パターンを用いたフォールト検出ツールに関する研究

2010SE214 鈴木淑博
指導教員：張漢明

1　はじめに

マルチコアプロセッサの出現や分散システムの普及に伴い，並行プログラミング技術の重要性が注目されている．一般的に，並行システムの記述をテストやレビューで検証することは困難であり，システムの状態を網羅的に検査するモデル検査の有用性が報告されている [1]．並行システムの検証では，検証が失敗した場合，その原因を究明する作業が困難であり，検証にかかるコストが高くなる要因の一つである．

本研究の目的は，並行システムの検証を行う前に，典型的なフォールトの可能性がある記述を指摘することにより，検証コストを削減することである．並行システムのある仕様に対して，その誤りを提示することにより，フォールトの検出を実現する．誤りを分類して定義することにより，誤りの詳細な情報提示が可能になる．

2　フォールトの検出方法

フォールト検出の枠組みを図 1 に示す．

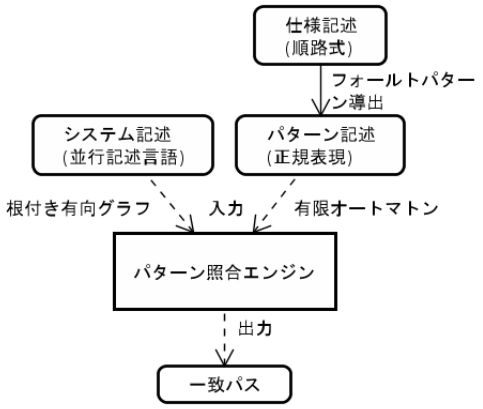

図 1　フォールト検出の枠組み

あるシステムから, ある仕様に対する違反, 即ちフォールトを検出する．まず, 仕様記述から, その仕様記述に対するフォールトのパターン記述を導出する．次にパターン照合エンジンによって, システムとフォールトのパターンを照合する．全体としてシステムのフォールト検出が実現される．

2.1　システム記述

並行システムの記述は根付き有向グラフに変換されることを前提とする．グラフの辺のラベルはイベントを表し，システムの振舞いをイベント列として表現する．

2.2　仕様記述

仕様は順路式 (Path Expression) [2] を用いて定義する．順路式は型定義された対象への操作のパターンを記述するものである．順路式は正規表現を拡張したもので，操作間の実行回数に関する制約や，実行するための条件等も記述することができる．

2.3　フォールトパターンの導出

仕様を示す順路式に，変換関数を適用することにより，フォールトのパターンを導出する [4]．変換関数は順路式の各式に対して定義される．

2.4　パターン記述

パターンは正規表現で記述される．正規表現に用いる演算子を表 1 に示す．

表 1

演算子	意味
;	逐次実行
+	排他選択
*	0 回以上の繰り返し

2.5　パターン照合エンジン

パターン照合エンジンは，システムからパターンに一致するパスを検出する．正規表現で記述されたパターンを有限オートマトンに変換して，パターンの照合を，オートマトンを受理する有向グラフのパスを検出する問題に帰着させている [3].

2.6　一致パスの例

パターン照合エンジンは, 照合の結果, パターンに一致するパスを出力する．しかし, イベント列をそのまま表示しても, パスが正規表現とどのように一致しているのかが理解し難い場合がある．例えば, 以下のようなパターン照合である．

【入力】
システム：略
パターン：(cmd;(a+b);rsp)*;cmd;(a+b);(cmd+a+b)
【出力】
パス：cmd a rsp cmd a rsp cmd a rsp cmd b rsp cmd b rsp cmd a rsp cmd b rsp cmd a b

そこで, パスを分析するための, より詳細な情報を付加する．

3　検出パスの詳細情報

検出したパスが正規表現とどのように一致しているかの情報を考える.

3.1　構造

正規表現の木構造と対応させて,「繰り返し回数」と「選択された子の番号」の情報を持った木構造を作る. 繰り返しが複数の場合は同じ数の子を追加する. 例として, 図 2 に 正規表現 (a;(a+b)*;c) と詳細情報 (aaacbabc) の木構造を示す.

