トップエスイー修了制作

Java VM上におけるJava PathFinderで検出した不具合再現の自動化

株式会社富士通研究所　大木 憲二　oki.kenji@jp.fujitsu.com

開発における問題点

マルチスレッドプログラムのデバッグは再現性がタイミングに依存するため困難である.
モデル検査ツールであるJava PathFinder(JPF)はスレッドの切り替えを加味して障害へ至る反例を検出してくれるが，独自のVM上でプログラムを動かすため，既存のデバッガを活用することができずその後のデバッグ作業が効率化されない.

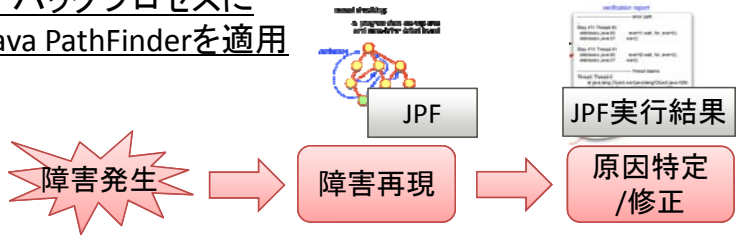
手法・ツールの適用による解決

Java PathFinderが検出した反例と同等の振る舞いをJavaデバッガ上で再現する方式を考案し，プロトタイプツールを制作した. ツールが生成する下記の情報を用いる事でJavaデバッガ上での再現の手間を減らしつつ，デバッグ作業を行えるようになる.

- スレッドの中断条件（ブレークポイント）
- スレッドの切替え順序

問題概要

デバッグプロセスに
Java PathFinderを適用



✓ Java VMと振る舞いが異なるかもしれない

✓ ログ出力のみのデバッグは大変

既存Javaデバッガとの連携が不可欠

デバッガ再現情報の生成

Java PathFinderの出力例（抜粋）

反例

```
---------------------------------- transition #13 thread: 1
ThreadChoiceFromSet {id:"LOCK"}
Human.java:19 : eat();
Human.java:29 : synchronized (this.stick1) {
---------------------------------- transition #14 thread: 2
ThreadChoiceFromSet {id:"LOCK"}
...
```

遷移直前の行

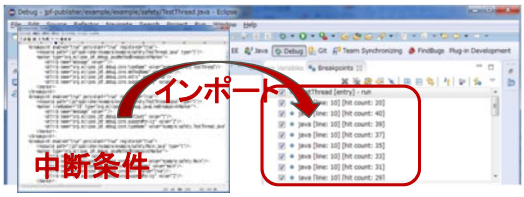
状態遷移

スレッドの中断条件：

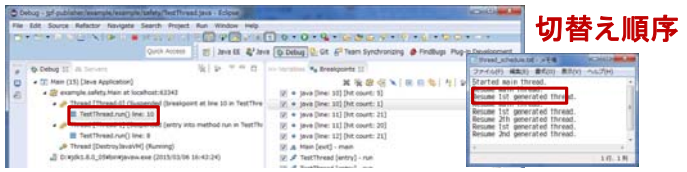
1. 実行スレッドが状態遷移直前のコード行に到達
2. 当該行が実行開始から表れた回数だけ到達
3. 状態遷移時に実行スレッドが変更

生成情報を用いた再現作業

1. ブレークポイントファイルをインポートし，プログラムを実行



2. プログラム中断後にスレッド切替え順序を見て次に動かすスレッドを確認/再開



切替え順序に記載されている分だけ2.を繰り返すことで，障害発生地点に到達

効果

Eclipseデバッガ上再現時の手番数比較

	ブレークポイントの追加・削除操作	スレッドの再開操作
ツール無	115回	57回
ツール有	1回(インポート)	11回

(※)Java PathFinder上で58回の状態遷移発生後にエラーに到達するプログラムを用いて計測

- 複数のブレークポイントが一括でインポートされるため，途中で追加や削除を行うことなく再現が可能
- スレッドが切り替わった場合のみ実行が中断されるため再開の手番数を削減

再現の手間が削減されデバッグ作業に注力