区間振る舞いモデルを用いた振る舞い検証支援に関する研究

M2014SE003 近藤 翔太
指導教員：沢田 篤史

1 はじめに

近年, 並行システムの普及により, 並行システムに求められる信頼性が高まっている. 並行システムの信頼性を向上させる技術としてモデル検査がある. モデル検査は, 並行システムの有限の状態空間を網羅的に探索して, 仕様を満たしているかどうかを自動で検証する技術である. モデル検査では実用規模の対象を検査する場合, 状態爆発を起こす可能性がある. モデル検査を実用的な時間で行うためには適切な抽象化を行い, システムの状態空間の広さを抑えることが重要となる.

並行システムでは複数の事象が際どいタイミングで発生することは避けられず, 際どいタイミングにおける事象の取りこぼしが障害の原因となる. しかし, 際どいタイミングにおける障害は再現性が低く, テストによる検証は困難である. 際どいタイミングにおける性質を検証するためには設計段階での振る舞い検証が重要である.

本研究の目的は並行システムにおける際どいタイミングの事象生起を考慮した振る舞い検証の支援を行うことである. 際どいタイミングにおける振る舞いを検証するためには際どいタイミングの定義が必要である.

本研究の基本的なアイデアは, 区間振る舞いモデルを用いて際どいタイミングにおける振る舞いを厳密に定義することである. また, 区間振る舞いモデルをもとに仕様を記述するためのライブラリを提供する. 区間振る舞いモデルではあるイベント生起から別のあるイベント生起までの間を区間と捉え, 区間で起こるイベントを関心のあるイベントのみに限定する. 着目するイベントを限定することで, 取りうるイベントの組み合わせを減らすことができる. また, 際どいタイミングにおける事象発生は区間で複数のイベントが起こるということに帰着する.

本研究では自動販売機システムを事例として区間振る舞いモデルを用いた振る舞い検証の有効性を確認した. 検証では, 区間振る舞いモデルを用いた検証を行うことで, 関心のあるイベントに関しては際どいタイミングにおける障害が起こらないことを保証した. また, 際どいタイミングにおける振る舞いを定式化することで, 仕様としてどれだけの性質を記述すればよいかが明確になった.

2 背景技術

2.1 CSP

2.1.1 CSP の概要

CSP[1] とはプロセス代数の一つで, 並行システムの振る舞いを記述, 分析するための形式言語である.CSP では並行システムを並行に動作する複数のプロセスとして捉える. プロセスはイベントの順序を規定したもので, イベントはメッセージ通信や動作を抽象化したものである.

2.1.2 詳細化関係

CSP には詳細化関係があり仕様が実現を満たしていることを表すことができる. CSP における仕様と実現の関係を以下に示す.

仕様 `P [= 実現 Q`

仕様 P と実現 Q はそれぞれプロセスを表す.P と Q をつなぐ演算子が詳細化関係を表しており,P は Q に詳細化されることを表している.

2.2 モデル検査

2.2.1 モデル検査の概要

モデル検査は並行システムの有限の状態空間を網羅的に探索して与えられた性質が成り立つかを確認する技術である [4].

2.2.2 状態空間の抽象化

モデル検査では実世界の対象を扱う際には状態爆発によって自動検証できない可能性がある. これらを解決するためには検査対象に対して適切な抽象化を行い, 検証したい性質と無関係な情報を捨て去ることで, 状態数を減らすことが重要である.

2.2.3 FDR

CSP を用いた代表的なモデル検査ツールの 1 つとして FDR(Failures Divergence Refinement checker)[2] がある. FDR は CSP における詳細化関係を自動的に検証することが出来る.

3 際どいタイミングの事象に関する問題点

並行システムでは複数のプロセスが非同期通信をしながら動作しているとき, あるイベントの送信から受信までの間に別のイベントが生起する可能性があり, 複数のイベントを同時に受信する可能性がある. このような際どいタイミングでイベントを受信した場合でもシステムが正しく動くことを検証する必要がある. 際どいタイミングにおける事象発生の問題点について,ATM システムの例を用いて説明する.

3.1 際どいタイミングで発生する欠陥の例

図 1 は ATM システムで現金の引き出しを行う際の事象取りこぼしの例である. ATM システムは 利用者側の ATM 端末と端末からの情報を集約して制御するホストに分かれている. 端末側は引き出しの要求をホストへ送ったあと, 一定時間応答が帰ってこなければ取引を中止し, リセットする.

