

On Probabilistic Program Equivalence and Refinement

Andrzej S. Murawski

Joël Ouaknine

OXFORD UNIVERSITY COMPUTING LABORATORY

Algorithmic Game Semantics

- Exploit the concrete nature of game models to automatize reasoning about programs.
- Explicit characterizations of program equivalence and approximation in terms of respectively language equivalence and inclusion, e.g. Idealized Algol.
- Representations of game models: type-theoretic constraints lead to correspondences with various classes of automata.

This talk: extension to randomized computation.

Why Probabilistic Algorithms?

- Randomization can substantially improve the complexity of algorithms.
- Randomized algorithms turn out simpler than deterministic ones of comparable performance.

Three examples to illustrate the issues involved.

- Randomized Quicksort
- Dining Cryptographers
- Primality Tests

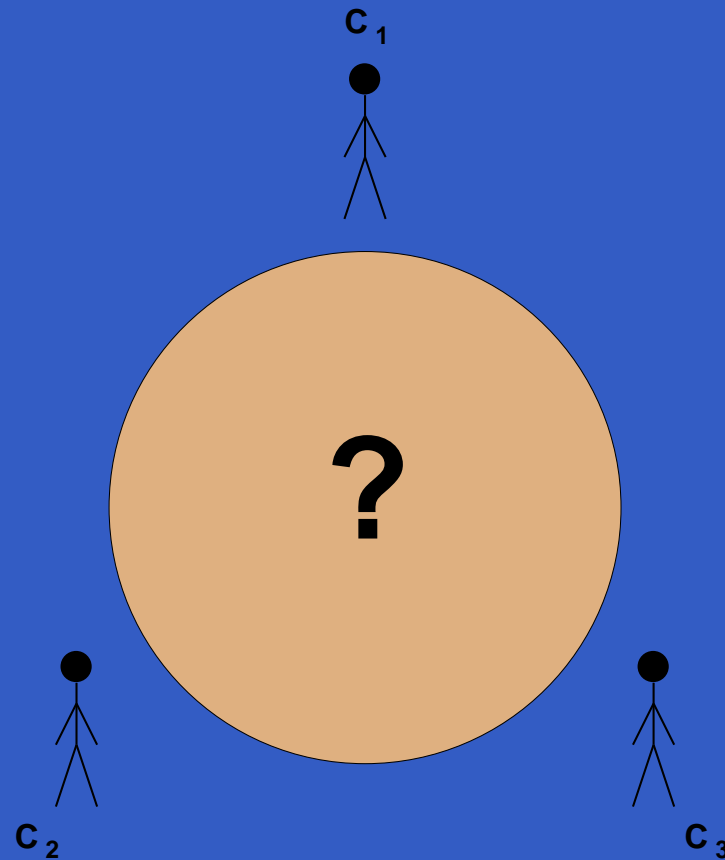
Randomized Quicksort (Las Vegas)

Randomization affects the runtime, but not the result.

10	4	18	<u>7</u>	9	13	6	15	2	5	8
4	6	2	5	<u>7</u>	8	10	18	9	13	15
2	4	5	6	<u>7</u>	8	9	10	13	15	18

- Randomization guarantees expected runtime of $O(n \log n)$. With very high probability the running time is not much more than the expectation.
- Thanks to random rearrangement the input is unlikely to be in one of the pathological orderings.

