

Searching for Safety Violations using Estimation of Distribution Algorithms

Jan Staunton and John Clark

University of York, SEBASE EPSRC project

SBST 2010 Workshop @ ICST 2010, Paris
April 6, 2010

Presentation Outline

- Introduction to Model Checking
- EDA-based Model Checking
- Experimentation
- Summary

Introduction to Model Checking

Model Checking

- Proposed as a way of checking concurrent reactive systems*,**
- Found to be useful in checking a broad class of system properties or specifications

* E. Clarke and E. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In Workshop on Logics of Programs, pages 52–71. Springer, 1981.

** J. Queille and J. Sifakis. Specification and verification of concurrent systems in CESAR. In Proceedings of the 5th Colloquium on International Symposium on Programming, pages 337–351. Springer, 1982.

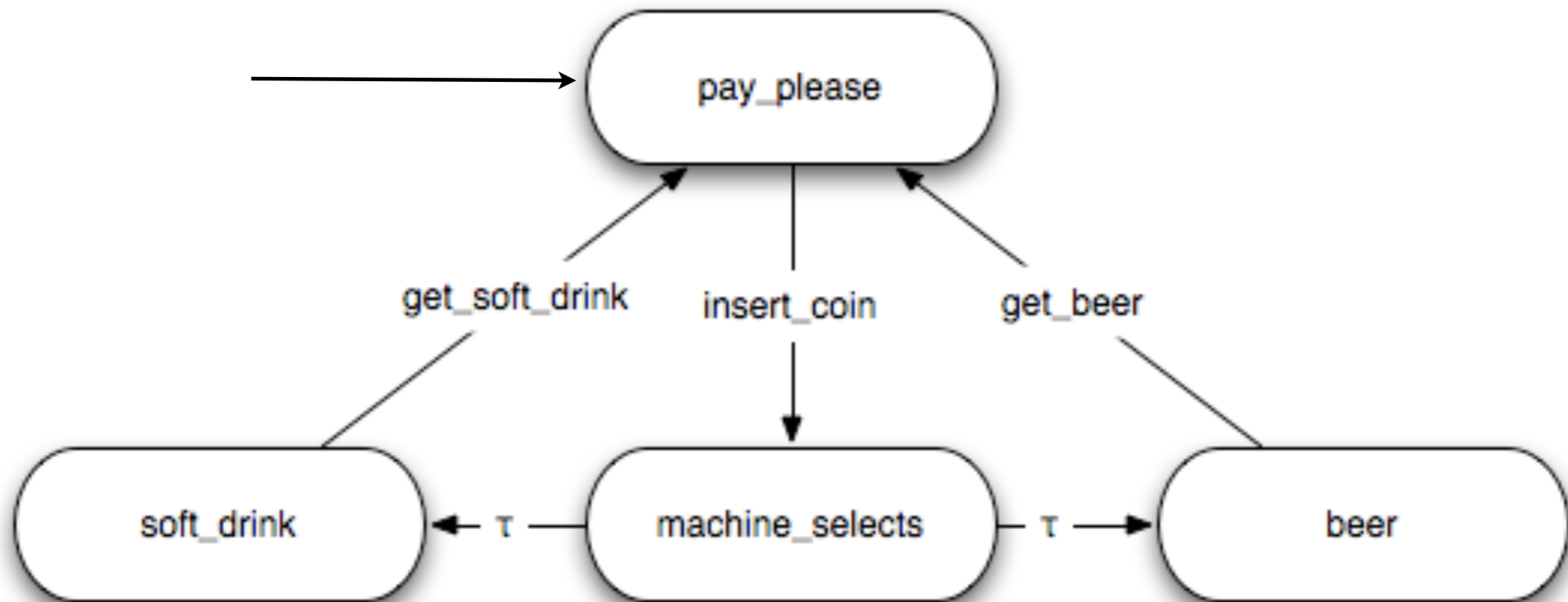
Basic concepts

- System is modelled as a state transition system
- Specifications are expressed in a propositional temporal logic
- Transition system is checked exhaustively to determine whether it is a model of the specification

* E. M. Clarke, O. Grumberg, and D.A. Peled. Model Checking. The MIT Press, January 2000.