しかし, 通信が非同期の場合, 端末が時間切れを検知してから, ホストへ取引中止のメッセージが届くまでの間に, センター側から応答が返ってきてしまう可能性がある. 端

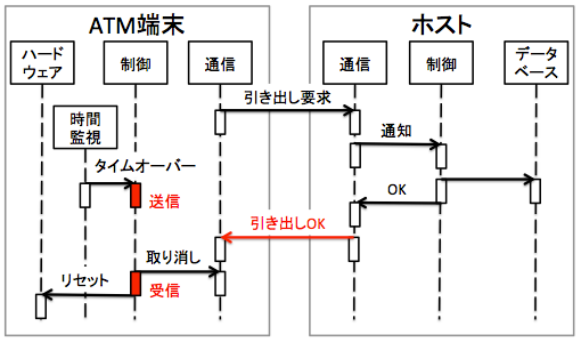

図 1　事象取りこぼしの例

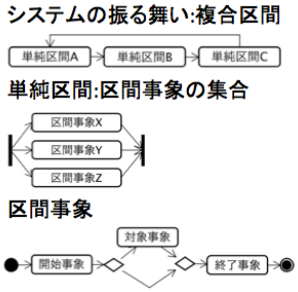

図 2　区間振る舞いモデル

末はタイムオーバーを検知した時点でリセット処理を行うので, センターからの応答を取りこぼしてしまう.

3.2　解決すべき課題

上記のような事象の取りこぼしを検証するには際どいタイミングにおける厳密な仕様記述を行う必要がある. 仕様記述を行う際の解決すべき課題点について述べる.

3.2.1　際どいタイミングの表現方法

際どいタイミングにおける振る舞いは検証の際にはイベントの系列に置き換えなければならない. 際どいタイミングをイベントの順序で記述するためには

- イベントの発生するコンポーネント
- 際どいタイミングで発生するイベント
- イベントの相互作用

を厳密に記述する必要がある. これらのイベントの取りうる系列を全て記述することは困難であり, 発生するイベントをどのように限定するかが技術的な課題となる.

3.2.2　仕様の定義

際どいタイミングにおける仕様を全て書ききるには, 仕様としてどのような性質を記述すれば良いかを明らかにする必要がある. そのためには, 際どいタイミングにおける振る舞いを定義する必要がある.

4　区間振る舞いモデル

4.1　区間振る舞いモデルの概要

区間振る舞いモデルの基本的なアイデアを以下に示す.

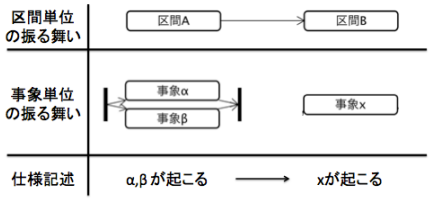

図 3　区間振る舞いモデルに基づく仕様

- システム全体の振る舞いを区間の合成として表現する
- 際どいタイミングにおける振る舞いを区間に局所化する

図 2 は区間振る舞いモデルの例を表したものである. 区間振る舞いモデルはシステム全体の振る舞いを区間の合成で表現する. 区間の最小単位を単純区間と呼び, 単純区間は区間内で起こりうる事象の集合として表現する. それぞれの事象には区間内で起こるか起こらないかを表すために開始事象と終了事象を導入する.

4.2　区間振る舞いモデルに基づく仕様

仕様では区間振る舞いモデルに基づいて, 事象単位の振る舞いと, 区間単位での振る舞いを記述する. 事象単位の振る舞いでは区間の中でどの事象が起こるかということを記述する. 区間単位の振る舞いでは区間で起こる事象の関連を記述する. 図 3 の例では区間単位の振る舞いとして区間 A のあと区間 B が起こり, 事象単位の振る舞いとして区間 A では事象 α, β が起こり, 区間 B では事象 x が起こるということを表している. 区間振る舞いモデルを用いることで, 際どいタイミングにおける振る舞いは区間の中で複数の事象が起こるという振る舞いで表現可能となる.

4.3　CSP による定義

4.3.1　区間振る舞いモデル

区間

区間は単純区間または複合区間で構成される.

区間:= 単純区間 | 複合区間

複合区間

複合区間は区間の系列として定義する.

複合区間 = Seq(区間, 区間)

CSP による定義を以下に示す.

SEQ(S1,S2) = S1; S2

複合区間は逐次の演算子を用いて再帰的な定義を可能とする. 区間の逐次合成は CSP の逐次合成演算子を用いて定義される.

単純区間

単純区間は区間事象の集合として定義する.