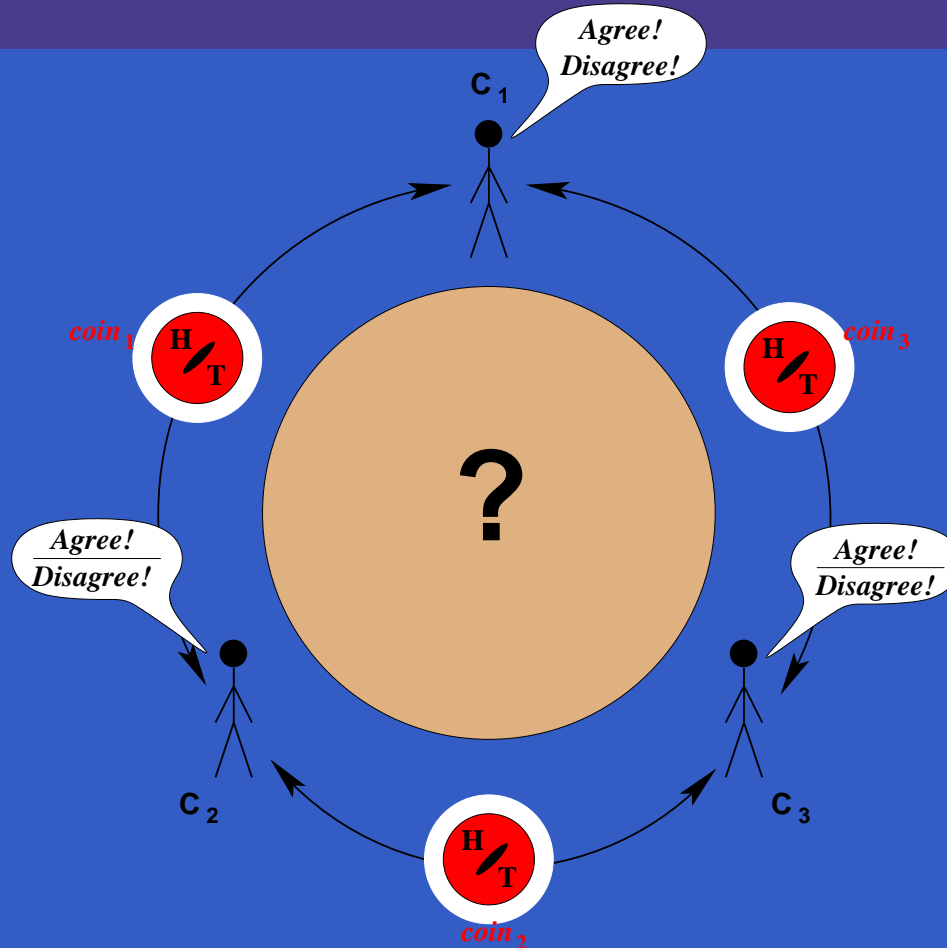
Dining Cryptographers



Who paid for the dinner? One of the diners or the NSA?

In the former case, the sponsor should remain anonymous.

The Randomized Solution



One of the cryptographers paid \iff # Disagree! is odd.

Fairness of coins guarantees anonymity.

Primality Testing (Monte Carlo)

Is n prime?

- Choose $a \in \{2, \dots, n - 1\}$ at random.
- If $a^{n-1} \equiv 1 \pmod{n}$ then PRIME else COMPOSITE.

Analysis:

- If n is prime, the algorithm always returns PRIME.
- If n is composite (and not a Carmichael number) the probability that the above algorithm returns PRIME is less than $\frac{1}{2}$.

Randomization gives high confidence in the answer.

Probabilistic Idealized Algol (Danos & Harmer)

$0, \dots, \text{max} : \text{exp}$ $\text{coin} : \text{exp}$ $\text{if } B \text{ then } M_1 \text{ else } M_0$
 $M := N$ $\text{while } B \text{ do } M$ $\lambda x.M$ MN

Can produce any rational probabilities.

Operational semantics (multiple evaluation trees $M \Downarrow_p V$):

$$\text{coin} \Downarrow_{\frac{1}{2}} 0 \qquad \text{coin} \Downarrow_{\frac{1}{2}} 1 \qquad \frac{B \Downarrow_p 1 \quad M \Downarrow_q V}{\text{if } B \text{ then } M_1 \text{ else } M_0 \Downarrow_{pq} V}$$

$$M \Downarrow_q V \quad \text{iff} \quad q = \sum_{M \Downarrow_p V} p$$

Probabilistic Equivalence

$\Gamma \vdash M_1, M_2$ are equivalent iff for all contexts $C[-]$ such that $C[M_1], C[M_2]$ are of ground type and for any V we have

$$C[M_1] \downarrow_p V \quad \text{iff} \quad C[M_2] \downarrow_p V.$$

Fully abstract game model (Danos & Harmer, LICS'00)

A strategy σ is a function $\sigma : \mathcal{L}^{\text{even}} \rightarrow [0, 1]$ satisfying:

- $\sigma(\epsilon) = 1$;
- for any $s \in \mathcal{L}^{\text{even}}, sa \in \mathcal{L}^{\text{odd}}$ we have

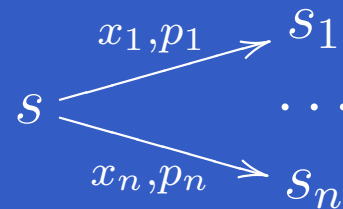
$$\sigma(s) \geq \sum_{sab \in \mathcal{L}^{\text{even}}} \sigma(sab).$$

Towards Applications

- Explicit characterization of program equivalence:

$$\forall_{s \in \mathcal{L}^{\text{comp}}} \llbracket M_1 \rrbracket(s) = \llbracket M_2 \rrbracket(s).$$

- Probabilistic automata



The probability of a given accepting run r

$$i \xrightarrow{x_1, p_1} s_1 \xrightarrow{x_2, p_2} \dots \xrightarrow{x_n, p_n} f$$

is $P(r) = p_1 \cdot \dots \cdot p_n$. Let $W(r) = x_1 \cdot \dots \cdot x_n$.

Equivalence of automata

Given an automaton \mathcal{A} define $\mathcal{A} : \Sigma^* \rightarrow [0, 1]$ by

$$\mathcal{A}(w) = \sum_{r, W(r)=w} P(r).$$

Two automata $\mathcal{A}_1, \mathcal{A}_2$ are equivalent if and only if

$$\forall w \in \Sigma^* \quad \mathcal{A}_1(w) = \mathcal{A}_2(w).$$

Probabilistic automata equivalence is decidable (Paz) in PTIME (Tzeng).

