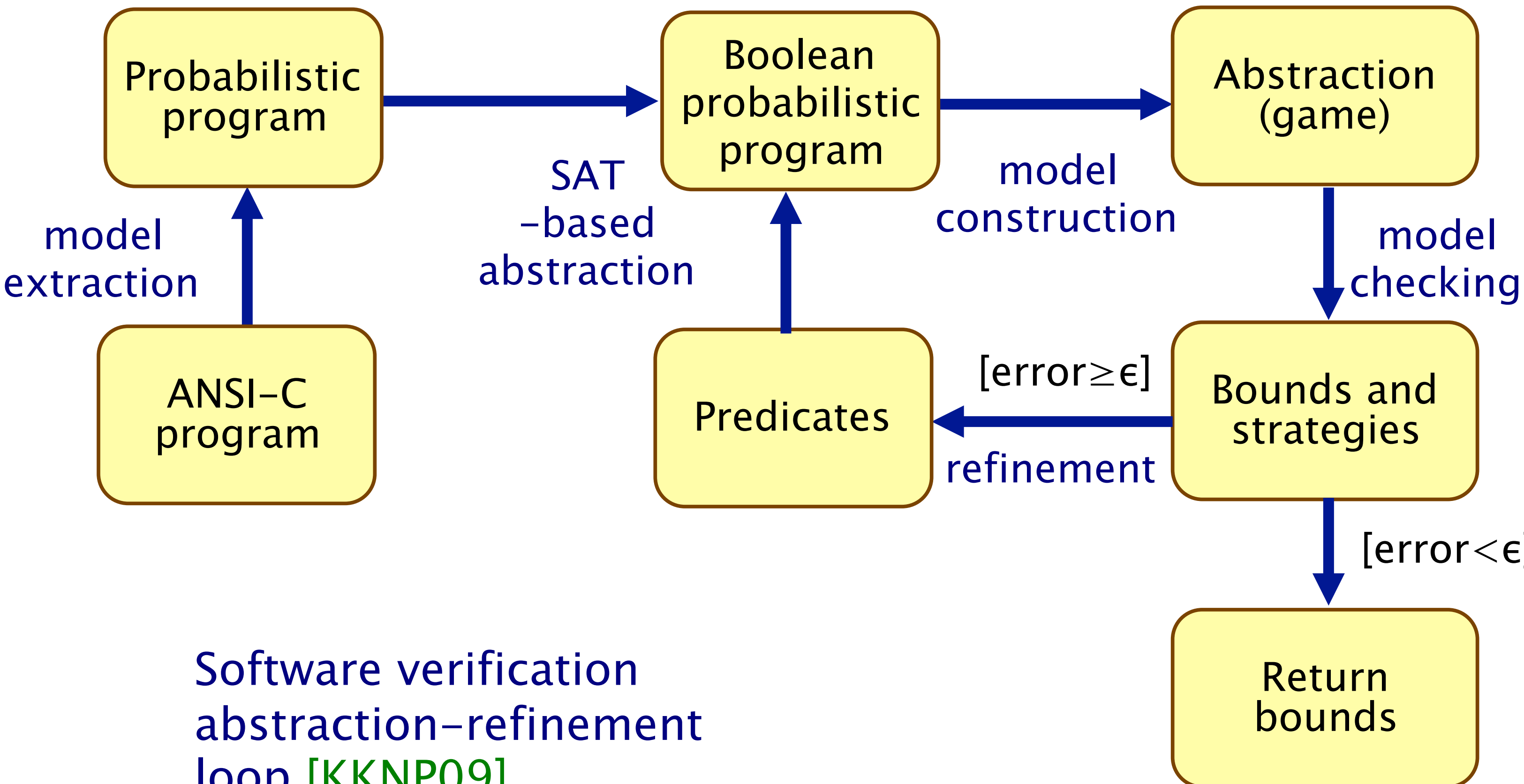
Example – simplified

```
bool fail = false;
int c = 0;
int main()
{
    // nondeterministic
    c = ndet(3);
    while (! fail && c > 0)
    {
        // probabilistic
        fail = coin(0.1);
        c--;
    }
}
```

Property:

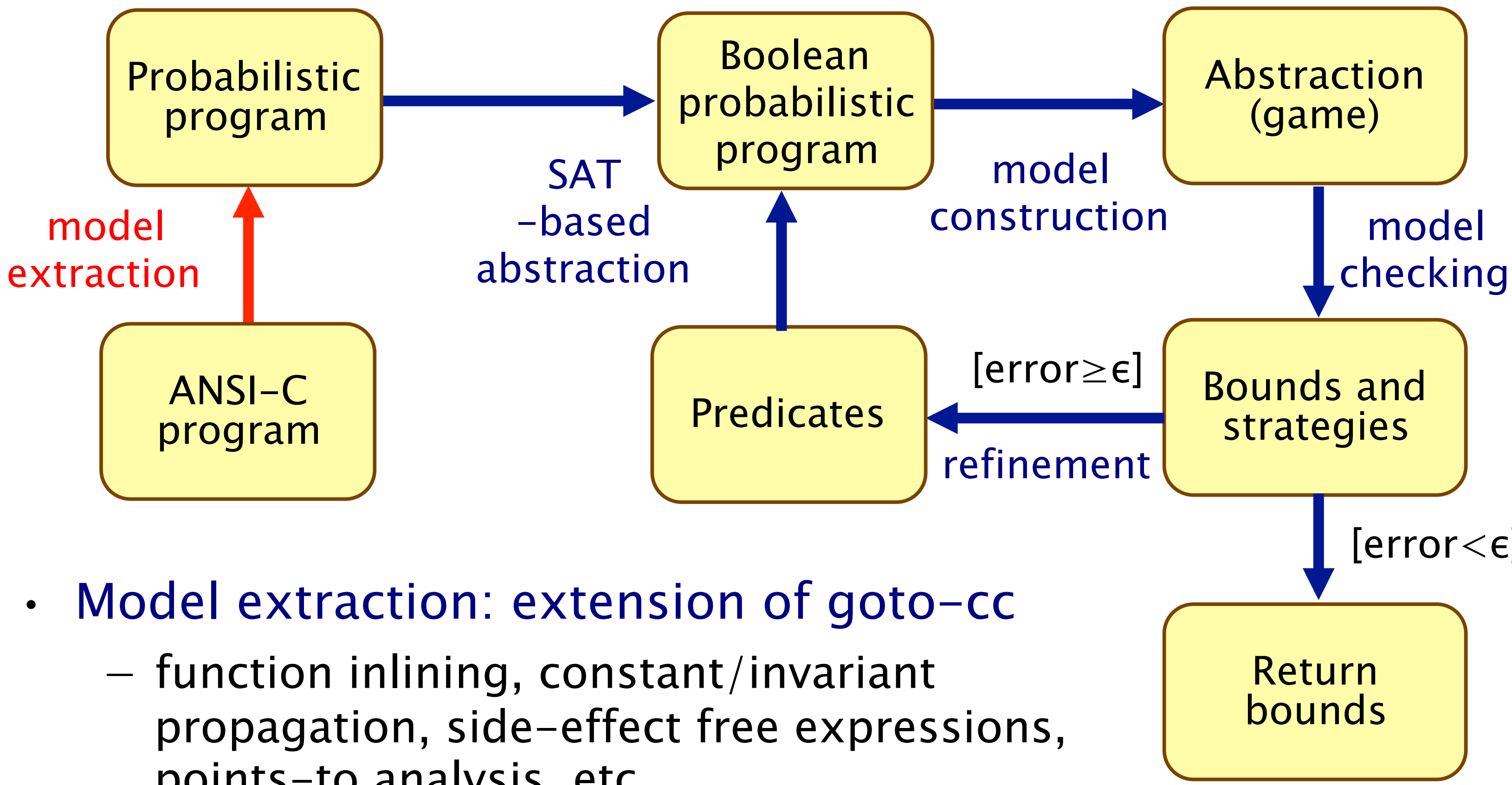
- "what is the minimum/maximum probability of the program terminating with fail being true?"

Abstraction-refinement loop



Software verification abstraction-refinement loop [KKNP09]

