More on Java PathFinder

Nastaran Shafiei

Department of Computer Science and Engineering
York University, Toronto, Canada

Overview

- JPF can be considered as a Java virtual machine that can execute the target application in all possible ways

Overview

- JPF can be considered as a Java virtual machine that can execute the target application in all possible ways
- Functionality of JPF can be extended through its listeners

Overview

- JPF can be considered as a Java virtual machine that can execute the target application in all possible ways
- Functionality of JPF can be extended through its listeners
    - Examples:
        - PreciseRaceDetector

Overview

- JPF can be considered as a Java virtual machine that can execute the target application in all possible ways
- Functionality of JPF can be extended through its listeners
  - Examples:
    - PreciseRaceDetector
    - ExecTracker

Overview

- JPF can be considered as a Java virtual machine that can execute the target application in all possible ways
- Functionality of JPF can be extended through its listeners
    - Examples:
        - PreciseRaceDetector
        - ExecTracker
        - StateSpaceDot

Overview

- JPF can be considered as a Java virtual machine that can execute the target application in all possible ways
- Functionality of JPF can be extended through its listeners
  - Examples:
    - PreciseRaceDetector
    - ExecTracker
    - StateSpaceDot
    - SimpleDot
    - ...

Dealing with Data Choices

- The Verify class can get control over choices
- The Verify class contains methods for creating choice generators for built-in Java types

Dealing with Data Choices

- The Verify class can get control over choices
- The Verify class contains methods for creating choice generators for built-in Java types
  - Examples:
    - Verify.getBoolean()
    - Verify.getInt(min,max)
    - Verify.getIntFromSet (1, 190, 390, 401)
    - Verify.getDoubleFromSet(-42.0, 0.0, 42.0)

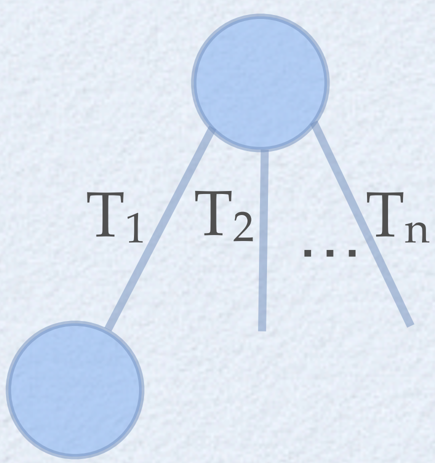
State Space Explosion

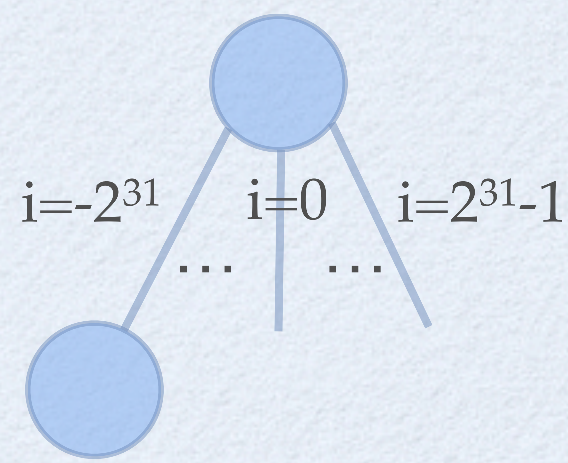
- Main issue in model checking
- Usually caused by non-determinism
  - Thread non-determinism

    i.e. the state space size can be exp. in the # of threads

  - Data non-determinism

    e.g. i = (new Random()).nextInt();





State Space Explosion

- Consider

  T_1: $\alpha_{11}\ \alpha_{12}\ \ldots\ \alpha_{1n}$

  T_2: $\alpha_{21}\ \alpha_{22}\ \ldots\ \alpha_{2m}$

State Space Explosion

- Consider

  T_1: $\alpha_{11}\ \alpha_{12}\ \ldots\ \alpha_{1n}$

  T_2: $\alpha_{21}\ \alpha_{22}\ \ldots\ \alpha_{2m}$

- How many interleavings?

State Space Explosion

- Consider

  T_1: α_{11} α_{12} ... α_{1n}

  T_2: α_{21} α_{22} ... α_{2m}

- How many interleavings?

