A semi-automatic dynamic memory management mechanism for embedded real-time Java applications

Nicolas Berthier

December 18, 2007
1 **Context**
- General context
- New memory management mechanism
- The problem

2 **A new solution**
- Idea
- Solution

3 **Experimentations**
- More programs are correctly managed
- But some are still not managed
Outline

1 Context
   - General context
   - New memory management mechanism
   - The problem

2 A new solution
   - Idea
   - Solution

3 Experimentations
   - More programs are correctly managed
   - But some are still not managed
Context

Embedded systems

Real-time systems
Context

- Embedded systems
- Real-time systems
- Small amount of memory

Semi-automatic memory management for RTJ applications
Context

**Embedded systems**

**Small amount of memory**

**Real-time systems**

**Java applications**

Semi-automatic memory management for RTJ applications
Context

- Embedded systems
- Real-time systems
- Small amount of memory
- Very small executable
- Java applications

Semi-automatic memory management for RTJ applications
Context

- Embedded systems
- Real-time systems
- Small amount of memory
- Very small executable
- Java applications
- Automatic dynamic memory management

Semi-automatic memory management for RTJ applications
Context

Embedded systems

Small amount of memory

Very small executable

Real-time systems

Java applications

Automatic dynamic memory management

Popular

Semi-automatic memory management for RTJ applications
Context

- Embedded systems
- Real-time systems
- Garbage collector
- Java applications

- Small amount of memory
- Very small executable
- Automatic dynamic memory management
- Popular
Context

- Embedded systems
  - Small amount of memory
- Real-time systems
  - Predictable execution time
- Java applications
  - Very small executable
  - Automatic dynamic memory management
  - Popular
- Garbage collector
Context

- Embedded systems
  - Small amount of memory
- Java applications
  - Very small executable
  - Automatic dynamic memory management
- Real-time systems
  - Predictable execution time

Garbage collector

Popular
New memory management mechanism

Static analysis of the program

- grouping objects in regions
  - a classic heap...
  - ...managed with regions

- single time deallocation of a complete region
New memory management mechanism

Static analysis of the program

- grouping objects in regions
- single time deallocation of a complete region

⇒ predictable time
It works...

⇒ correct execution

Semi-automatic memory management for RTJ applications
A new solution

Experimentations

Conclusion

... but not always

⇒ increasing memory usage

Semi-automatic memory management for RTJ applications
Outline

1 Context
   • General context
   • New memory management mechanism
   • The problem

2 A new solution
   • Idea
   • Solution

3 Experimentations
   • More programs are correctly managed
   • But some are still not managed
Idea

Using a reference counting garbage collector
Idea

Using a reference counting garbage collector
Idea

Using a reference counting garbage collector
Idea

Using a reference counting garbage collector
Idea

Using a reference counting garbage collector
Idea

Using a reference counting garbage collector
Idea

Using a reference counting garbage collector
Idea

Using a reference counting garbage collector
Idea

Using a reference counting garbage collector
Idea

Using a reference counting garbage collector
Idea

Using a reference counting garbage collector

- **Advantage:**
  - respects the real-time constraints

- **Drawback:**
  - cannot manage cyclic structures
Idea

Using a reference counting garbage collector

- Advantage:
  - respects the real-time constraints

- Drawback:
  - cannot manage cyclic structures
Idea

Using a reference counting garbage collector

- Advantage:
  - respects the real-time constraints

- Drawback:
  - cannot manage cyclic structures
Solution

**Hybrid mechanism**

- region based memory management
- reference counting garbage collector

⇒ respects real-time constraints
1 Context
- General context
- New memory management mechanism
- The problem

2 A new solution
- Idea
- Solution

3 Experimentations
- More programs are correctly managed
- But some are still not managed
More programs are correctly managed

⇒ constant heap occupation
But some are still not managed

⇒ the heap occupation is still increasing!
But some are still not managed

⇒ complex data structures (e.g. irregular graphs)
Conclusion

Results

- more programs are correctly managed
- identification of pathological cases
  - manipulation of complex structures
  - irregular memory usage
Conclusion

Results
- more programs are correctly managed
- identification of pathological cases
  - manipulation of complex structures
  - irregular memory usage

Perspectives
- creating a new analysis to detect potential cycles
Thank you for your attention

Do you have any questions?