Hardness for Explicit State Software Model Checking Benchmarks

Neha Rungta and Eric G. Mercer

Software Model Checking Lab

Computer Science Department

Brigham Young University

Provo, UT - 84602

Acknowledgements

Ira & Mary Lou Fulton for the Fulton Supercomputing Lab at BYU

Marylou4, cluster of 618 nodes
Among top 50 supercomputers

Acknowledgements

Matt Dwyer at UNL
Suzette Person at UNL

Shmuel Ur at
IBM Research Lab, Haifa

Introduction

- Concurrent systems have become ubiquitous
- Growing complexity challenges ad-hoc testing methods
- Vector simulation finds bugs in the early design phase
- Code coverage techniques are not feasible
- Low-level scheduling decisions create concurrency errors
- Motivates a need for a formal approach to find these errors

Simple program

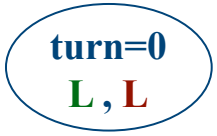
P_a
a_0: while *True* do
a_1: wait (turn = 0)
a_2: turn = 1
end while

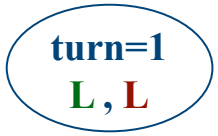
P_b:
b_0: while *True* do
b_1: wait (turn =1)
b_2: turn = 0
end while

Simple program

```
Pa
 a0:  while True do
 a1:          wait (turn = 0)
 a2:          turn = 1
       end while


Pb:
b0: while True do
b1:          wait (turn =1)
b2:          turn = 0
     end while
```

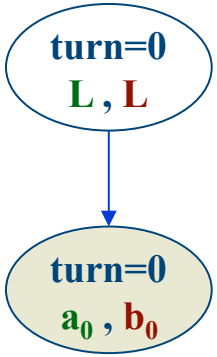


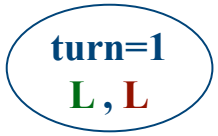


Simple program

```
Pa
 a0:  while True do
 a1:          wait (turn = 0)
 a2:          turn = 1
       end while

Pb:
b0: while True do
b1:          wait (turn =1)
b2:          turn = 0
     end while
```

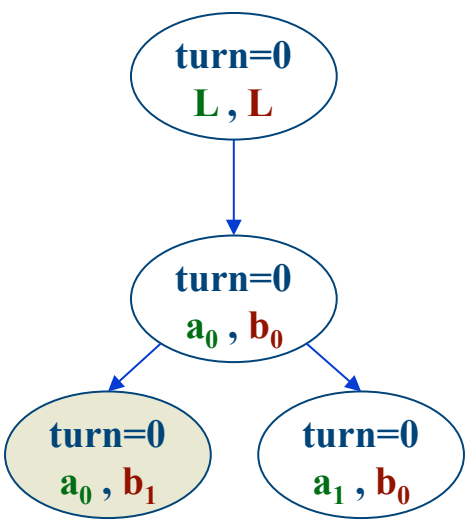




Simple program

```
Pa
a0:  while True do
a1:          wait (turn = 0)
a2:          turn = 1
       end while

Pb:
b0: while True do
b1:          wait (turn =1)
b2:          turn = 0
     end while
```

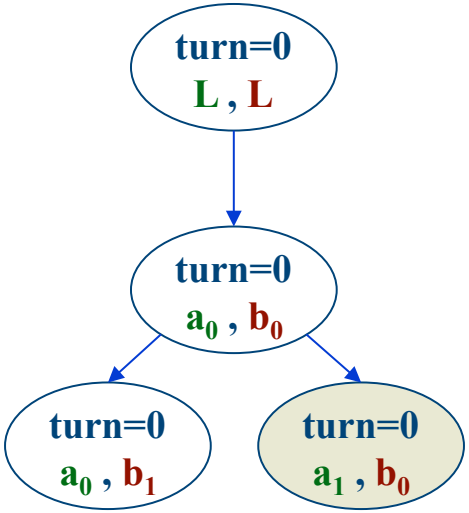


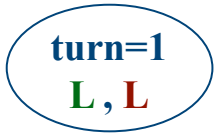
turn=1
L , L

Simple program

```
Pa
 a0:  while True do
 a1:          wait (turn = 0)
 a2:          turn = 1
       end while

Pb:
b0: while True do
b1:          wait (turn =1)
b2:          turn = 0
     end while
```

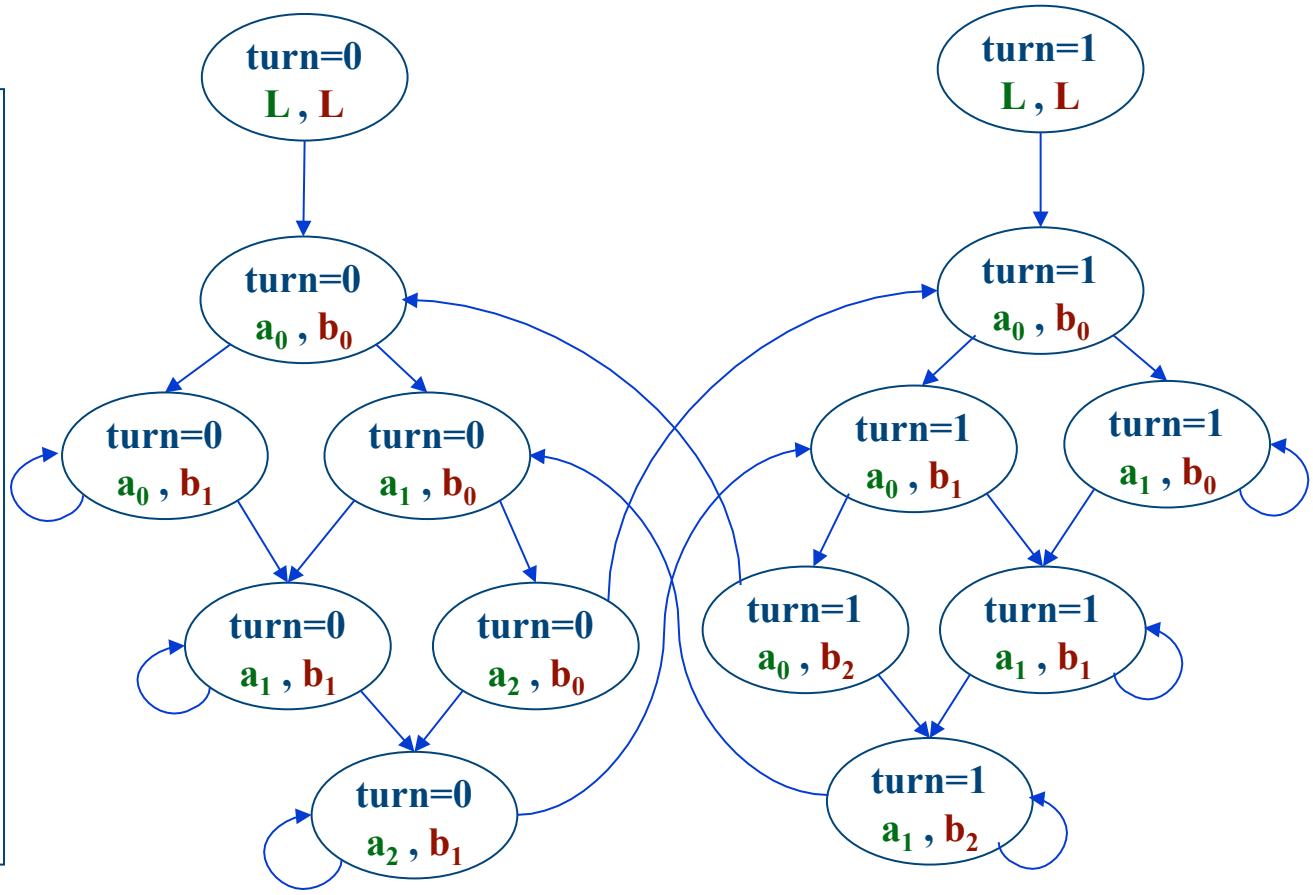




Simple program

P_a
a_0: while *True* do
a_1: wait (turn = 0)
a_2: turn = 1
end while

P_b:
b_0: while *True* do
b_1: wait (turn =1)
b_2: turn = 0
end while



Techniques

- Predicate Abstraction
- Guided Model Checking
- Distributed / Parallel Model Checking
- Partial Order Reduction