  The same as the number of ways to choose n elements among a set of n+m

$$\binom{m+n}{n} = \frac{(m+n)!}{n!\ m!}$$

State Space Explosion

- Consider

  T_1: $\alpha_{11}\ \alpha_{12}\ \ldots\ \alpha_{1n}$

  T_2: $\alpha_{21}\ \alpha_{22}\ \ldots\ \alpha_{2n}$

  T_3: $\alpha_{21}\ \alpha_{22}\ \ldots\ \alpha_{2n}$

  $\vdots$

  T_k: $\alpha_{k1}\ \alpha_{k2}\ \ldots\ \alpha_{kn}$

State Space Explosion

- Consider

  T_1: $\alpha_{11}\ \alpha_{12}\ \ldots\ \alpha_{1n}$

  T_2: $\alpha_{21}\ \alpha_{22}\ \ldots\ \alpha_{2n}$

  T_3: $\alpha_{21}\ \alpha_{22}\ \ldots\ \alpha_{2n}$

  $\vdots$

  T_k: $\alpha_{k1}\ \alpha_{k2}\ \ldots\ \alpha_{kn}$

- How many interleavings?

State Space Explosion

- Consider

  T_1: α_{11} α_{12} ... α_{1n}

  T_2: α_{21} α_{22} ... α_{2n}

  T_3: α_{21} α_{22} ... α_{2n}

  $\vdots$

  T_k: α_{k1} α_{k2} ... α_{kn}

- How many interleavings?

$$\binom{2n}{n}\binom{3n}{n}\binom{3n}{n}\ldots\binom{kn}{n}$$

State Space Explosion

$$\binom{2n}{n}\binom{3n}{n}\binom{4n}{n}\cdots\binom{kn}{n} =$$

$$\frac{(2n)!}{n!\,n!} \times \frac{(3n)!}{n!\,(2n)!} \times \frac{(4n)!}{n!\,(3n)!} \times \ldots \times \frac{(kn)!}{n!\,((k-1)n)!}$$

State Space Explosion

$$\binom{2n}{n}\binom{3n}{n}\binom{4n}{n}\cdots\binom{kn}{n} =$$

$$\frac{(2n)!}{n!\,n!} \times \frac{(3n)!}{n!\,(2n)!} \times \frac{(4n)!}{n!\,(3n)!} \times \cdots \times \frac{(kn)!}{n!\,((k-1)n)!} = \frac{(kn)!}{(n!)^k}$$

State Space Explosion

$$\binom{2n}{n}\binom{3n}{n}\binom{4n}{n}\cdots\binom{kn}{n} =$$

$$\frac{(2n)!}{n!\,n!} \times \frac{(3n)!}{n!\,(2n)!} \times \frac{(4n)!}{n!\,(3n)!} \times \cdots \times \frac{(kn)!}{n!\,((k-1)n)!} = \frac{(kn)!}{(n!)^k}$$

$$\frac{1 \ldots n}{1 \ldots n} \times \frac{n+1 \ldots 2n}{1 \ldots n} \times \frac{2n+1 \ldots 3n}{1 \ldots n} \times \cdots \times \frac{(k-1)n+1 \ldots kn}{1 \ldots n}$$

State Space Explosion

$$\binom{2n}{n}\binom{3n}{n}\binom{4n}{n}\cdots\binom{kn}{n} =$$

$$\frac{(\cancel{2n})!}{n!\,n!} \times \frac{(\cancel{3n})!}{n!\,(\cancel{2n})!} \times \frac{(\cancel{4n})!}{n!\,(\cancel{3n})!} \times \cdots \times \frac{(kn)!}{n!\,(\cancel{(k-1)n})!} = \frac{(kn)!}{(n!)^k}$$

$$\underbrace{\frac{1 \ldots n}{1 \ldots n}}_{=1} \times \underbrace{\frac{n+1 \ldots 2n}{1 \ldots n}}_{\geq 2^n} \times \underbrace{\frac{2n+1 \ldots 3n}{1 \ldots n}}_{\geq 2^n} \times \cdots \times \underbrace{\frac{(k-1)n+1 \ldots kn}{1 \ldots n}}_{\geq 2^n} \geq (2^n)^{k-1}$$