Example

- A vending machine that, once a coin has been inserted, will return either a beer or a soft drink*



* C. Baier and J. Katoen. Principles of Model Checking. The MIT Press, 2008.

It has one major
drawback... (as well as
other minor ones)

sad face

State-space explosion!



* Public domain image, US Government

Problem we aim to solve

- Find faults in concurrent programs by focusing search on paths more likely to contain faults
- Specific focus on deadlock in Java programs
- Deadlock is a safety property, i.e. “System must not deadlock”

Previous work using metaheuristics

- Directed model-checking^{*} community use heuristics to help narrow down state space
- Alba et al.^{**} and Godefroid^{***} have applied GAs in this arena with some success
- Ant Colony Optimisation has also been tested^{****}

* S. Edelkamp, A.L. Lafuente, and S. Leue. Directed explicit model checking with HSF-SPIN. In Proceedings of the 8th international SPIN workshop on Model checking of software, pages 57–79. Springer-Verlag New York, Inc. New York, NY, USA, 2001.

** E. Alba, F. Chicano, M. Ferreira, and J. Gomez-Pulido. Finding deadlocks in large concurrent java programs using genetic algorithms. In Proceedings of the 10th annual conference on Genetic and evolutionary computation, pages 1735–1742. ACM New York, NY, USA, 2008.

*** P. Godefroid and S. Khurshid. Exploring very large state spaces using genetic algorithms. International Journal on Software Tools for Technology Transfer (STTT), 6(2):117–127, 2004.

**** E. Alba and F. Chicano. Finding safety errors with ACO. In Proceedings of the 9th annual conference on Genetic and evolutionary computation, pages 1066–1073. ACM Press New York, NY, USA, 2007.