Developing New Algorithms

- In-lab Testing
- Compare different algorithms
- Requires a set of benchmarks
- Look at set compiled by Matt Dwyer et al. in FSE 2006

Benchmarks

Subject	Source	Error	Classes	SLOC
Account	IBM	Deadlock/Race	3	66
Accountsubtype	IBM	Race	6	91
Airline	IBM	Race	2	31
AllocateVector	IBM	No Lock	3	85
LinkedList	IBM	Atomicity	5	117
Piper	IBM	Deadlock	2	71

Real Programs

Subject	Source	Error	Classes	SLOC
AlarmClock	Bandera	Null Pointer Exception	6	12
BoundedBuffer	Bandera	Deadlock	5	65
Daisy	other	Assertion Violation	21	744
DEOS	JPF	Assertion Violation	24	838
Elevator	other	ArrayIdxOOBExcpn	12	934
ReplicatedWorker	Bandera	Deadlock	14	304
RaxExtended	JPF	Race	11	127
ReaderWriter	Bandera	Deadlock, Race	6	103

Kernel Examples

Subject	Source	Error	Classes	SLOC
Clean	CBP	Deadlock	4	51
Deadlock	Bandera	Deadlock	2	24
DiningPhil	Bandera	Deadlock	3	25
LoseNotify	CBP	Deadlock	4	41
NestedMonitor	Bandera	Deadlock	6	53
ProducerConsumer	Bandera	Race	8	87
Reorder	CBP	Atomicity	4	44
SleepingBarber	Bandera	Deadlock	4	66
TwoStage	CBP	Two-stage Access	5	52
WrongLock	CBP	Wrong lock	4	38

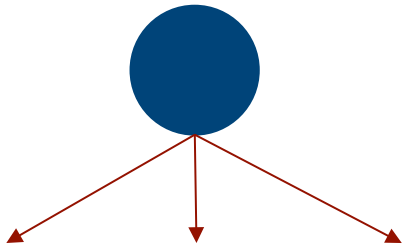
Quality of benchmarks

- Work by Dwyer et al. in FSE 06
- Characterizes hardness of models
- Probability of encountering an error
- Metric is known as path error density
- Compute the value using random walk

Random Walk

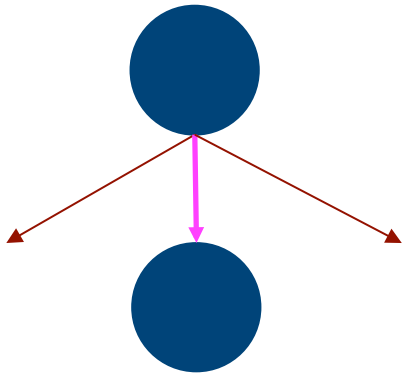
- **Start the search from the start state**

Random Walk



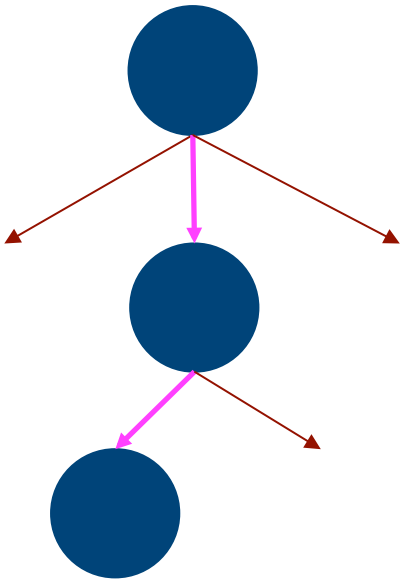
- **Consider all the enabled transitions at the state**

Random Walk



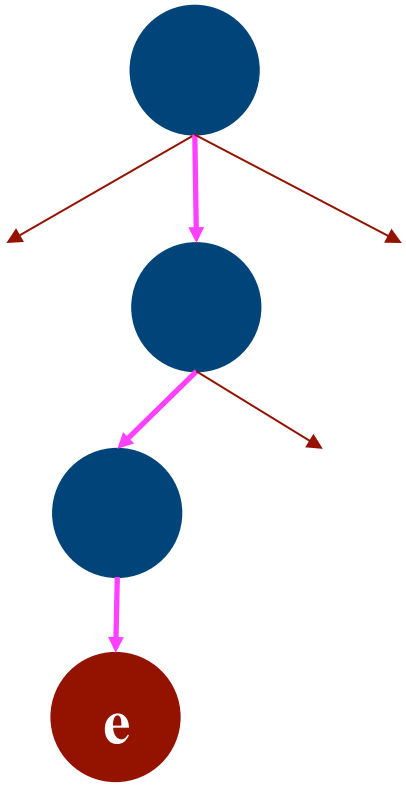
- **Randomly pick a transition to explore**
- **No knowledge of previously visited states**

Random Walk



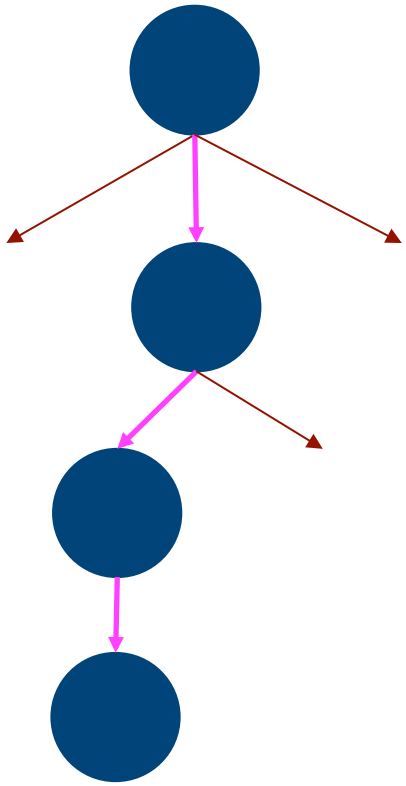
- **Again, randomly pick a transition to explore**

Random Walk



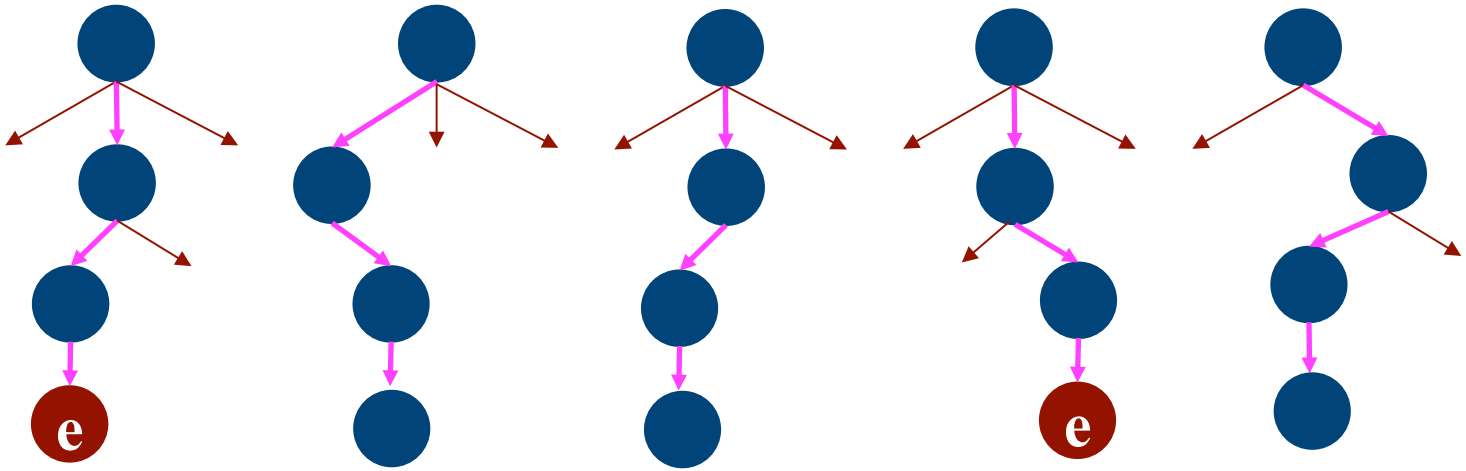
- **Stop the search when an error is encountered**