Abstraction-refinement loop



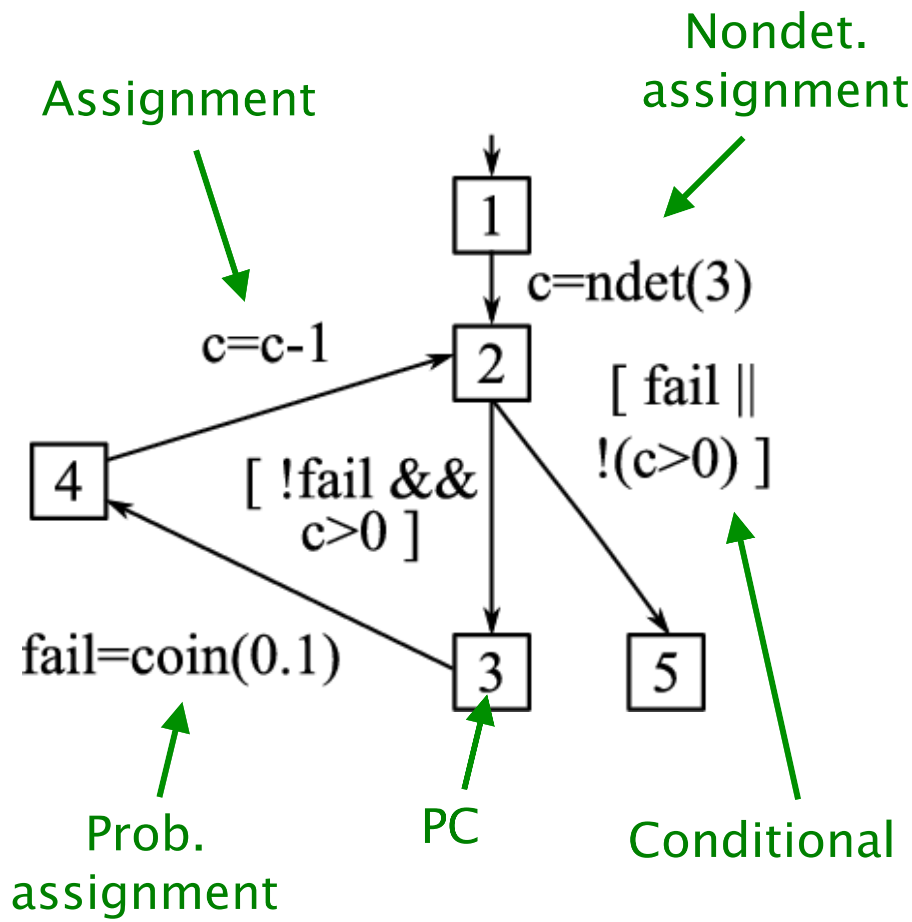
- Model extraction: extension of goto-cc
  - function inlining, constant/invariant propagation, side-effect free expressions, points-to analysis, etc.
- Probabilistic program
  - probabilistic control flow graph
  - Markov decision process (MDP) semantics

Back to example

C code:

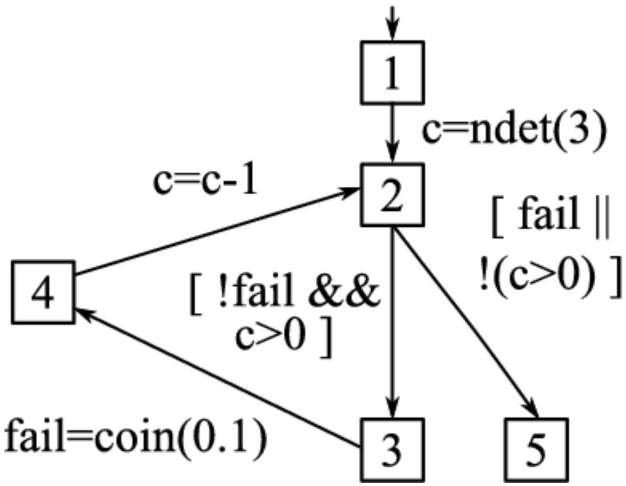
```
bool fail = false;
int c = 0;
int main()
{
    // nondeterministic
    c = ndet(3);
    while (! fail && c > 0)
    {
        // probabilistic
        fail = coin(0.1);
        c--;
    }
}
```

Probabilistic program:

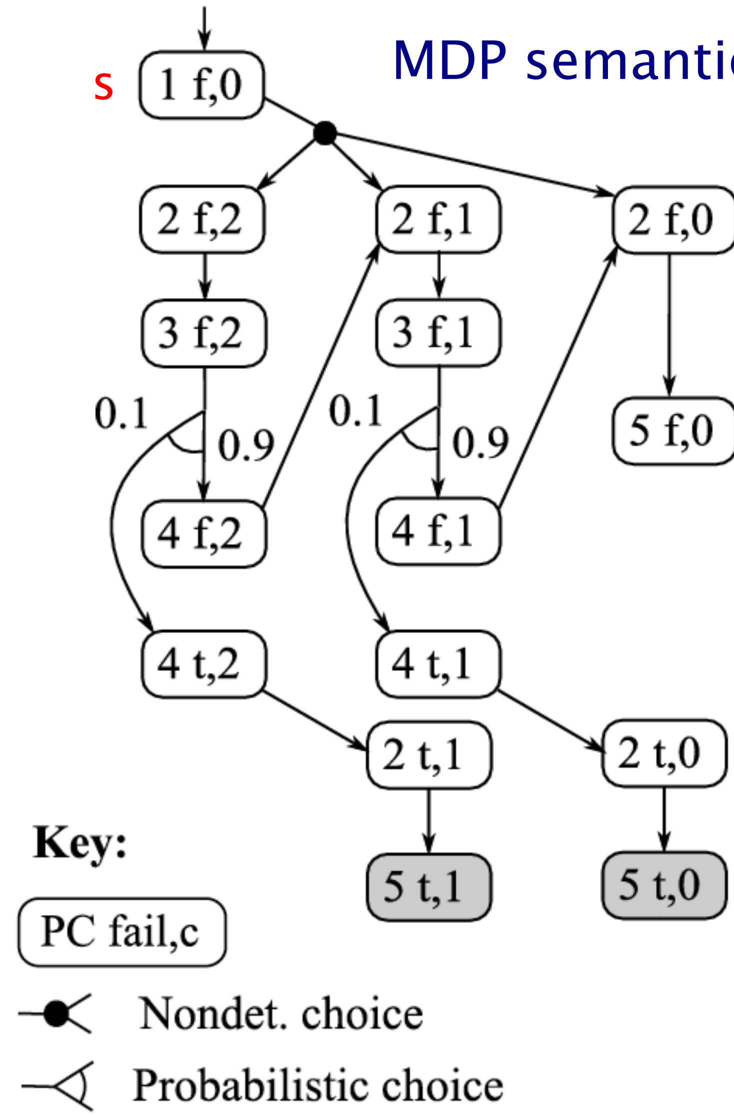


Probabilistic program as MDP

Probabilistic program:



MDP semantics:

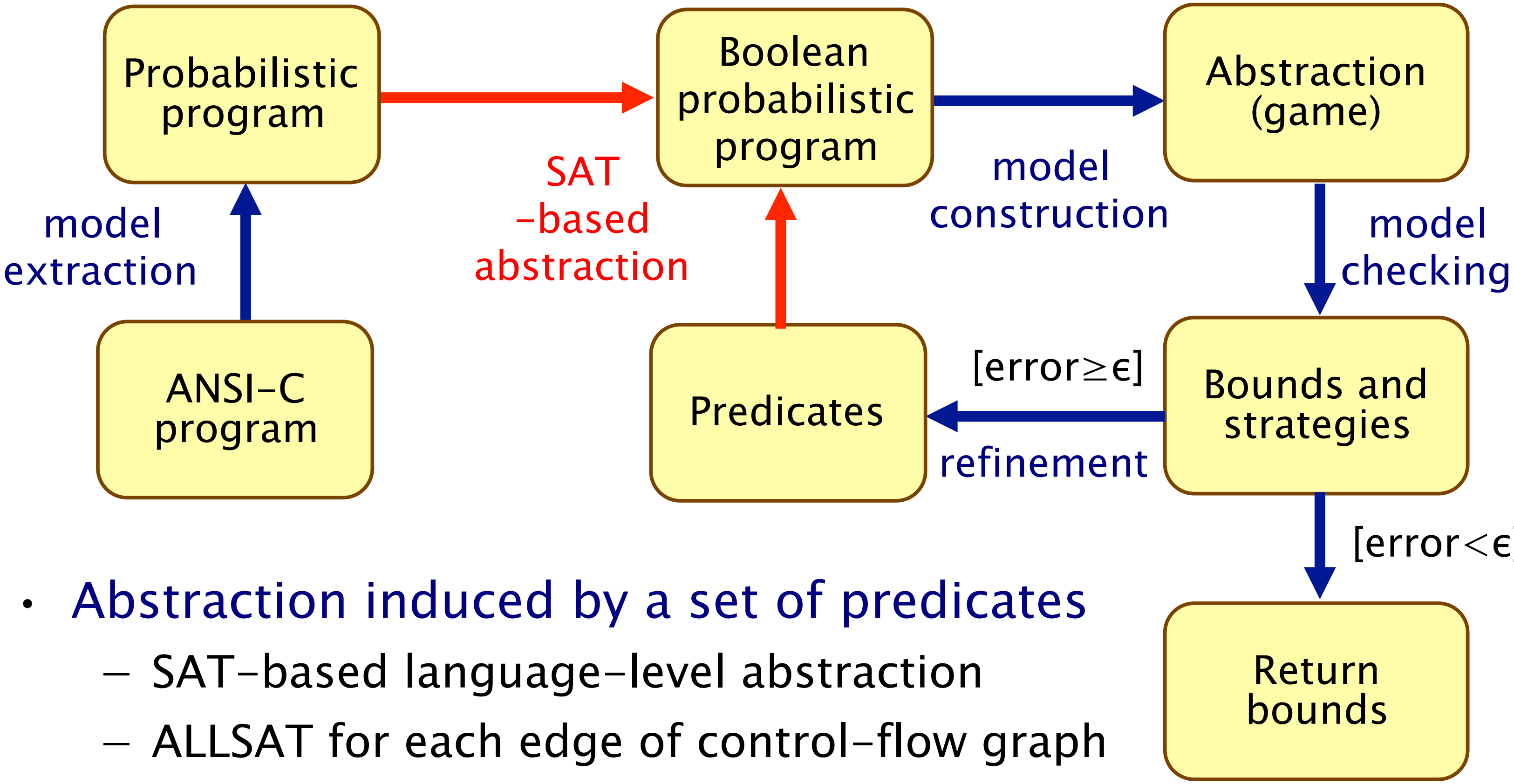


Property:

p_s^{min} (PC=5∧fail) = 0

p_s^{max} (PC=5∧fail) = 0.19

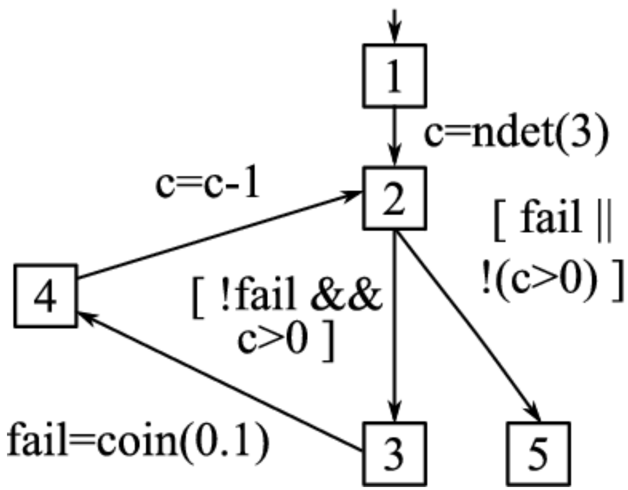
Abstraction-refinement loop



- Abstraction induced by a set of predicates
  - SAT-based language-level abstraction
  - ALLSAT for each edge of control-flow graph
  - implemented in extension of SATABS
- Boolean probabilistic program
  - (predicate) abstraction of probabilistic program
  - stochastic two player game semantics

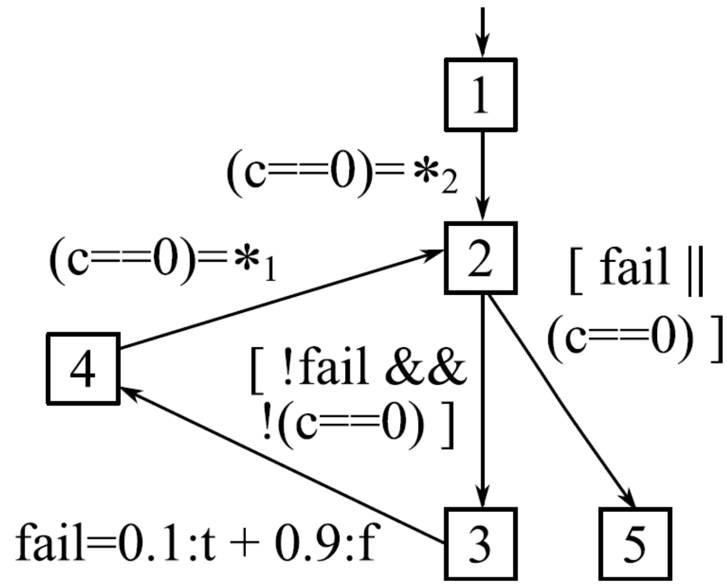
Back to example

Probabilistic program:



Boolean probabilistic program:

(2 predicates: fail, c==0)

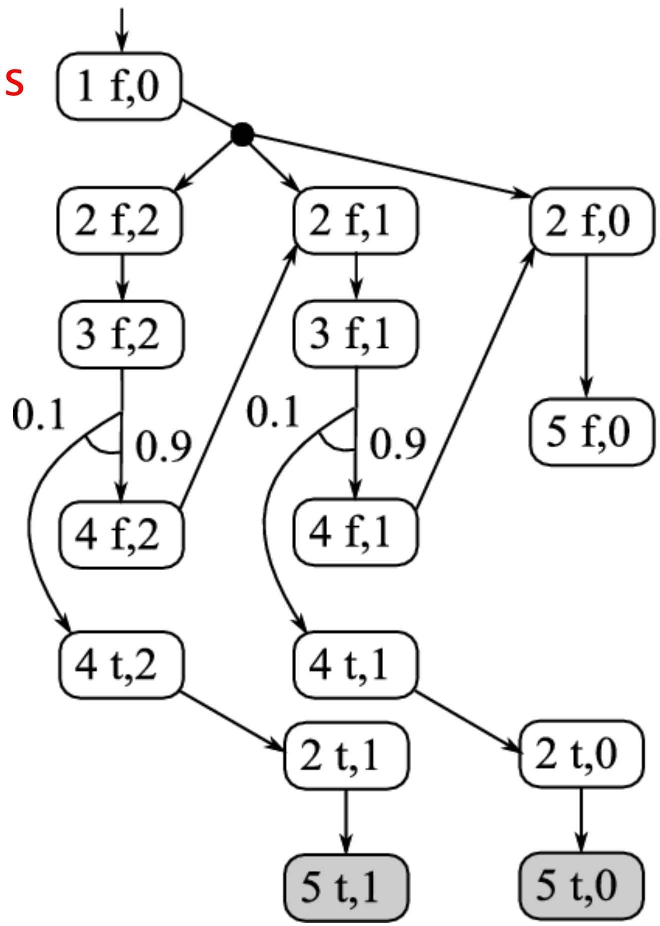


Key: $*_1$ Player 1 choice

$*_2$ Player 2 choice

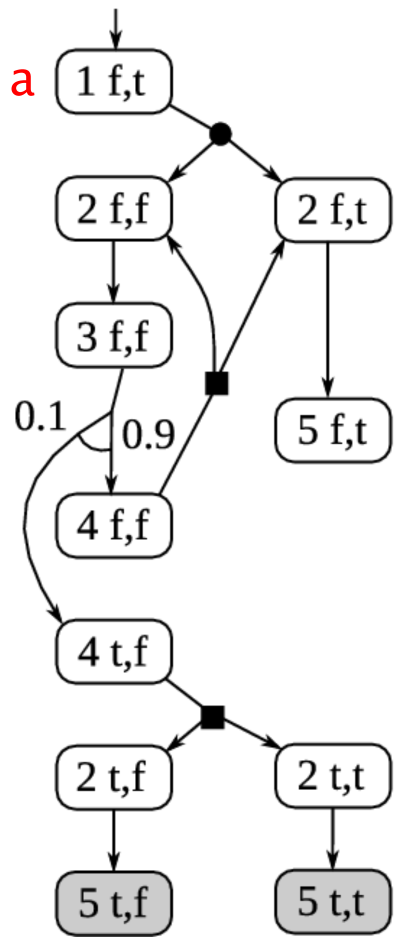
Back to example

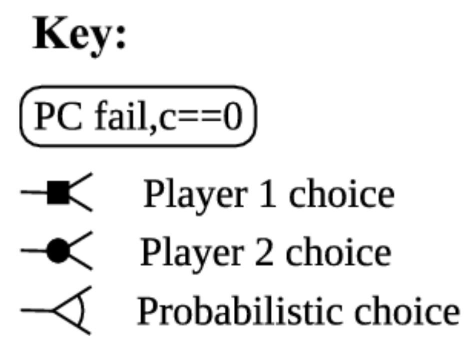
Concrete program (MDP):



$p_s^{max}(PC=5 \wedge fail) = 0.19$

Abstraction (game):