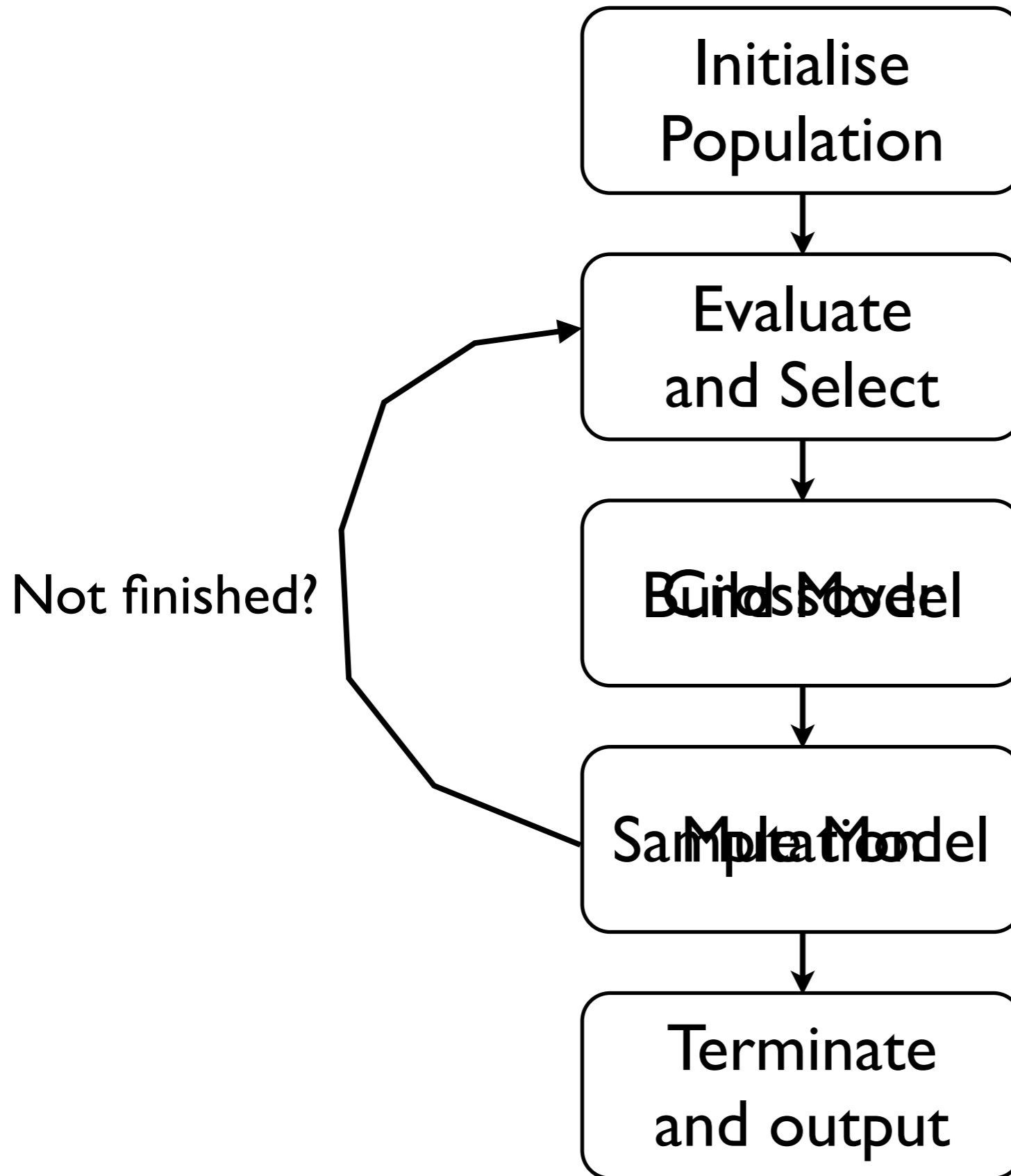
EDA-based Model Checking

Quick EDA Introduction

- Estimation of Distribution Algorithms (EDAs) are an effective metaheuristic search paradigm
- Similar to Genetic Algorithms (GAs)
- Replace the crossover and mutation stages with modelling and sampling stages
- Also known as Probabilistic Model Building Genetic Algorithms (PMBGAs)

* M. Pelikan, D.E. Goldberg, and F.G. Lobo. A survey of optimization by building and using probabilistic models. *Computational optimization and applications*, 21(1):5–20, 2002.

Estimation of Genetic Distances Algorithm



Encoding Paths

- In this work, a simple string representation is used
- Paths in a transition system can be viewed as a sequence of actions that cause transitions between states
- Sequences are represented as strings, alphabet is the set of possible actions in the transition system
- Alphabet is certainly modelling language specific, and may even be problem specific

Modelling Strings

- To model paths in the transition system, N-gram GP* is used
- N-gram GP models the joint probabilities of n-grams of length n
- An n-gram is a subsequence of characters of length n from a longer sequence

* R. Poli and N.F. McPhee. A linear estimation-of-distribution GP system. Lecture Notes in Computer Science, 4971:206–217, 2008.

Learning the Model (2-grams)