Random Walk



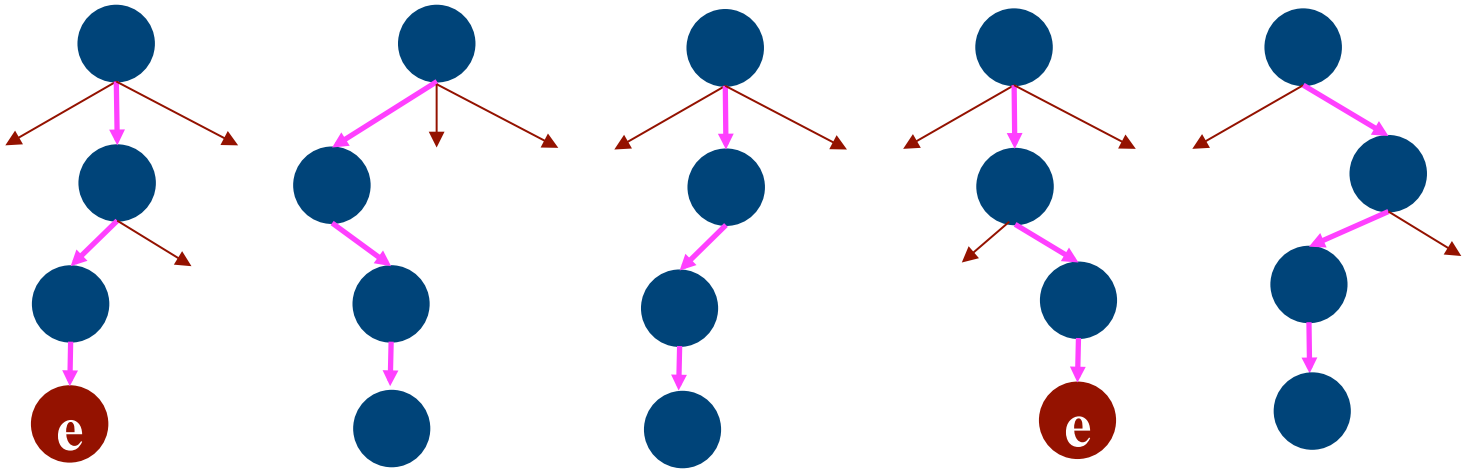
- **Also stop the search if a state with no successors**

Path Error density



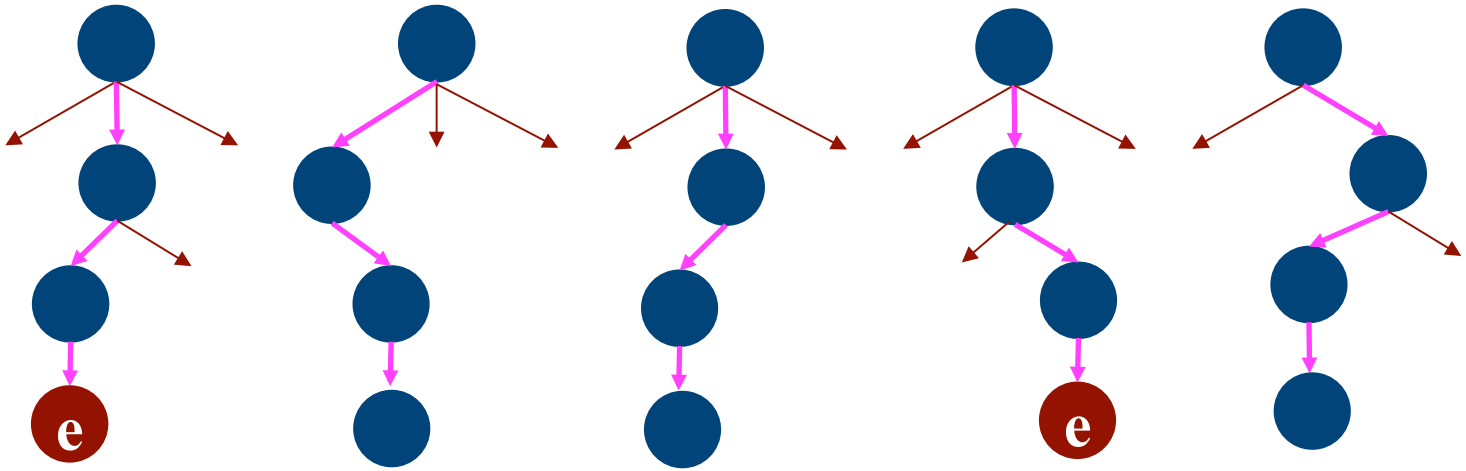
- **Large number of random walk experiments**

Path Error density



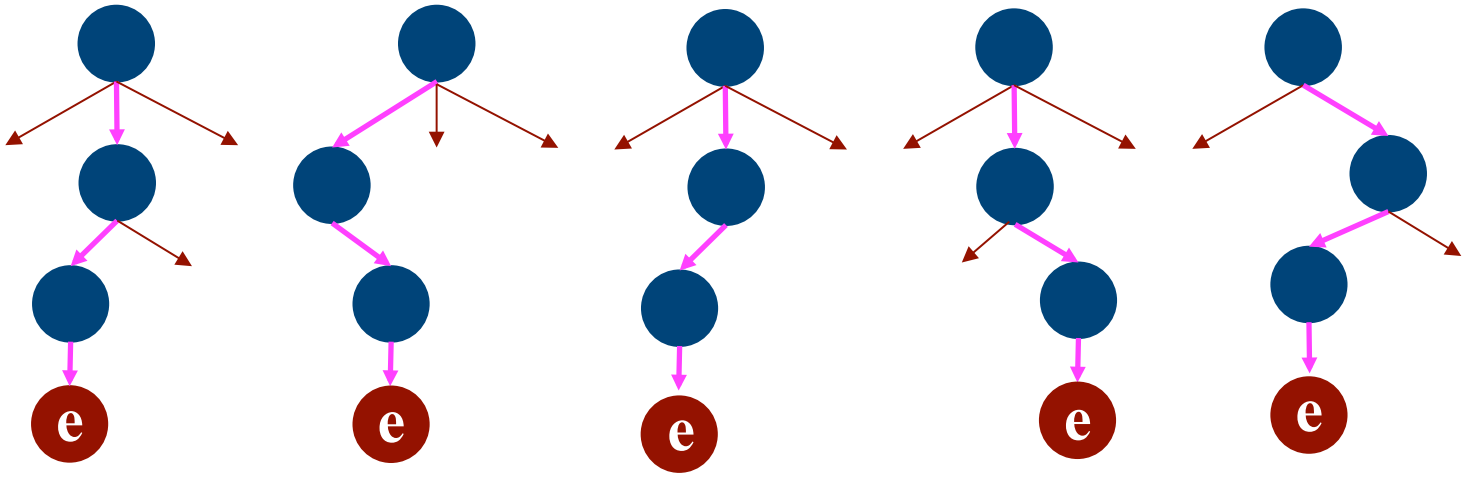
- **Large number of random walk experiments**
- **Error discovering runs over total runs is hardness**