State Space Explosion

$$\binom{2n}{n}\binom{3n}{n}\binom{4n}{n}\cdots\binom{kn}{n} =$$

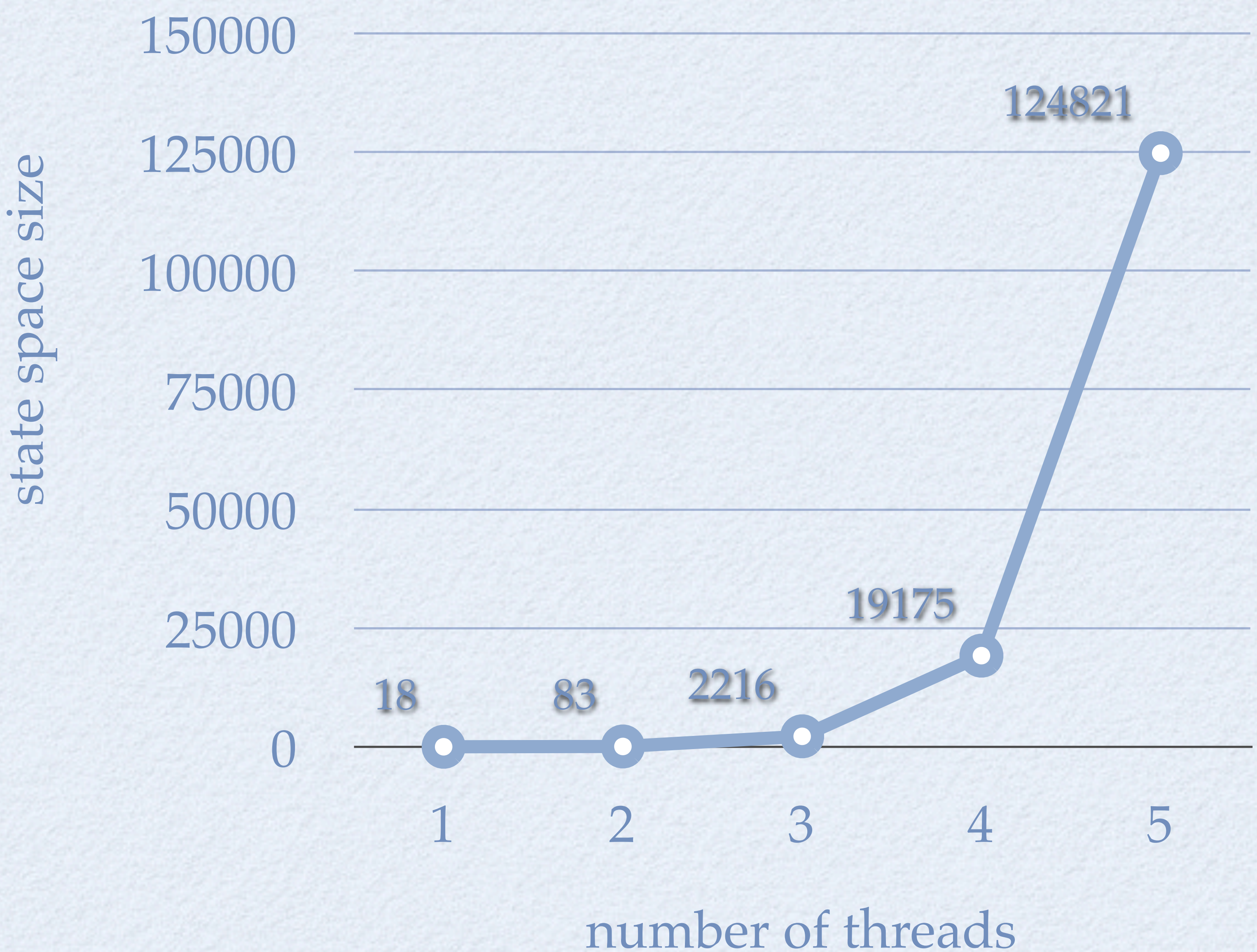
$$\frac{\cancel{(2n)}!}{n!\,n!} \times \frac{\cancel{(3n)}!}{n!\,\cancel{(2n)}!} \times \frac{\cancel{(4n)}!}{n!\,\cancel{(3n)}!} \times \cdots \times \frac{(kn)!}{n!\,(\cancel{(k-1)n})!} = \frac{(kn)!}{(n!)^k}$$

$$\underbrace{\frac{1 \ldots n}{1 \ldots n}}_{=1} \times \underbrace{\frac{n+1 \ldots 2n}{1 \ldots n}}_{\geq 2^n} \times \underbrace{\frac{2n+1 \ldots 3n}{1 \ldots n}}_{\geq 2^n} \times \cdots \times \underbrace{\frac{(k-1)n+1 \ldots kn}{1 \ldots n}}_{\geq 2^n} \geq (2^n)^{k-1}$$