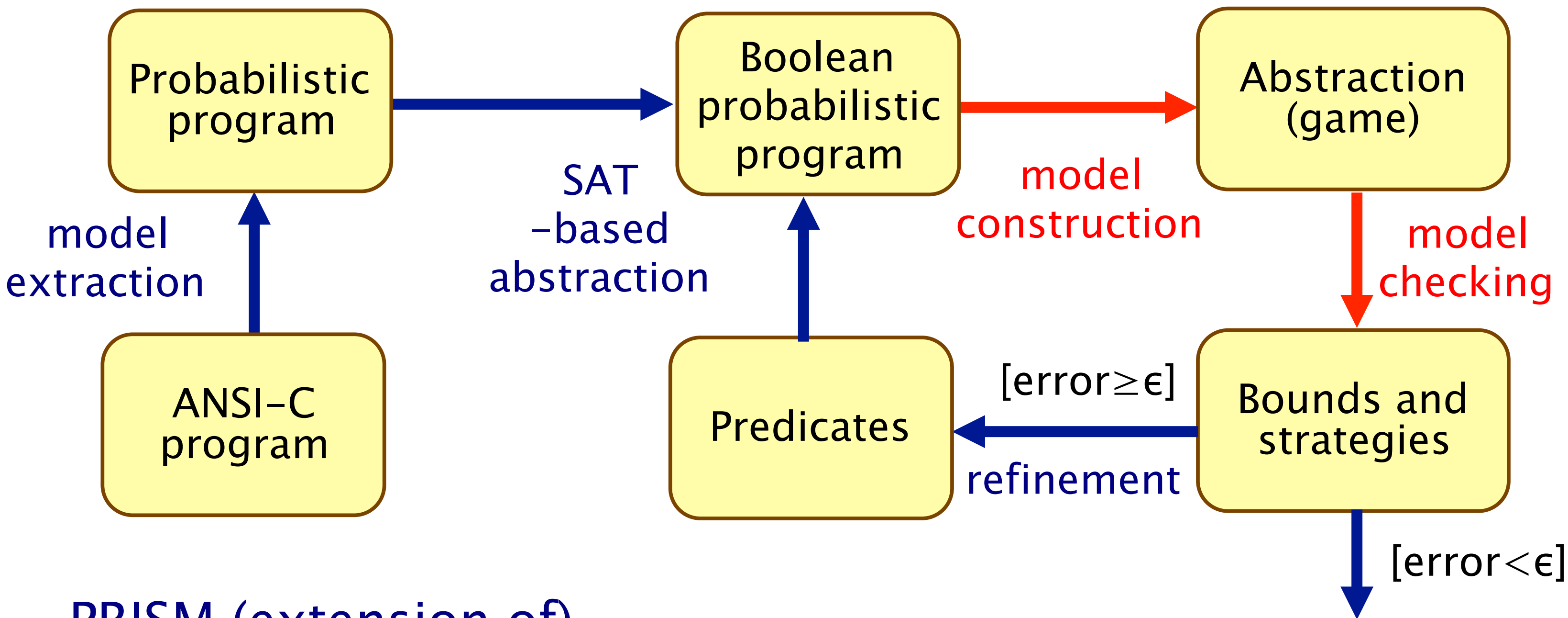


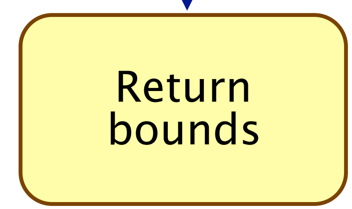


$p_a^{lb,max}(PC=5 \wedge fail) = 0.1$

$p_a^{ub,max}(PC=5 \wedge fail) = 1$

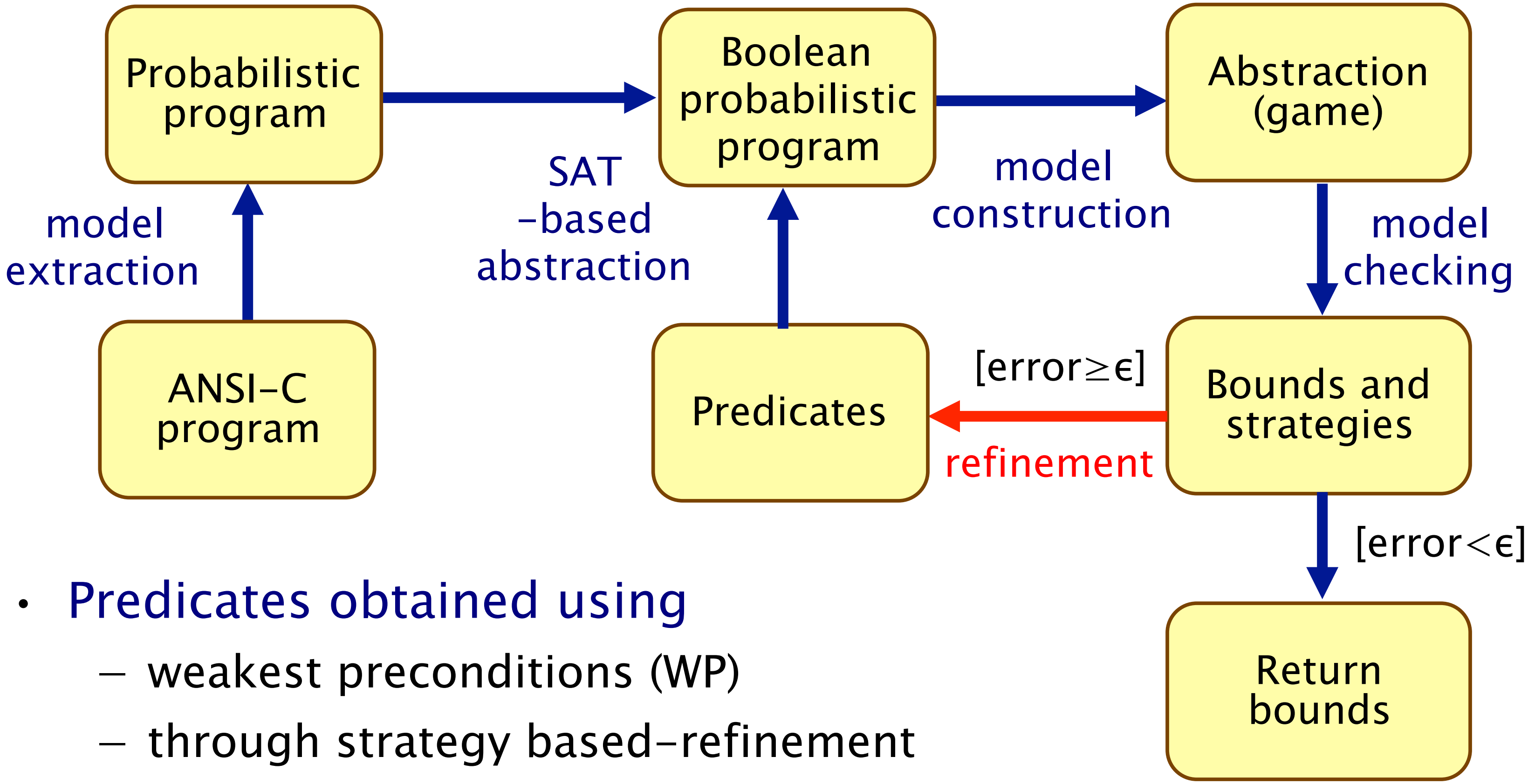
Abstraction-refinement loop





- PRISM (extension of)
  - adapted for verification of stochastic games
  - uses symbolic data structures (MTBDDs)
- Bounds and strategy
  - returned for a given probabilistic or expected reachability property

Abstraction-refinement loop



- Predicates obtained using
  - weakest preconditions (WP)
  - through strategy based-refinement
  - includes predicate localisation, reachability analysis, symbolic simulation,...

Experimental results

- Successfully applied to several Linux network utilities:
  - PING (tool for establishing network connectivity)
  - TFTP (file-transfer protocol client)
- Code characteristics
  - 1 KLOC of non-trivial ANSI-C code
  - Loss of packets modelled by probabilistic choice
  - Linux kernel calls modelled by nondeterministic choice
- Example properties
  - “maximum probability of establishing a write request”
  - “maximum expected amount of data that is sent before timeout”
  - “maximum expected number of echo requests required to establish connectivity”

Summary (Part 4)

- Abstraction: essential for large/infinite-state systems
  - this lecture: abstractions of MDPs as stochastic games
  - separation of nondeterminism from MDP/abstraction
  - yields lower/upper bounds on min/max probabilities
- Quantitative abstraction refinement
  - fully automatic generation of abstractions
  - iterative refinement based on quantitative measure of ‘error’
  - works well in practice…
- Quantitative software verification
  - ANSI-C + probabilistic behaviour
  - tool chain using state-of-the-art techniques and tools
- Next: probabilistic timed automata