A selected string \longrightarrow B A B A B B

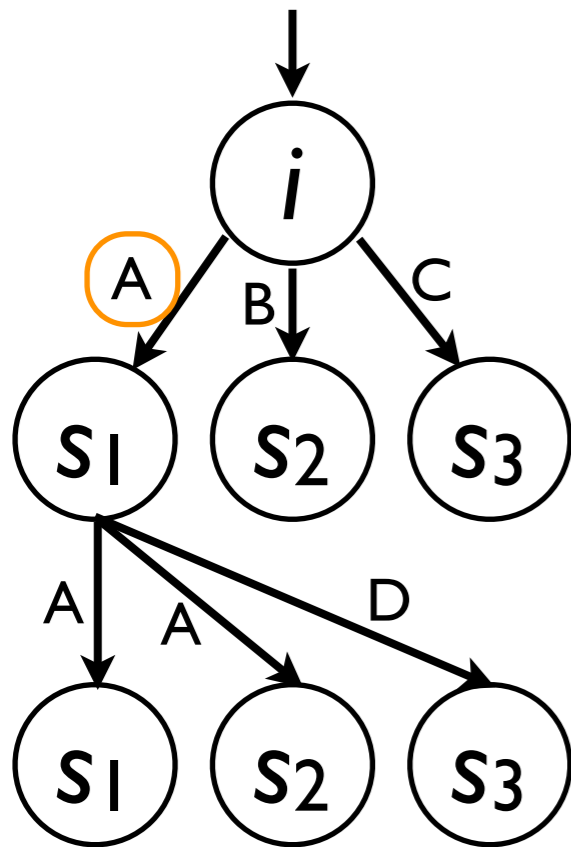
Model:

B A: B=2
A B: A=1, B=1

Normalised Model:

B A: B=1.0
A B: A=0.5, B=0.5

Sampling the Model



Path:

(I)

(I , A)

Model:

I : $A=0.8$, $B=0.15$, $C=0.05$

I A : $A=0.3$, $D=0.7$

A few special cases

Case 1:

Excess actions in model
for given n-gram

Solution:

Cull actions from model
(for this choice only)
and normalise

Case 2:

Some actions from state
are not “described” by
model

Solution:

Set the probabilities of
“non-described” actions
to some suitably low
value

Last special case

- Last special case is when the n-gram was not observed at all in the solutions from which the model was learned
- In this case, a “blank” distribution is constructed, with each action having an equal probability of being chosen

Fitness Function

Require: A, B are Individuals;
if $A.error_found \neq B.error_found$ **then**
 return IndividualWithErrorFound(A,B);
else if $A.error_found$ **and** $B.error_found$ **then**
 return IndividualWithShortestPath(A,B);
else
 return IndividualWithHighestBlockMetric(A,B);
end if

IndividualWithHighestBlockMetric(A,B):

Require: I is an Individual;
aggregateBlocked = 0;
for all $States\ s \in I.Path$ **do**
 aggregateBlocked += s.NumberOfBlockedThreads;
end for

Other details

- Truncation selection used
- Elitism used, one individual copied from old to new population
- Mutation operator implemented