$\Longrightarrow$ The number of interleavings is exponential in *n* and *k*

State Space Explosion

- Example: ReaderWriter program



Partial Order Reduction (POR)

- Reduce the number of orderings to be analyzed
- Based on identifying the independent actions
- Correctness criterion (TS: original system, TS′: reduced system)
  1. TS & TS′ are equivalent with respect to the desired property p

     $TS \models p$ iff $TS' \models p$
  2. TS′ should be smaller than TS

Partial Order Reduction (POR)

- Example

  T_1: α (x=1)

  T_2: β (y=1)

  T_3: γ (z=1)

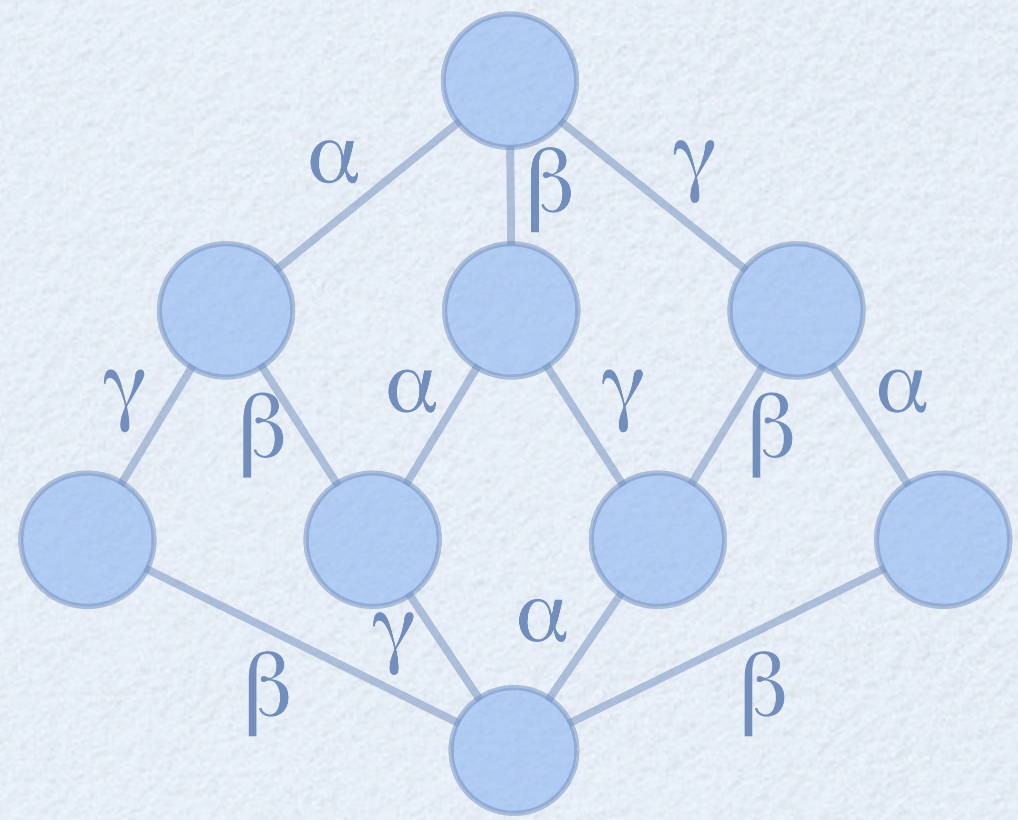
Partial Order Reduction (POR)

- Example

  T_1: α (x=1)

  T_2: β (y=1)

  T_3: γ (z=1)



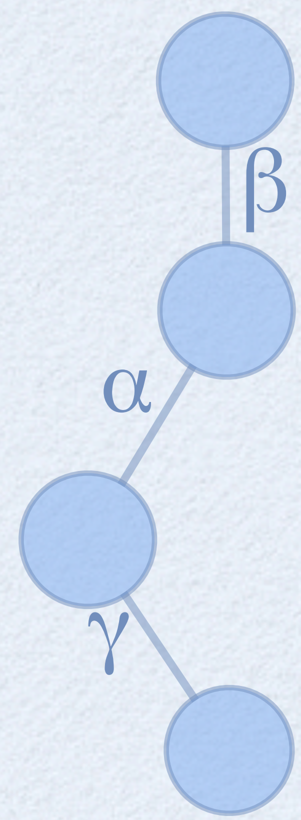
Partial Order Reduction (POR)

- Example

  T_1: α (x=1)

  T_2: β (y=1)

  T_3: γ (z=1)



Analyzing 1 ordering instead of 3!

