トップエスイー修了制作

Alloy: 時間経過に伴う状態遷移モデルのフレームワーク

NECソリューションイノベータ　　谷岡　祐志

背景と課題

【背景】形式手法の一つであるAlloyは，軽量でありインクリメンタルな開発が可能であるため，ソフトウェア開発の各フェーズでの品質向上に寄与する可能性が高く実務利用に期待される．
【課題】実務では，モデル記述自体やモデルの検討に多くの時間を割くことはできないため，ゼロベースでモデル検討を行うことは困難．

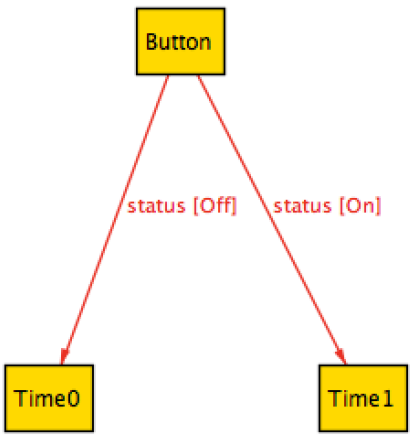
手法・ツールの適用による解決

【目的】モデル記述の効率化
・モデル記述の一部自動化（テンプレート化）．
・記述漏れ防止．
【解決手法】フレームワークを開発して利用する．
・時間経過に伴う状態遷移モデルについて記述の構造を分析，可変部分を明確にすることでフレームワークとして利用．

状態遷移モデル

時間経過に伴う状態遷移モデルのAlloy記述方法を提案する．
・時間を全順序集合「Time」とする．
・状態を持つ対象を集合とする．
・状態を集合とし，「状態」から「時間」の関係を定義する．
・状態遷移を，時間t，
t’∈Timeに対して，tからt’に時間経過する状態変化として定義する．

（例）左図は，上記に従い状態遷移「Buttonを押す」を記述し，Alloy Analyzerにてシミュレートしたもの．時間経過「Time0」から「Time1」に対して，状態が「Off」から「On」に変化することを示している．



フレームワークの概要

フレームワークは，左記に提案した状態遷移モデルの記述を補助するテンプレートを出力する．テンプレートの決められた部分に情報を追記することでAlloyコードのモデルを構築する．

左記の例に対しては，「Buttonを押す」という状態遷移名，関連する集合名として「Button」を入力することでテンプレートを出力する．

評価

- 時間経過に伴う状態遷移モデル記述の自動化

  2つのサンプルモデルに対して，フレームワークを適用し，モデル全体のAlloyコード行数と自動で構築される行数について調査．
  （フレームワークの適用結果）

	全体（行）	自動記述（行）	比率
サンプルA	76	33	43%
サンプルB	113	46	40%

  ⇒40%程度の自動化を達成．

- 状態遷移の記述に於けるフレーム条件の記述漏れを防止

  ⇒決められた記述箇所に追記を行うことで防止可能．

結果の考察と今後の課題

- 全コードの自動生成
  今回のフレームワークでは，全体のうち40%程度の自動化であり，100%ではないが，軽量でインクリメンタルなモデル構築というAlloyの特性を活かす上で十分であり，全コードの自動生成は不要であると考える．

- 残りのコードの記述には，完成するモデル記述がどのようなものであるかを事前に知っておく必要があり，ゼロベースでのフレームワークの利用は何らかの補助が必要である．

- 時間経過に伴う状態遷移モデル以外のモデルの解析例としては，ソフトウェア開発を行う上で利用する様々なデータ構造についてモデル検討を行い，記述構造や可変部分を明確にする．