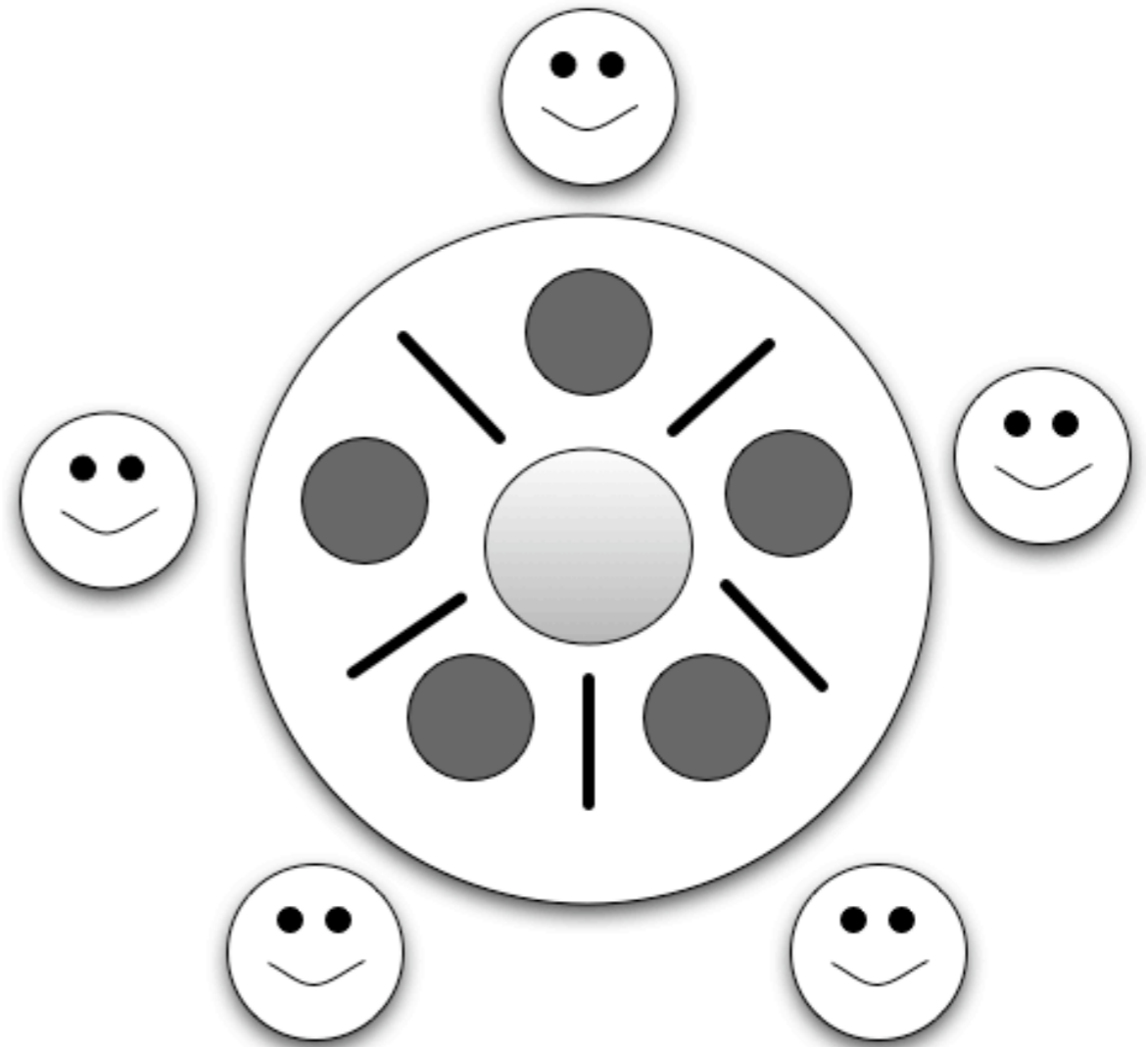
Experimentation

Experimentation

- Implemented technique using Java PathFinder and the ECJ toolkit
- To test the technique, we used the Dining Philosophers coordination problem
- Naive coordination policy of philosophers tested, known to JPF as “DiningPhil”
- Compared algorithm against Depth-first search, breadth-first search and random search

Dining Philosophers Problem

```
public void run() {  
    // think!  
    synchronized (left) {  
        synchronized (right) {  
            // eat!  
        }  
    }  
}
```



Example Path and Alphabet

```
1.<null transition>
2.<[synthetic] [clinit]<clinit>><[synthetic] [clinit]<clinit>><invokeclinit>
3.<java/lang/ThreadGroup.java:859><(java/lang/ThreadGroup.java:859)><monitorexit>
4.<examples/DiningPhil.java:38><      synchronized (left) {}><runstart>
5.<examples/DiningPhil.java:38><      synchronized (left) {}><getfield>
6.<examples/DiningPhil.java:38><      synchronized (left) {}><monitorenter>
7.<examples/DiningPhil.java:33><      start();><invokevirtual>
8.<java/lang/ThreadGroup.java:844><(java/lang/ThreadGroup.java:844)><monitorenter>
9.<examples/DiningPhil.java:39><      synchronized (right) {}><getfield>
10.<examples/DiningPhil.java:39><      synchronized (right) {}><monitorenter>
11.<examples/DiningPhil.java:41><      }><monitorexit>
12.<java/lang/ThreadGroup.java:859><(java/lang/ThreadGroup.java:859)><monitorexit>
13.<examples/DiningPhil.java:38><      synchronized (left) {}><runstart>
14.<examples/DiningPhil.java:42><      }><monitorexit>
15.<examples/DiningPhil.java:33><      start();><invokevirtual>
16.<examples/DiningPhil.java:38><      synchronized (left) {}><getfield>
17.<examples/DiningPhil.java:41><      }><monitorexit>
18.<examples/DiningPhil.java:42><      }><monitorexit>
```