Partial Order Reduction (POR)

- Generalization: Analyzing 1 ordering, instead of n!
  - Reduced system: grows linearly in n
  - Original system: grows exp. in number of components
- Assumption
  - No synchronizations are involved, e.g. shared variables
  - The property of interest is independent of intermediate states

JPF Reduction Techniques

- Partial order reduction (POR)
- Garbage collection
- State compression

POR of JPF

- Based on combining a sequence of bytecode instructions in a thread that do not have any effects outside it
- On accessing fields, JPF performs some tests to decide to break the transition, e.g.
  - Does not break the transition,
    - if the field is protected by lock
    - if the field is defined as final
    - if the field belongs to an immutable object
    - ...

POR of JPF

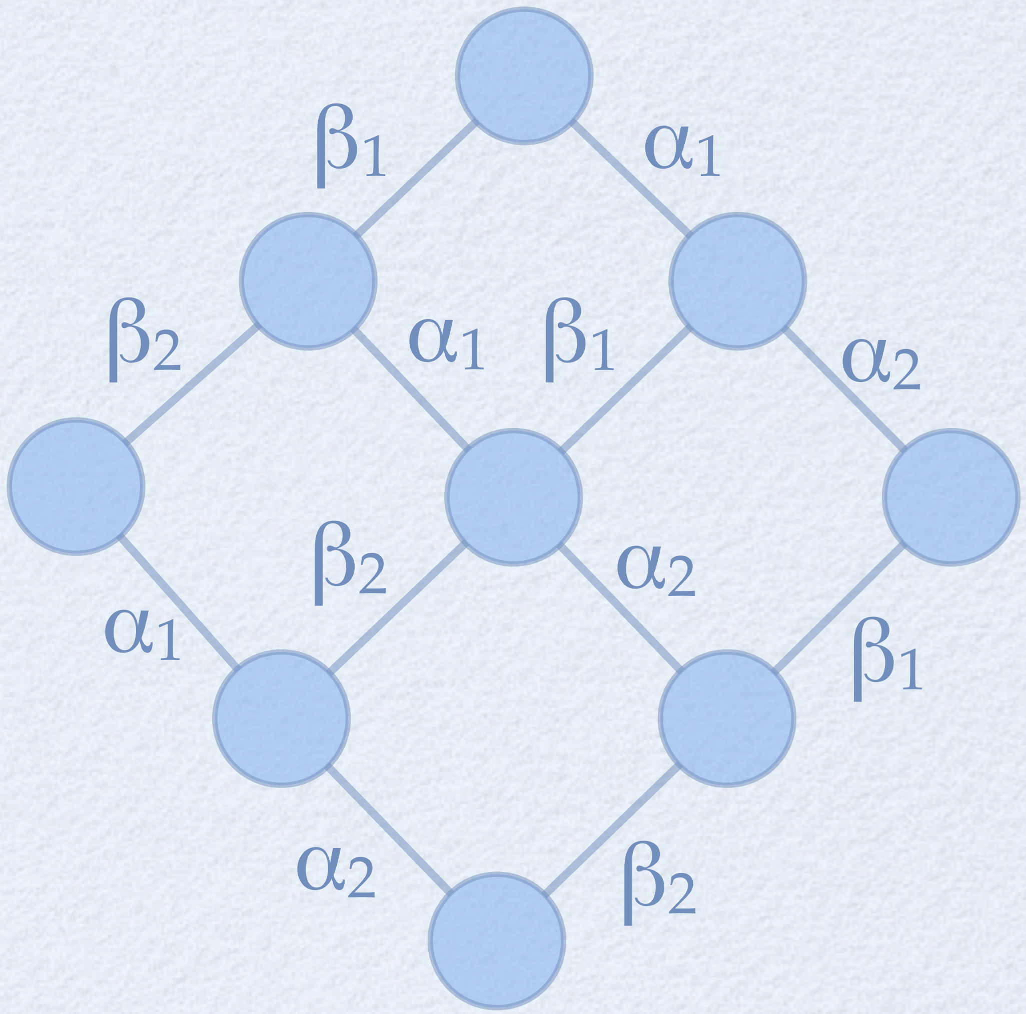
- Example,

  T1: α_1 α_2

  T2: β_1 β_2

POR of JPF

- Example,
  - T1: α_1 α_2
  - T2: β_1 β_2

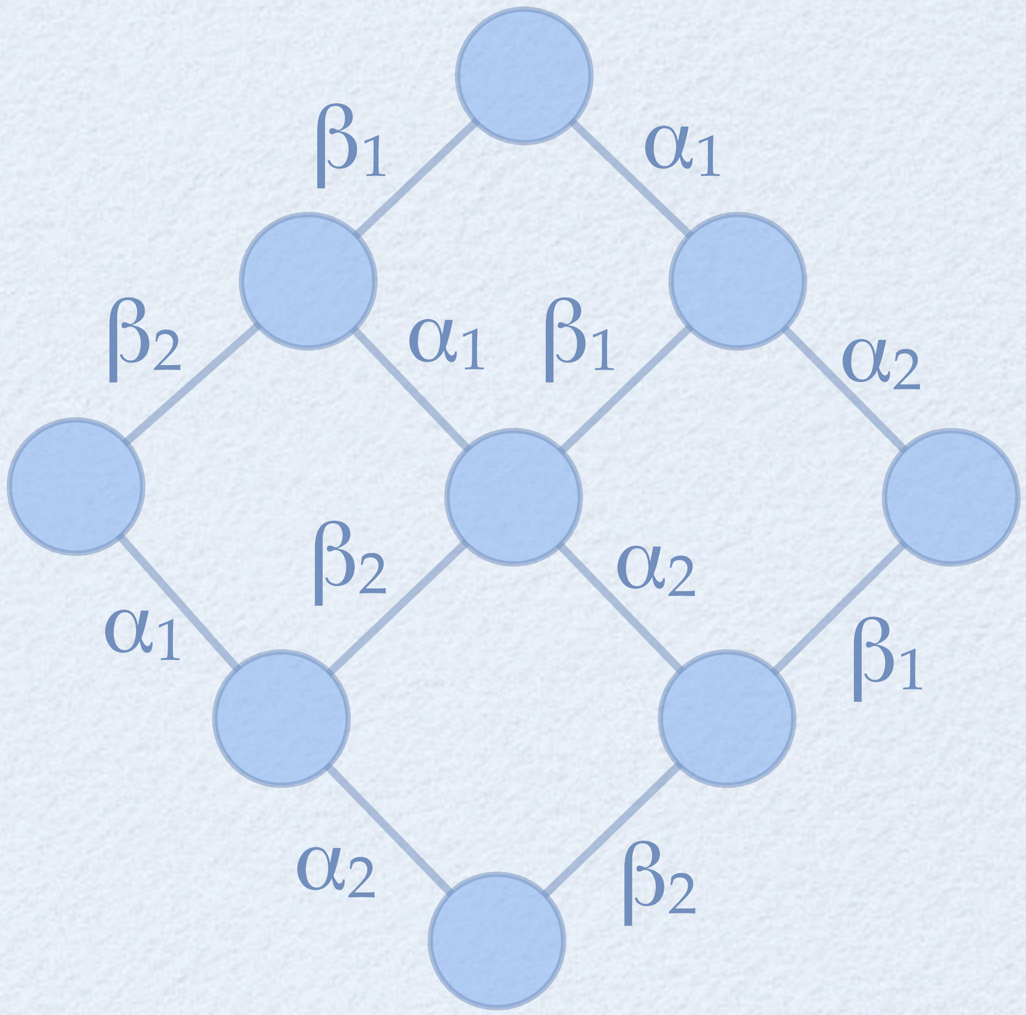


POR of JPF

- Example,

  T1: α_1 α_2

  T2: β_1 β_2



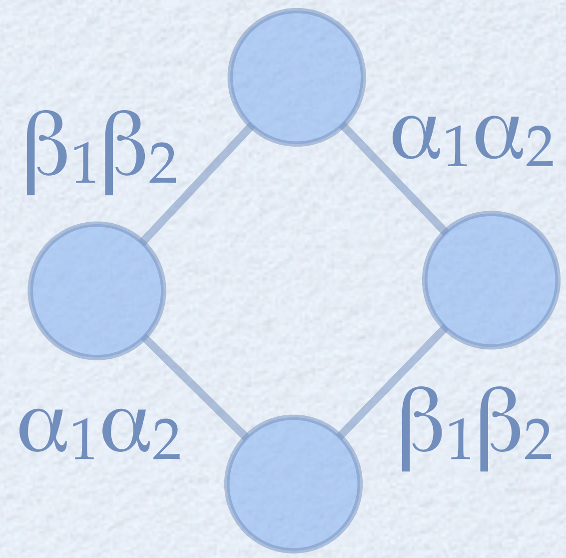
- α_1: read a final field
- α_2 : write to a non-shared field
- β_1: read a final field
- β_2: write to a non-shared field