単純区間:= 区間事象の集合

CSP による定義を以下に示す.

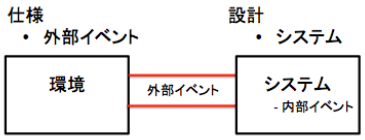

図 4 検証の前提

```
SECTION(S) =
 |||(start, end, event):s@
  EVENT_WITH_START_AND_END(start,end,event)
```

SECTION は区間事象の集合 (S) を引数とした関数として定義する.S で指定した区間事象のプロセスがインターリーブで合成されることを表している.

区間事象

区間事象は, 開始事象, 終了事象, 対象事象の 3 項組として定義する.

区間事象:= (開始事象, 終了事象, 対象事象)

区間事象の振る舞いは開始事象と終了事象の間で対象事象が起こる, もしくは起こらないと表される. CSP による定義を以下に示す.

```
EVENT_WITH_START_AND_END(start,end,event)
  = start ->
    (event -> end -> SKIP [] end -> SKIP)
```

EVENT_WITH_START_AND_END は区間事象の振る舞いを表したもので, 開始事象 (start), 終了事象 (end), 対象事象 (event) を引数とした関数として定義する.

4.3.2 仕様

仕様を記述するために必要な CSP ライブラリを用意した.

事象単位の振る舞い

事象単位の振る舞いを表すために必要な記述と CSP による定義を以下に示す.

区間内で事象のどれかが起こる

```
Or(P1, P2) = P1 |~| P2
```

区間内で事象が複数起こる

```
And(P1, P2) = P1 ||| P2
```

Or と And はどちらも,2 つの事象を引数とした関数として定義し,Or は指定した事象のうちどちらかが起こる,And は指定した事象が両方起こるという振る舞いを表している.

区間単位の振る舞い

区間単位の振る舞いは区間毎の振る舞いの関連を記述する. 区間単位の振る舞いの定義を以下に示す.

区間単位の振る舞い = Seq(仕様, 仕様)

5 区間振る舞いモデルに基づく検証の枠組み

5.1 概要

振る舞い検証では検査の対象はシステムの振る舞いを設計したものであり, 検証を行う際には仕様と設計を記述する必要がある. 前提として図 4 に表したように環境とシステムの境界のイベントを外部イベント, システム内部の動作を表すイベントを内部イベントと定義した. 検証では設計に対して入力を与えたときの振る舞いと, 仕様が外部イベントに着目した際に等しいことを確かめる.

5.2 仕様

仕様では外部イベントを用いて, 区間振る舞いモデルに基づく入力と, 入力に対して期待する結果を記述する.

入力

区間における入力は,4 章で定義した事象単位の振る舞いを用いて区間で起こる事象を記述する.

結果

入力イベントに対して期待するイベント系列を記述する. 結果も入力と同様に事象単位の振る舞いを用いて区間内で起こる事象を記述する. 入力と入力に対する結果の関係は, 区間単位の振る舞いで表現する.

5.3 設計

設計ではシステムを構成するそれぞれのプロセスの振る舞いを状態遷移機械で記述する. 対象とする状態遷移機械は, ミーリ機械を用いて表現する. ミーリ機械は

(S, Σ, Λ, T, G, s)

の 6 要素で構成されている. それぞれの要素の性質を以下に示す. それぞれの構成要素の意味は CSP ライブラリを用いて定義する.

- S:状態の有限集合
- Σ:イベントの有限集合
- Λ:アクションの有限集合
- T:遷移関数 (S × Σ → S)
- G:出力関数 (S × Σ → Λ)
- s:開始状態 (s ∈ S)

5.4 検証

振る舞い検証では, 設計に対して入力を与えたときの振る舞いと結果が等価であることを検証する. CSP では仕様が実現を満たすことを以下のように表現する.

```
仕様 [FD= TARGET(実現, 着目する事象)
TARGET(S, targets) =
  S \ diff(Events, targets)
```

\はイベントの隠蔽演算子で,diff は差集合を表す. 区間振る舞いモデルに基づく検証式は次のように定義する.

```
仕様 [FD= TARGET(検査対象, 外部イベント)
検査対象 = CONSTRAINT(設計,SYN_SEC, 入力)
```

6 事例：自動販売機

自動販売機システムの事例を用いて区間振る舞いモデルを用いた振る舞い検証の例を示す.