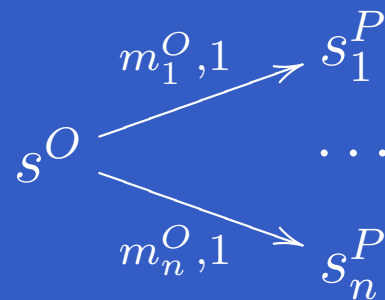
The PTIME algorithm is a nice application of linear algebra.

The Second-Order Fragment

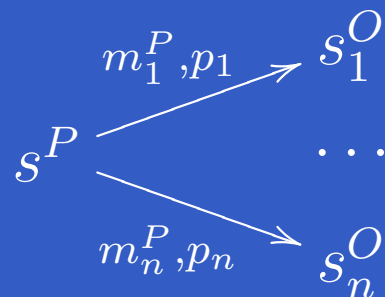
First-order procedures, also indeterminate ones.

Automata of a special kind: O-states s^O and P-states s^P .

- O-states



- P-states



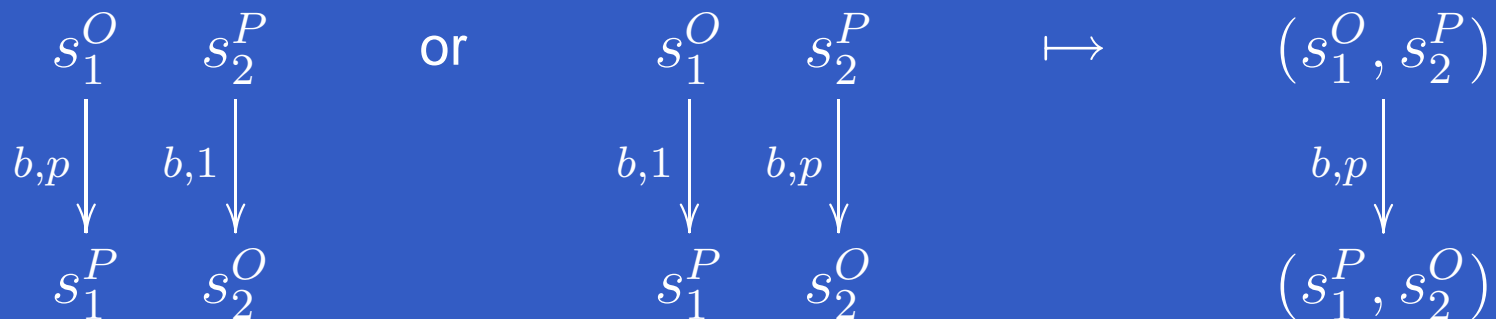
$$\sum_{i=1}^n p_i \leq 1$$

Modelling Composition with Automata

Composition Formula

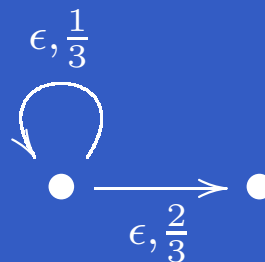
$$(\sigma; \tau)(s) = \sum_{u \in \text{wit}_B(s)} \sigma(u \upharpoonright A, B) \cdot \tau(u \upharpoonright B, C)$$

Synchronized Product of Automata



Hiding

- Isolate ϵ -transitions



- Compute the closure $\bullet \xrightarrow{\epsilon, 1} \bullet$ using the formula

$$c_{ij}^{(k)} = c_{ij}^{(k-1)} + c_{ik}^{(k-1)} \cdot \left(\sum_{i=0}^{\infty} (c_{kk}^{(k-1)})^i \right) \cdot c_{kj}^{(k-1)}.$$

Optimization: decompose into strongly connected components (Mohri)

Results

Second-order probabilistic program equivalence is decidable (in EXPTIME).

Applications:

- (Iterative) Randomized Quicksort = Insertion Sort (for arrays of a fixed size)
- The Dining Cryptographers protocol is correct.
- The Dining Cryptographers protocol guarantees anonymity, if one of the cryptographers has paid.
- Comparisons of distribution generators.

What can't we do?

THRESHOLD PROBLEM: given $\lambda > 0$ and a probabilistic automaton \mathcal{A} determine whether $\exists_w \mathcal{A}(w) > \lambda$.

Fundamental limitation: THRESHOLD PROBLEM is undecidable.

For example, define probabilistic refinement $M_1 \sqsubseteq M_2$ by

$$\text{if } C[M_1] \downarrow_p V \text{ then } C[M_2] \downarrow_q V \text{ and } q \geq p$$

for any $C[-]$ such that $C[M_1], C[M_2]$ are of ground type.

Then, using game semantics, one can show that \sqsubseteq is undecidable by a reduction from the THRESHOLD PROBLEM.

Future Work

Details: CONCUR'05

[http : //web.comlab.ox.ac.uk/oucl/work/andrzej.murawski/](http://web.comlab.ox.ac.uk/oucl/work/andrzej.murawski/)

Further plans

- Build a model checker.
- Try to address program approximation.