POR of JPF

- Example,

  T1: α_1 α_2

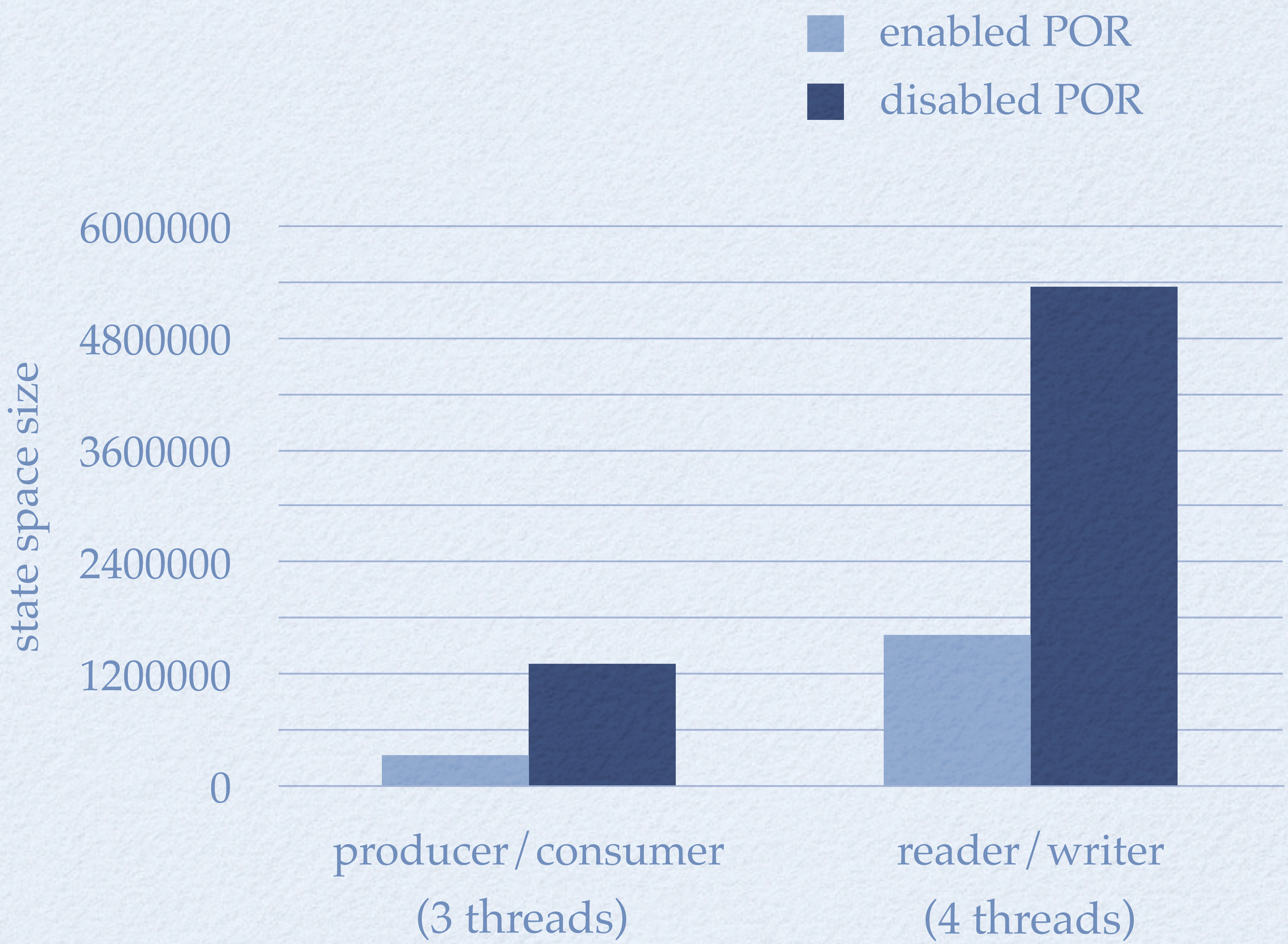
  T2: β_1 β_2



- α_1: read a final field
- α_2 : write to a non-shared field

- β_1: read a final field
- $\beta_{2:}$ write to a non-shared field

The Effect of JPF POR



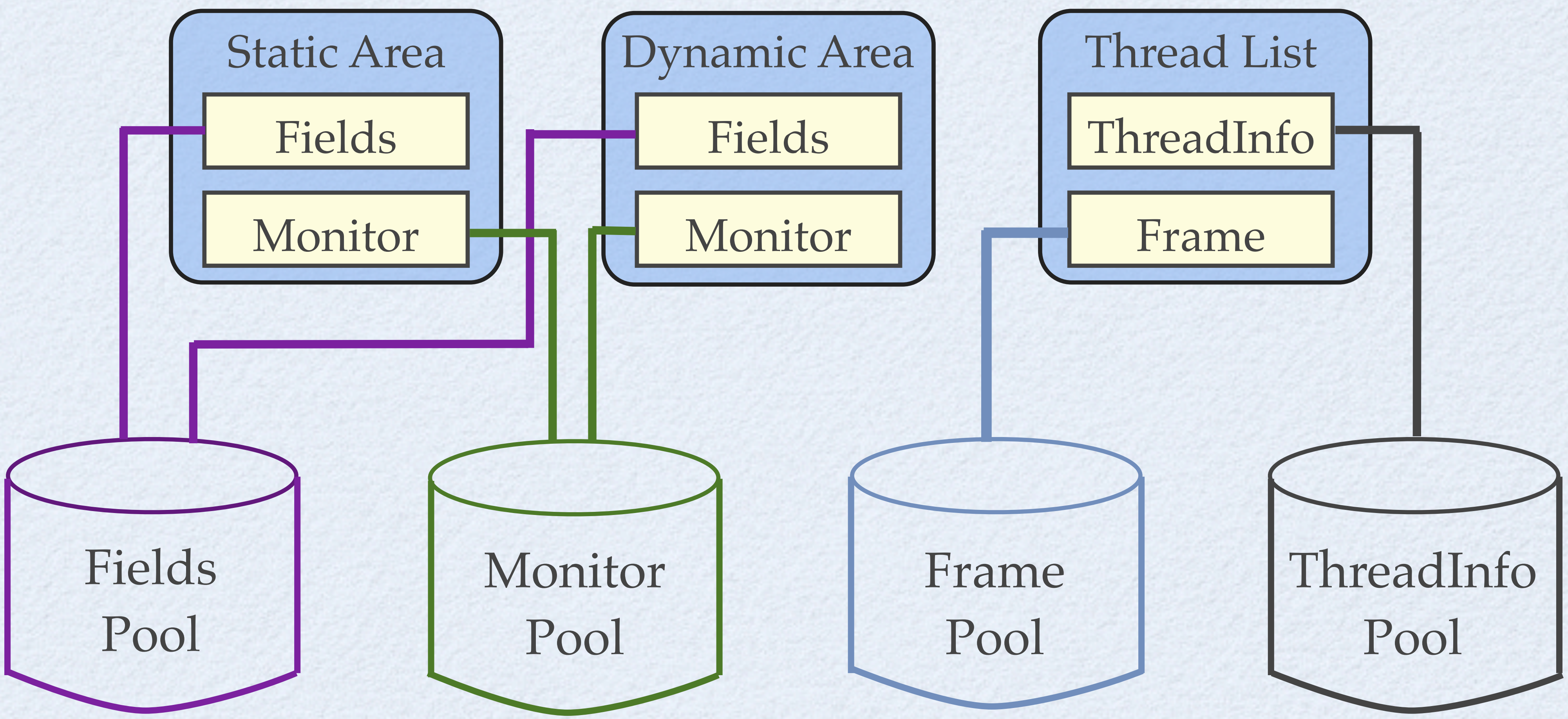
Garbage Collection

- Based on mark and sweep algorithm
  - Marking phase: marks all the objects referenced by the current state
  - Sweeping phase: removes all the objects that have not been marked
- Example: Without garbage collection, model checking the following code leads to infinite number of states

```
while(true)
{
    new Object();
}
```

State Compression

- JPF uses the collapse method originally used in SPIN
- Basic idea: a transition may change only a small part of the state



Control the # Interleavings

- Using the Verify class to control the number of threads interleavings that JPF has to explore
  - Make a part of the code atomic

```
Verify.beginAtomic();
    ....
Verify.endAtomic();
```

} executed by JPF in one transition

Control the # Interleavings

- Restrict the search
  - If the provided expression evaluates to true, JPF does not continue to execute the current path, and backtracks to the previous non-deterministic choice point.

```
Verify.ignoreIf(boolean_expression);
```

Questions?