Path Error density



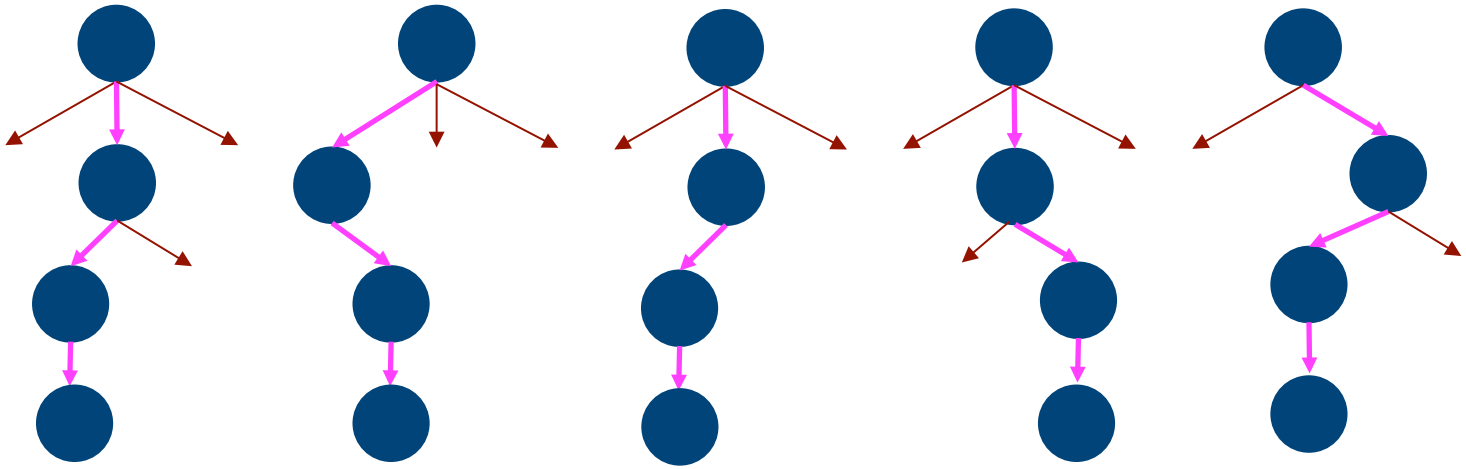
- **Large number of random walk experiments**
- **Error discovering runs over total runs is hardness**
- **For this example : 2 / 5 = 0.40**

Easy Model



- **All explored paths have an error**
- **Path error density is 1.00**

Hard Model



- **No error found along explored paths**
- **Path error density is 0.00**

Easy Subjects (Dwyer et al. FSE '06)

Subject	Density	Subject	Density
NestedMonitor()	1.000	LoseNotify(1,12,18)	1.000
LoseNotify(10,10,120)	1.000	LinkedListNoSync(4,100)	1.000
DiningPhil(12,6)	1.000	BoundedBuffer(2,10,11)	0.996
DiningPhil(12,8)	1.000	AccountSubtype(5,5)	0.996
SleepingBarber()	1.000	Clean(10,10,120)	0.994
DiningPhil(12,10)	1.000	Clean(2,2,24)	0.958

Average Models (Dwyer et al. FSE '06)

Subject	Density	Subject	Density
AlarmClock(4)	0.232	AlarmClock(9)	0.227
AllocateVector(8,20,1)	0.750	AllocateVector(2,20,4)	0.575
AllocateVector(2,20,1)	0.204	Deadlock(1)	0.754
Deadlock(2)	0.638	DEOS(true)	0.415
Wronglock(1,1)	0.406	Account-NoExcepChk()	0.114
Account-NoDeadlkChk()	0.663	AccountSubtype(8,1)	0.526
Airline(6,1)	0.657	Airline(6,2)	0.836
ProducerConsumer(2,2,4)	0.146	ProducerConsumer(2,4,4)	0.294
ReplicatedWorker(5,2,0,10.7,.05)	0.261	ReplicatedWorkers(8,2,.0,10,.001)	0.703
RWNoDeadLkCk(2,2,100)	0.495	RWNoExcepChk(2,2,100)	0.433
RaxExtended(2,3)	0.761	RaxExtended(4,3)	0.794
Airline(20,2)	0.757	Airline(20,8)	0.809
Clean(10,10,1)	0.477	Piper(2,8,4)	0.336
Piper(2,16,8)	0.118	ProducerConsumer(2,8,4)	0.244
Wronglock(1,10)	0.712	Wronglock(10,1)	0.642

Hard models (Dwyer et al. FSE '06)

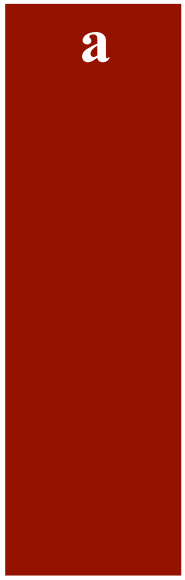
Subject	Density	Subject	Density
Piper(2,4,4)	0.073	Reorder(1,1)	0.002
AllocateVector(2,100,1)	0.038	Daisy()	0.0007
BoundedBuffer(2,4,4,1)	0.027	DEOS(false)	0.000
TwoStage(2,5)	0.019	Reorder(1,5)	0.000
TwoStage(1,1)	0.016	Elevator()	0.000
TwoStage(2,2)	0.012	LinkedListSyn(4,100)	0.000
Clean(1,1,12)	0.009	BoundedBuffer(3,6,6,1)	0.000

PRSS

(Dwyer et al. ICSE '07)

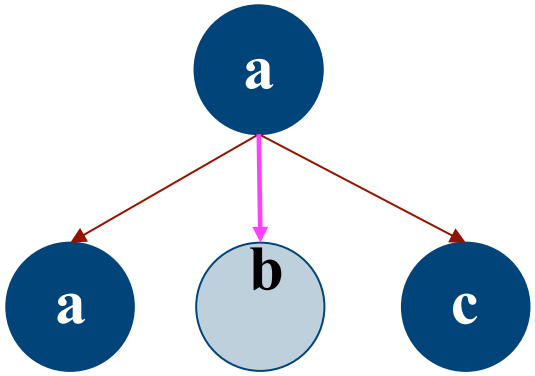
- Parallel Random Depth-first search
- Search technique to find errors
- Evaluated on 7 hard to semi-hard models
- Effective in error discovery