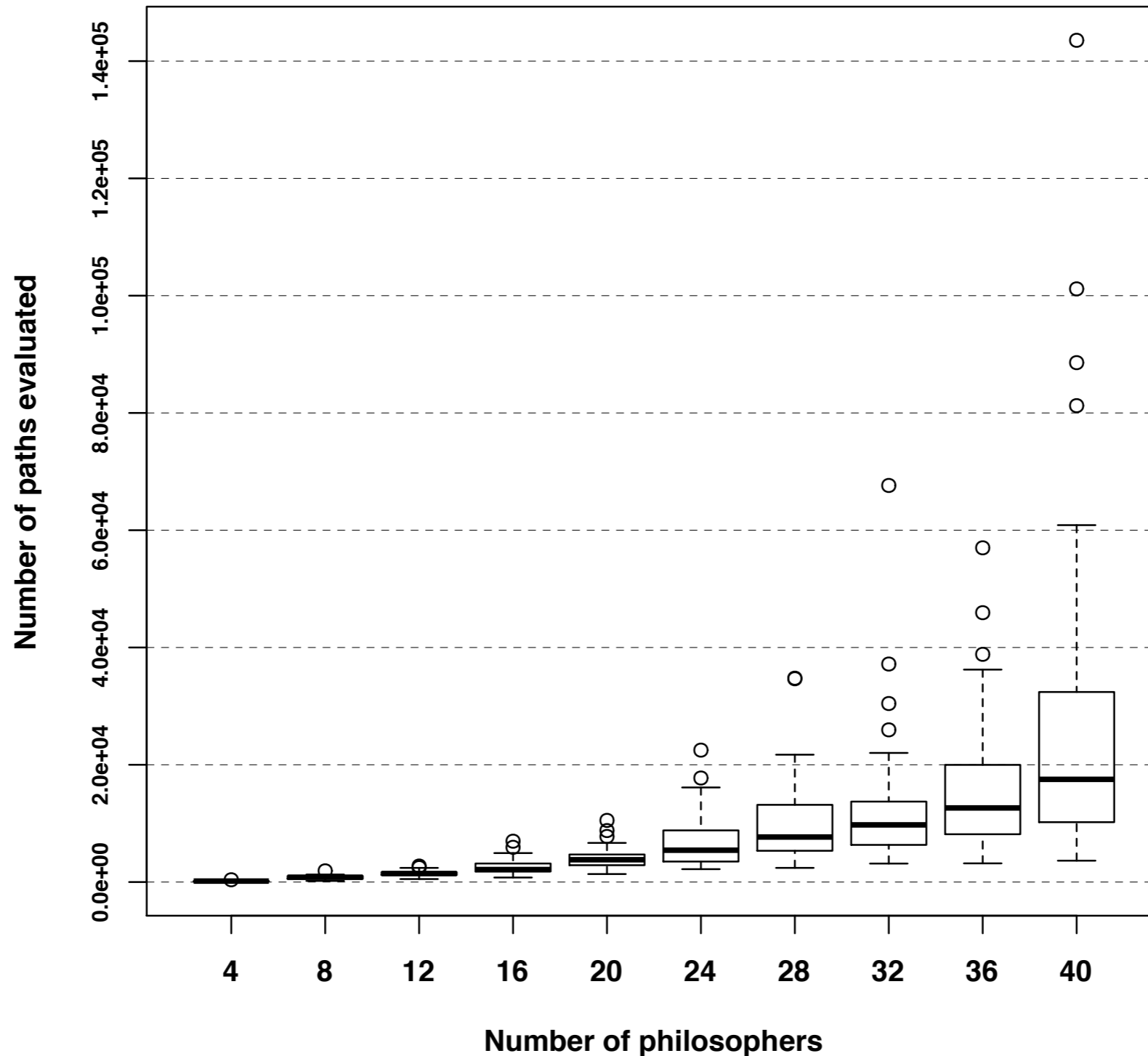
Parameters

- 50 independent executions of EDA and random search technique
- Population size: 150
- N-gram size: 3
- Truncation selection used, best 20% of individuals used to build model
- Mutation probability set to 0.001
- Elitism set at 1
- Entire population is replaced with newly sampled individuals
- Initial population generated randomly
- Termination criterion: Solution is found

Problem Size		DFS	BFS	RS	EDA
4	Errors / runs	1/1	1/1	50/50	50/50
	Mean Time (s)	(+)0.491	(+)3.971	(-)8.277	7.356
	Min. Time (s)	0.491	3.971	3.453	4.175
	Max. Time (s)	0.491	3.971	18.165	11.933
	Mean Max. Mem. Usage (MB)	(-)482	(+)80	(+)972	843.4
	Mean States Visited	(-)312	(-)13029	(+)13510	5028
	Mean Paths Evaluated	-	-	(+)385.3	143.32
	Mean Generations	-	-	-	0.74
8	Errors / runs	1/1	0/1	0/50	50/50
	Mean Time (s)	(-)25.191	-	-	23.303
	Min. Time (s)	25.191	-	-	11.428
	Max. Time (s)	25.191	-	-	40.859
	Mean Max. Mem. Usage (MB)	(-)496	-	-	1244
	Mean States Visited	(+)238264	-	-	57470
	Mean Paths Evaluated	-	-	-	811.6
	Mean Generations	-	-	-	4.96
12	Errors / runs	1/1	0/1	0/50	50/50
	Mean Time (s)	(+)16375.392 (4h32m55s)	-	-	54.778
	Min. Time (s)	16375.392 (4h32m55s)	-	-	24.420
	Max. Time (s)	16375.392 (4h32m55s)	-	-	107.376
	Mean Max. Mem. Usage (MB)	(-)3233	-	-	2177
	Mean States Visited	(+)150841824	-	-	158134
	Mean Paths Evaluated	-	-	-	1498
	Mean Generations	-	-	-	9.64
16	Errors / runs	0/1	0/1	0/50	50/50
	Mean Time (s)	-	-	-	123.012
	Min. Time (s)	-	-	-	38.498
	Max. Time (s)	-	-	-	299.005
	Mean Max. Mem. Usage (MB)	-	-	-	3049
	Mean States Visited	-	-	-	357997
	Mean Paths Evaluated	-	-	-	2549
	Mean Generations	-	-	-	16.66

Potential to scale...

Number of paths evaluated against number of philosophers



A brief comment about problem families

- It was noted that the models for the 4 philosopher problem was mighty similar to the 40 philosopher problem
- It seems plausible (this is not yet validated) that the model learned during the exploration of the four philosopher problem can be exploited to solve larger instances
- Family-based approaches are typical when model checking hardware designs
- Model could yield important insights into the system under test

Summary

Summary

- Presented an effective, extensible and intuitive way of searching transition systems
- Very general technique, can be applied wherever model checking is applicable
- Has the potential to yield insight into a problem by analysing the learned models

Future Plans

- Looking at commercial software
- Plans to look at liveness properties using SPIN
- Application to large systems, targeting an operating system

Thanks!

Any questions?