6.1 自動販売機システムの概要

図 5 は自動販売機システムのハードウェア構成を表したものである. 自動販売機システムには販売機能があり,

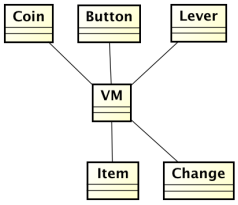

図 5 自動販売機システムの構成

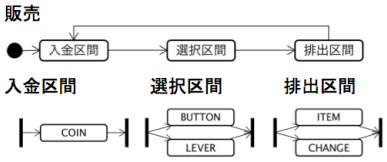

図 6 区間振る舞いモデル

ユーザーがボタンを押したら商品排出, レバーを引いたら返金を行う.VM コンポーネントではボタンもしくはレバーが引かれたときに, もう片方のセンサを停止させるような制御を行っているが, ボタンとレバーを同時に操作すると, 際どいタイミングでボタンとレバーの両方を検知する可能性がある.VM ではボタンとレバーが同時に押されたときには必ず商品排出するという制御を行っている.

検証ではボタンとレバーが同時に押されたときに制御が正しく行われていることを検証した.

6.2 区間振る舞いモデル

図 6 は自動販売機の区間振る舞いモデルを表したものである. 自動販売機システムには貨幣を投入する区間, 商品を選択する区間, 商品が排出される区間の 3 つの区間が存在する. ここで COIN,BUTTON,LEVER,ITEM,CHANGE はそれぞれ対象事象の 3 項組として定義される. 例として COIN の定義を以下に示す.

```
COIN = (env_ev.Coin.on,
        env_ev.Coin.off,
        send.Coin.inserted)
```

組み込みシステムにおけるセンサの開始, 終了はそれぞれハードウェアにおける ON,OFF に相当する.

6.3 振る舞い仕様

図 7 は際どいタイミングにおける仕様をアクティビティ図を用いて表したものである. 振る舞い仕様ではボタンとレバーを押したら商品を排出するという振る舞いを外部事象を用いて記述した. CSP による仕様記述を以下に示す.

```
SPEC=
```

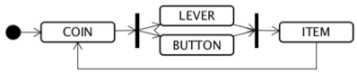

図 7 自動販売機システムの仕様

```
Seq(COIN,Seq(And(BUTTON,LEVER),ITEM));SPEC
```

6.4 検証

検証では外部事象に着目した際の仕様とシステムへの入力の結果が等価であることを検証した. 検証式は以下の通りである.

```
assert SPEC [FD= TARGET(PL_MODEL, ExtEvents)
```

7 考察

7.1 区間振る舞いモデルの意義

本研究では, 区間振る舞いモデルを用いてモデル検査における検査対象の抽象化の方法を提示した. 環境で生起するイベントを区間ごとに関心のあるイベントのみに限定することで, 取りうるイベントの組み合わせを削減した. また, 際どいタイミングにおける振る舞いを区間に集約することで, 区間振る舞いモデルのもとでは際どいタイミングにおける障害は起こらないことを保証できる.

7.2 関連研究

7.2.1 図式表現による支援

図式表現をモデル検査器を用いて検証する試みが行われている. 横川らは UML モデルからモデル検査用の検証コードを自動生成するツールを用いた検証支援を行っている [6]. これらは汎用の検証の枠組みを与えているが, 検証の基準となる仕様を記述する方法は提示されていない.

7.2.2 水平リファイメントの支援

リファイメントに基づいた設計手法として, 新たな機能を拡張することで仕様を充実させる水平リファイメントの方法が提案されている [5]. 水平リファイメントでは拡張前後の振る舞いの整合性の検証条件が与えられるが, 区間振る舞いモデルを用いることで追加した機能の振る舞いを区間の事象として追加するだけで整合性の検証が可能となる.

参考文献

[1] C.A.R.Hoare, *Communicating Sequential Processes,* Prentice-Hall, 1985.

[2] University of Oxford, “*FDR3 - The CSP refinement checker,* ” http://www.cs.ox.ac.uk/projects/fdr/.”, 2015.

[3] 張 漢明, 野呂 昌満, 沢田 篤史, ”同時性を考慮した並行システムの振る舞い検証に関する考察, ” 情報処理学会研究報告, vol.2015-EMB-36, no.13, 2015.

[4] 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, pp.72-86, 2006.

[5] 中島 震, “リファイメントプランニング, ” ソフトウェアサイエンス, 112(373), pp.13-18, Jan 2013.

[6] 横川 智教, 天嵜 聡介, 佐藤 洋一郎, 有本 和民, 宮崎 仁, “UML による組み込みソフトウェア設計の検証支援環境の開発, ” SEC journal, Vol.10, No.6, Mar 2015.