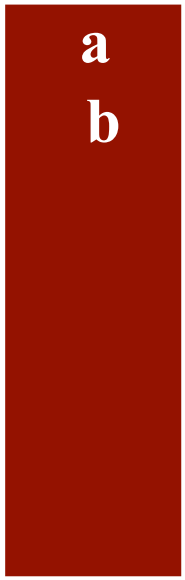
Random DFS



Visited States

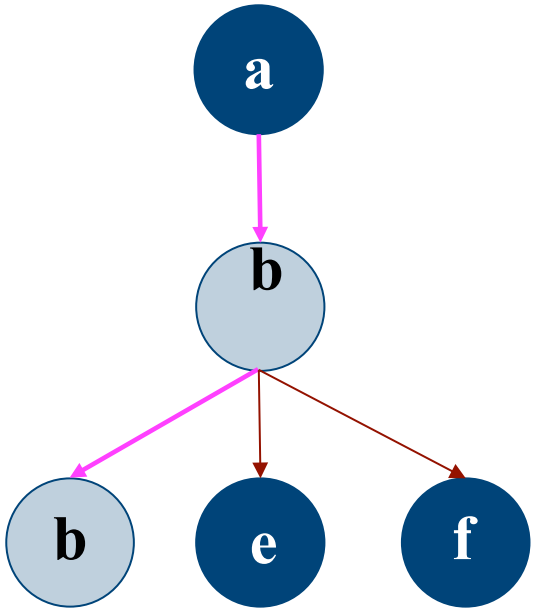
Random DFS

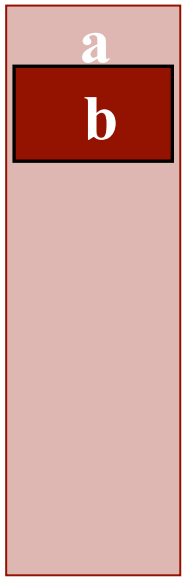




Visited States

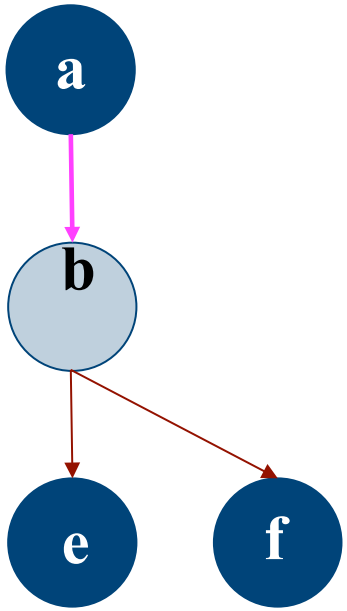
Random DFS

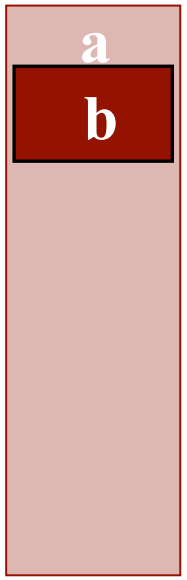




Visited States

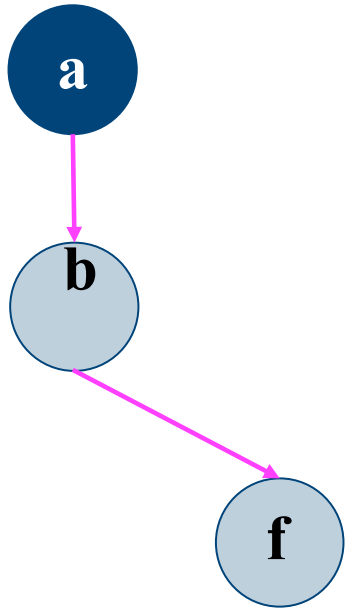
Random DFS

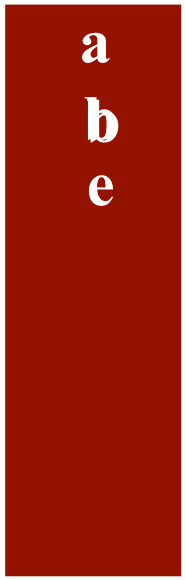




Visited States

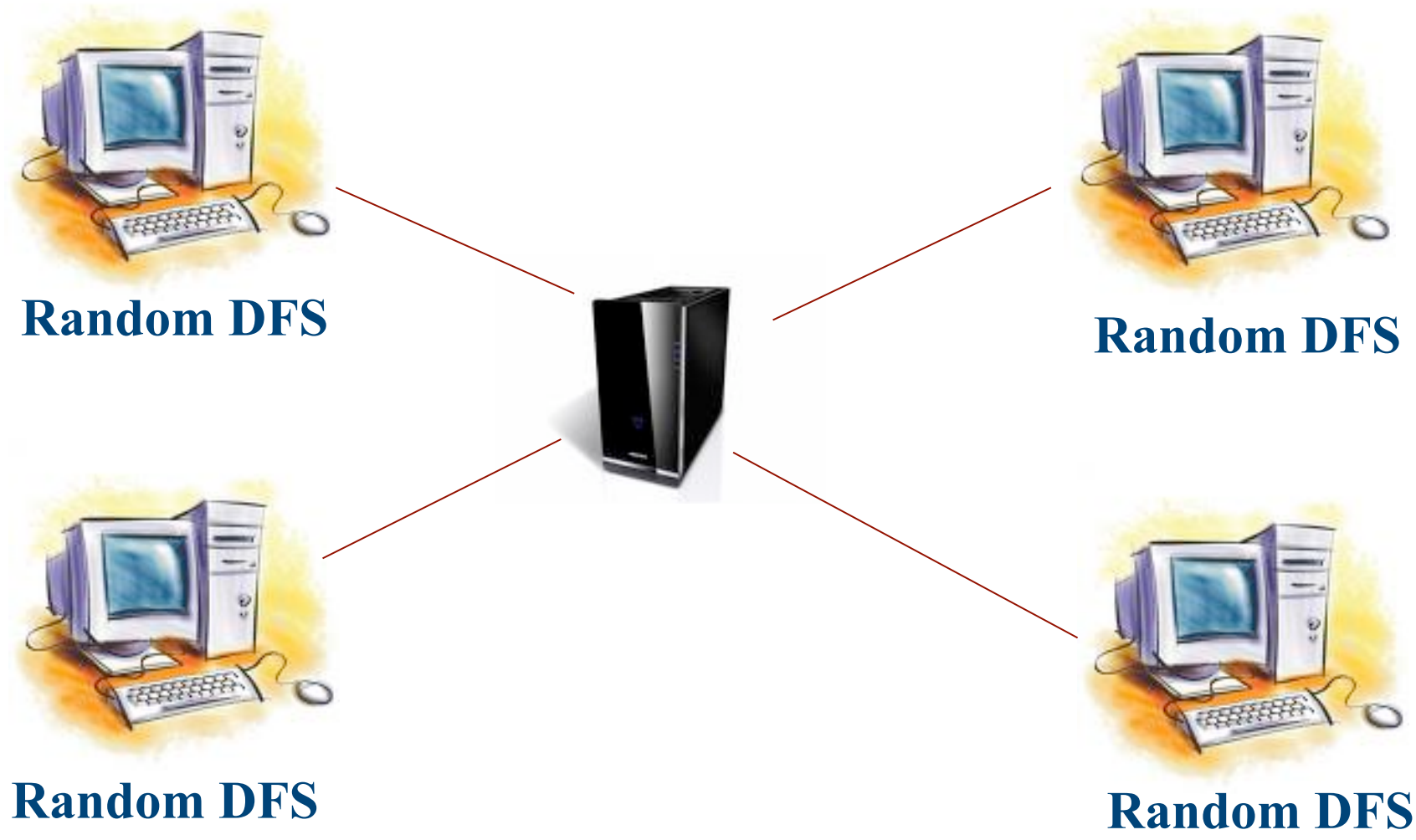
Random DFS





Visited States

PRSS



Images courtsey: blogs.zdnet.com, libcoop.com

PRSS

Random DFS
1 hour bound

Random DFS
1 hour bound

Random DFS
1 hour bound

Random DFS
1 hour bound

Images courtsey: blogs.zdnet.com, libcoop.com

PRSS

Random DFS
1 hour bound

Random DFS
1 hour bound

Found Error

Random DFS
1 hour bound

Random DFS
1 hour bound

Images courtsey: blogs.zdnet.com, libcoop.com

PRSS

(Dwyer et al. ICSE '07)

- Run on *N* parallel nodes
- One node (out of *N*) guaranteed to find an error in the model
- Computes *N* for seven models
- Models have fairly low path error density

PRSS Results (Dwyer et al. ICSE '07)

Subject	Nodes - N	Mean States
BoundedBuffer(3,6,6,1)	20	313,290
Daisy()	5	37,715
DEOS(false)	15	143,860
Elevator()	10	116,960
RaxExtended(4,3,false)	5	176
ReplicatedWorkers(5,2,0.0,10.7,0.05)	15	226,790
RWNoDeadLckCk(2,2,100)	10	1,847