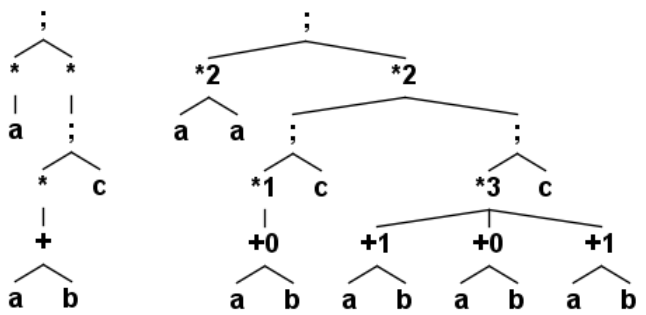

図 2　正規表現と詳細情報の木構造

3.2　詳細情報の作成方法

3.2.1　正規表現上のパス

パスがどのような順で正規表現上のイベントノードを辿るのかを調べる必要がある. 正規表現からパターン照合に用いる非決定性有限オートマトンを作成する際, あらかじめ, 正規表現上のイベントとオートマトン上のイベントの対応関係を記録しておく. 次に非決定性有限オートマトンを決定性有限オートマトンに変換することで状態遷移先を一意にし探索を高速化する. しかし, 非決定的な遷移を含む正規表現 (a+a;b, (a;b)*;a 等) を, 決定性有限オートマトンに変換すると, 正規表現上のイベントとオートマトン上のイベントの対応関係を一部失ってしまう. したがって, パターン照合では決定性有限オートマトンを用いるが, 照合で得られたパスから正規表現上のパスを探索する際は, 非決定性有限オートマトンを用いる.

3.2.2　木構造の作成

次に詳細情報の木構造を作成する. 正規表現上のパスを順に訪れ, 正規表現の繰り返し末尾に相当するノードを訪れた場合は, 対応する繰り返しノードの子となっている部分木を複製し, 新しい子として追加する. 選択に相当するノードを訪れた場合は, 対応する選択ノードに, 選択された子の番号を記録する.

3.2.3　文字列の作成

木構造を根から訪問していき, 表 2 のようにイベントと逐次についてはそのまま表示し. 選択は, どちらの部分木が選択されたかを下線で表示し, 繰り返しは, 繰り返し回数を添えて表示する. 繰り返しの中身については指定された深さまで展開して表示するようにする.

表 2

正規表現	表示
e	e
a;b	a;b
a+b	a+b
a*	a#num

3.3　検出例

詳細情報を生成することで, 2.6 節で示したパターン照合において以下のように結果表示できる.

【入力】
システム：略
パターン：(cmd;(a+b);rsp)*;cmd;(a+b);(cmd+a+b)
【出力】
詳細情報：(cmd;(a+b);rsp)#7;cmd;(a+b);(cmd+a+b)

4　表示方法の改良

詳細情報の表示方法を切り替えられるようにすることで, 人間にとってさらに理解しやすいものになると考える. 例えば, 知りたい部分の繰り返しだけを展開して表示したり, 選択されなかった木を排除して表示することである. そのためには GUI で対話的に表示方法を切り替えられるようにすれば良い.

5　おわりに

本研究では, フォールト検出ツールを用いて, 並行システムから, 仕様に対するフォールトを検出する方法、および, その出力方法の改良について議論した. 順路式特有の表記とフォールトパターンの導出方法についてはまだ明らかでないので今後, 議論する必要がある.

参考文献

[1] 中島震：モデル検査法のソフトウェアデザイン検証への応用, コンピュータソフトウェア, Vol.23, No.2 (2006), pp. 72–86.

[2] Habermann, A. N.: *Path Expression*, Department of Computer Science, Carnegie-Mellon University (1975).

[3] 山内宏也, ”並行システム記述に対するパスの照合の研究, ” 南山大学院 2012 年度修士論文, 2013.

[4] 川瀬信吾, ”フォールトのパターン化によるモデル検査支援に関する研究, ” 南山大学院 2013 年度修士論文, 2014.