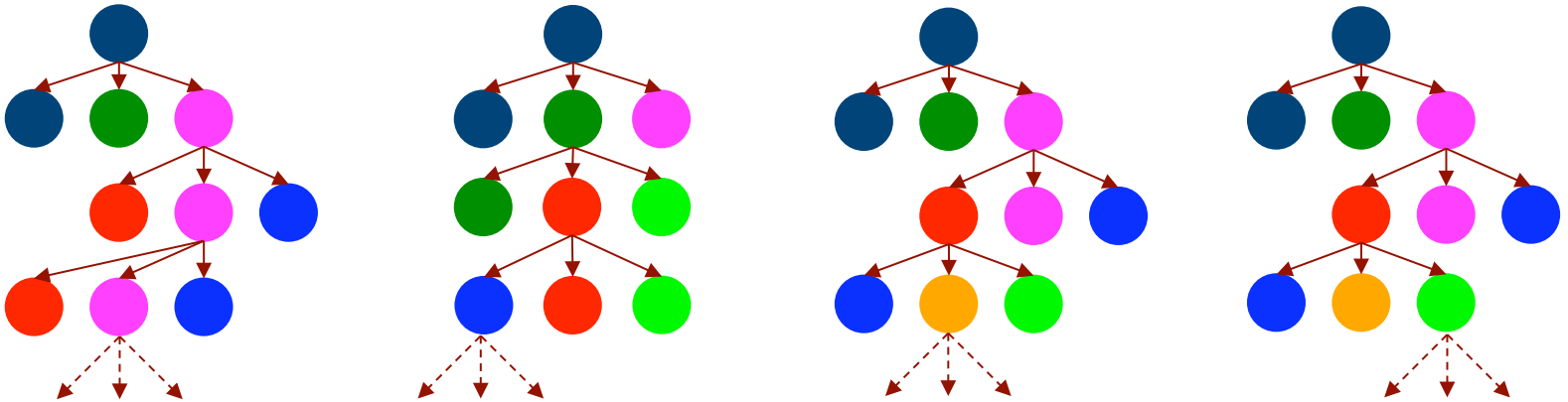
Contradictory

- Models categorized hard with stateless random walk
- Easily discoverable errors with stateful random DFS
- Is it wise to compare compare stateless and stateful techniques for a semantic metric?
- Need a stateful hardness measure

Tighter Bound

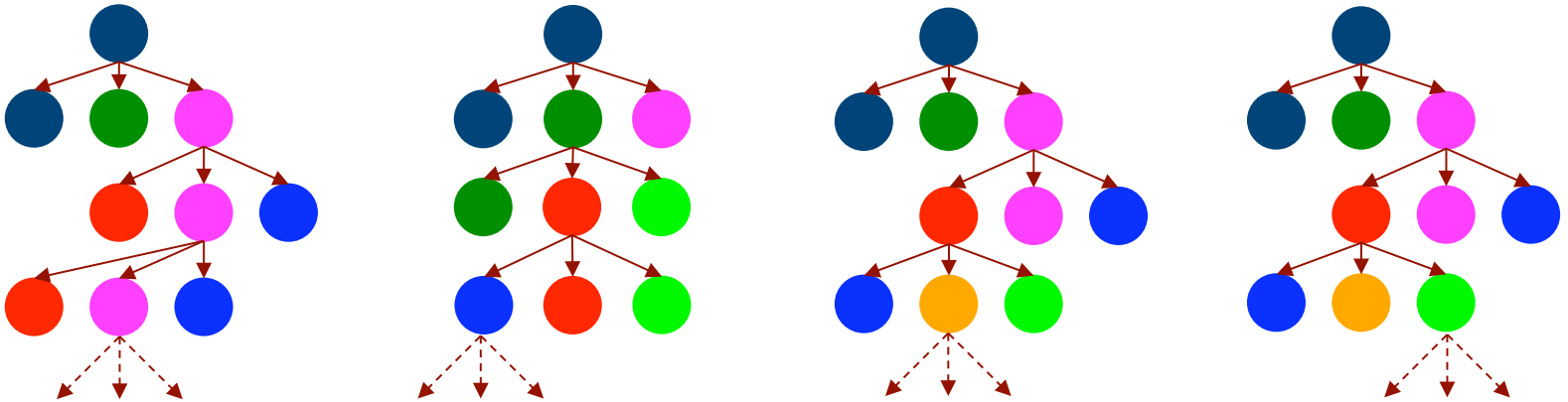
- Use Random DFS as hardness measure
- Basic stateful search technique
- Provides a tighter hardness bound
- Other stateful techniques need to overcome it

R-DFS Error Density



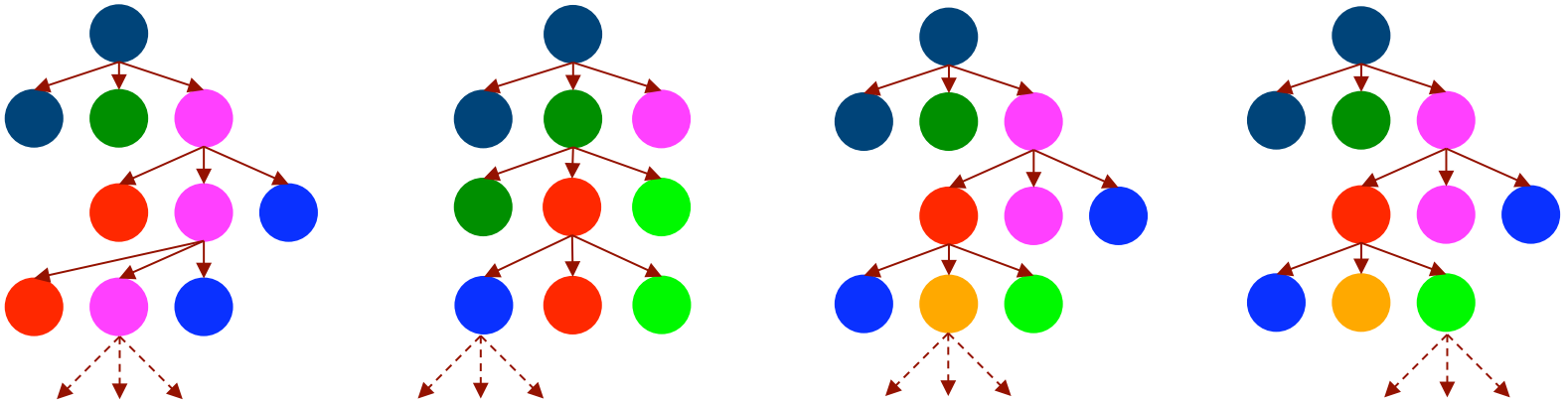
- **100 Random DFS trials with 1 hour time bound**

R-DFS Error Density



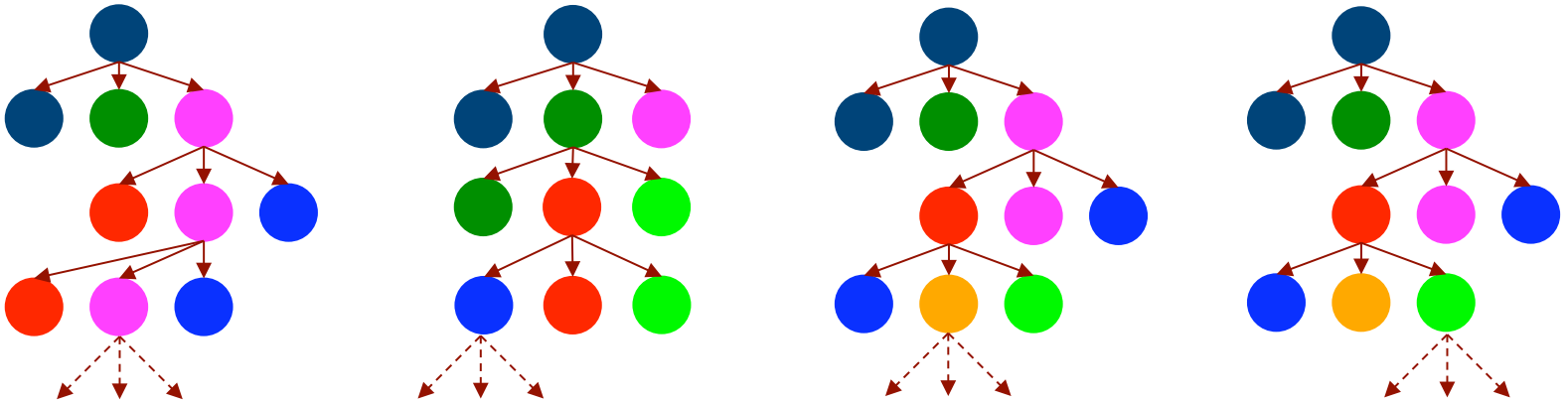
- **100 Random DFS trials with 1 hour time bound**
- **Record the error discovery trials**

R-DFS Error Density



- **100 Random DFS trials with 1 hour time bound**
- **Record the error discovery trials**
- **Ratio of error discovering trials over total trials**

R-DFS Error Density



- 100 Random DFS trials with 1 hour time bound
- Record the error discovery trials
- Ratio of error discovering trials over total trials
- 1.00 indicates easy while 0.00 is a hard model

R-DFS error density

Subject	Density	Subject	Density
Account-NoDeadlkChk(6)	1.00	ProducerConsumer(2,4,4)	1.00
Airline(6,2)	1.00	ProducerConsumer(2,2,4)	1.00
Airline(6,1)	1.00	RaxExtended(2,3)	1.00
AlarmClock(9)	1.00	Reorder(1,1)	1.00
AlarmClock(4)	1.00	Reorder(1,5)	1.00
AllocateVector(2,20,4)	1.00	TwoStage(1,1)	1.00
AllocateVector(2,20,1)	1.00	TwoStage(2,5)	1.00
AllocateVector(2,100,1)	1.00	TwoStage(2,2)	1.00
Clean(1,1,12)	1.00	Wronglock(10,1)	1.00
Deadlock(1)	1.00	Wronglock(1,10)	1.00
Deadlock(2)	1.00	Wronglock(1,1)	1.00
DEOS(true)	1.00	AllocateVector(8,20,1)	0.99
LinkedListSync(4,100)	1.00	Piper(2,8,4)	0.98
Piper(2,4,4)	1.00	ReplicatedWorkers(8,2,0,10,.001)	0.97
ProducerConsumer(2,8,4)	1.00	Clean(10,10,1)	0.96

R-DFS error density

Subject	Density	Subject	Density
Account-NoDeadlkChk(6)	1.00	ProducerConsumer(2,4,4)	1.00
Airline(6,2)	1.00	ProducerConsumer(2,2,4)	1.00
Airline(6,1)	1.00	RaxExtended(2,3)	1.00
AlarmClock(9)	1.00	Reorder(1,1)	1.00
AlarmClock(4)	1.00	Reorder(1,5)	1.00
AllocateVector(2,20,4)	1.00	TwoStage(1,1)	1.00
AllocateVector(2,20,1)	1.00	TwoStage(2,5)	1.00
AllocateVector(2,100,1)	1.00	TwoStage(2,2)	1.00
Clean(1,1,12)	1.00	Wronglock(10,1)	1.00
Deadlock(1)	1.00	Wronglock(1,10)	1.00
Deadlock(2)	1.00	Wronglock(1,1)	1.00
DEOS(true)	1.00	AllocateVector(8,20,1)	0.99
LinkedListSync(4,100)	1.00	Piper(2,8,4)	0.98
Piper(2,4,4)	1.00	ReplicatedWorkers(8,2,0,10,.001)	0.97
ProducerConsumer(2,8,4)	1.00	Clean(10,10,1)	0.96

R-DFS error density

Subject	Density	Subject	Density
RWNoExcepChk(2,2,100)	0.80	Account-NoExcpChk()	0.48
Airline(20,8)	0.49	AccountSubtype(8,1)	0.34

R-DFS error density

Subject	Density	Subject	Density
Airline(20,2)	0.01		

Effect of the time bound

- R-DFS error density depends on time bound
- Study the effect of the time on error density

Results

	R-DFS Error Density		
Subject	1 hour	2 hour	3 hour
Airline(20,2)	0.01	0.00	0.00
Reorder(9,1)	0.06	0.45	0.37
Twostage(7,1)	0.41	0.69	0.93
Twostage(8,1)	0.04	0.03	0.07
TwoStage(10,1)	0.00	0.00	0.00
Wronglock(10,1)	0.18	0.20	0.20

Results

	R-DFS Error Density		
Subject	1 hour	2 hour	3 hour
Airline(20,2)	0.01	0.00	0.00
Reorder(9,1)	0.06	0.45	0.37
Twostage(7,1)	0.41	0.69	0.93
Twostage(8,1)	0.04	0.03	0.07
TwoStage(10,1)	0.00	0.00	0.00
Wronglock(10,1)	0.18	0.20	0.20

Results

	R-DFS Error Density		
Subject	1 hour	2 hour	3 hour
Airline(20,2)	0.01	0.00	0.00
Reorder(9,1)	0.06	0.45	0.37
Twostage(7,1)	0.41	0.69	0.93
Twostage(8,1)	0.04	0.03	0.07
TwoStage(10,1)	0.00	0.00	0.00
Wronglock(10,1)	0.18	0.20	0.20

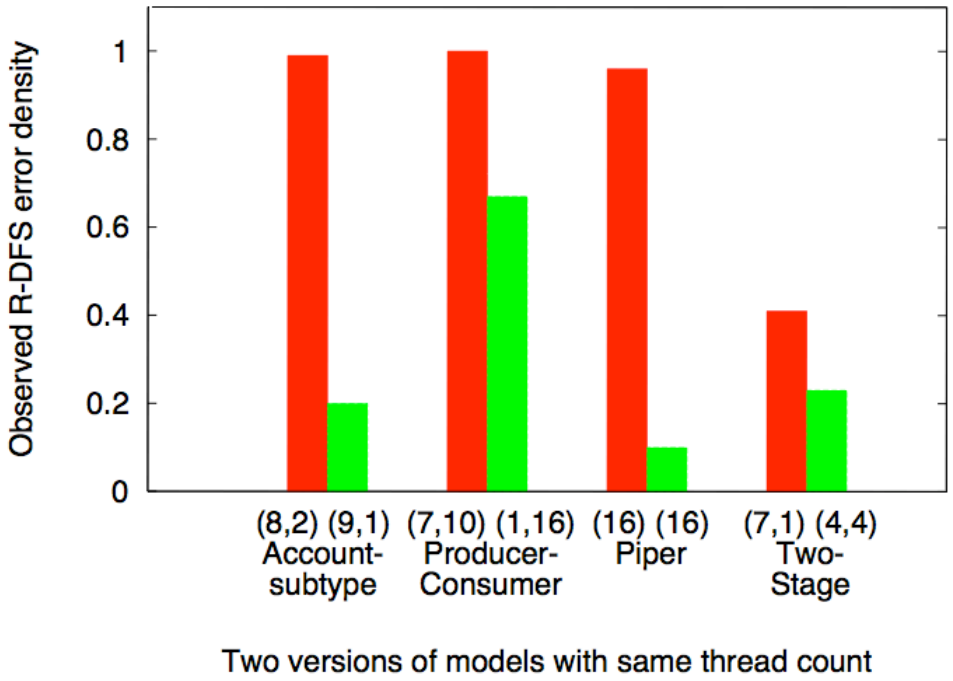
Creating hard models

- Hard models in terms of R-DFS error density
- Given a fixed time bound
- How to make a model hard?

Thread count

- Number of threads in a model
- Exponential growth in the transition graph
- Traditional semantic measure of hardness
- Does it affect observed R-DFS error density?

No it does Not !



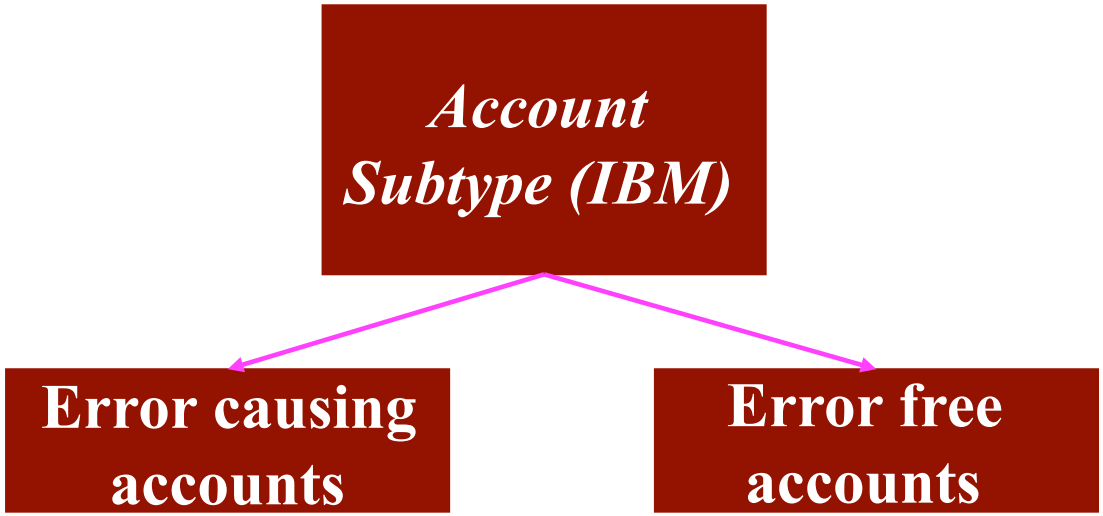
Controlling factors

- Depth of errors in transition graph
- Thread count that manifest an error
- Aid in designing hard benchmarks

Specific Thread type

- Control threads that manifest an error
- Threads that cause a race condition
- Threads that check program correctness

Threads Manifesting Errors



Thread Specific Results

Subject	Density	Min States
AccountSubtype (No-Error, Error)		
AccountSubtype(9,1)	0.20	281
AccountSubtype(10,1)	0.19	309
AccountSubtype(11,1)	0.13	332
AccountSubtype(20,1)	0.00	-
AccountSubtype(8,2)	0.99	294
AccountSubtype(8,8)	1.00	541

Thread Specific Results

Subject	Density	Min States
AccountSubtype (No-Error, Error)		
AccountSubtype(9,1)	0.20	281
AccountSubtype(10,1)	0.19	309
AccountSubtype(11,1)	0.13	332
AccountSubtype(20,1)	0.00	-
AccountSubtype(8,2)	0.99	294
AccountSubtype(8,8)	1.00	541

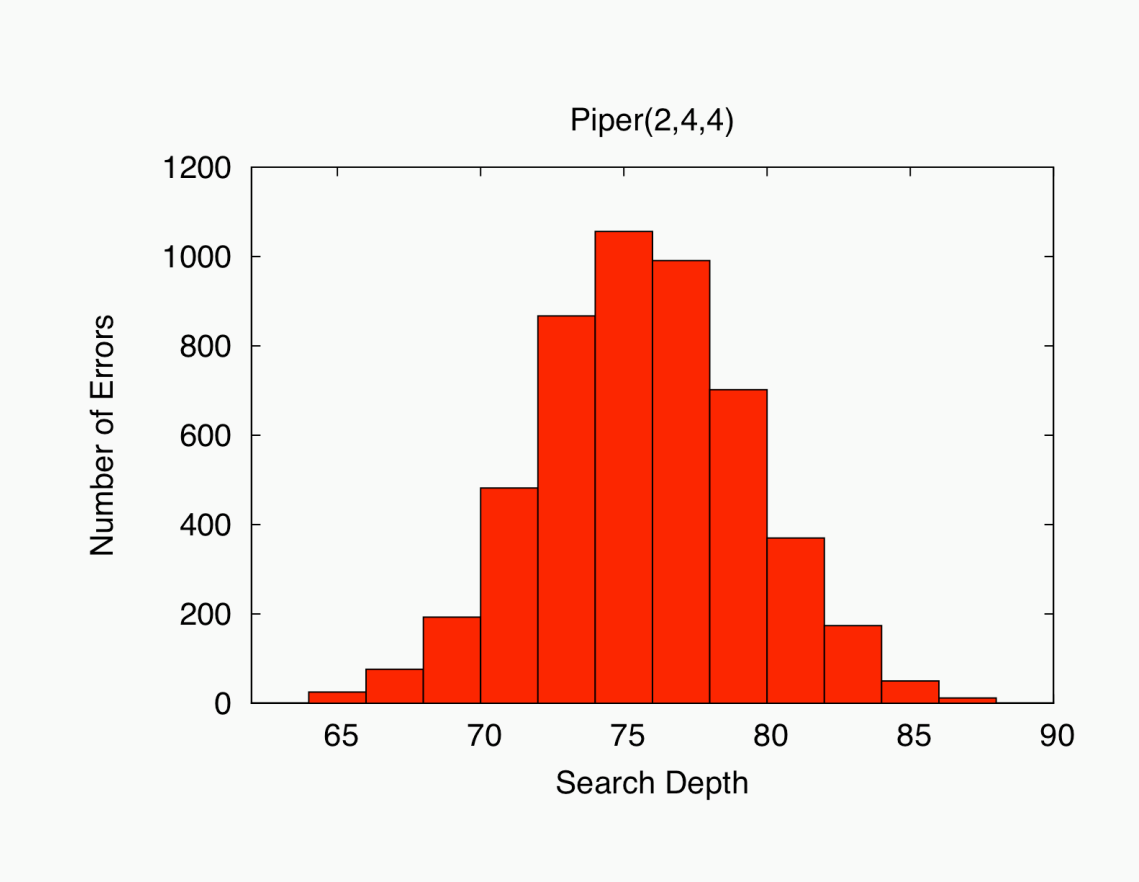
Thread Specific Results

Subject	Density	Min States
AccountSubtype (No-Error, Error)		
AccountSubtype(9,1)	0.20	281
AccountSubtype(10,1)	0.19	309
AccountSubtype(11,1)	0.13	332
AccountSubtype(20,1)	0.00	-
AccountSubtype(8,2)	0.99	294
AccountSubtype(8,8)	1.00	541

Depth of Errors

- Depth of errors in the transition graph
- Variations of model with the same thread count

Depth of Errors



Error Depth Results

Subject	Density	Min States
Piper(2,8,5)	0.59	152
Piper(2,8,6)	0.10	160
Piper(2,8,7)	0.01	7,016,824
Airline(20,7)	0.30	116
Airline(20,6)	0.19	166
Airline(20,4)	0.03	2,839,090
Airline(20,1)	0.00	-

Error Depth Results

Subject	Density	Min States
Piper(2,8,5)	0.59	152
Piper(2,8,6)	0.10	160
Piper(2,8,7)	0.01	7,016,824
Airline(20,7)	0.30	116
Airline(20,6)	0.19	166
Airline(20,4)	0.03	2,839,090
Airline(20,1)	0.00	-

Error Depth Results

Subject	Density	Min States
Piper(2,8,5)	0.59	152
Piper(2,8,6)	0.10	160
Piper(2,8,7)	0.01	7,016,824
Airline(20,7)	0.30	116
Airline(20,6)	0.19	166
Airline(20,4)	0.03	2,839,090
Airline(20,1)	0.00	-

Threats to Validity

- Path error density and R-DFS error density are semantic-based hardness measure
- Tool implementation dependent
- Values differ in two version of JPF
- Models characterized for specific tool
- R-DFS error density is a tighter bound over path error density for a specific tool

Conclusions & Future Work

- Benchmarking stateful techniques
- A stateful hardness criteria
- Randomized DFS is a good lower bound
- Possible to make hard models
- Other factors affecting hardness?
- Identifying patterns?

Questions

Software Model Checking Lab
Computer Science Department
Brigham Young University
Provo, Utah

Neha Rungta: neha@cs.byu.edu
Eric G. Mercer: egm@cs.byu.edu

http://vv.cs.byu.edu