Vol.2021-EMB-57 No.3
2021/6/28

関数リアクティブプログラミング言語のための時間制約付きイベントの記述方式

堀 紗知子[1,a)]　森口 草介[1,b)]　渡部 卓雄[1,c)]

概要：本研究では，小規模組込みシステム向け関数リアクティブプログラミング言語 Emfrp において時間制約のあるイベントを検出する際の記述方式を提案する．例えばボタンを短く 1 回押す，ダブルクリックする，長押しするといった，ある事象が時間制約付きで複数回発生するようなイベントを検出することを考える．イベントに含まれる事象の回数を変更したり，同じ事象を観察する異なるイベントを区別して検出しようとすると，コードが複雑化し，かつ既存のコードへの変更が必要になる．このような問題の解決を目的とした時間制約付きイベントの記述方式と，その記述から Emfrp のコードを生成する手法を提案する．例題の記述を通して提案手法を利用したときの定義や変更のしやすさについて議論する．

1. はじめに

関数リアクティブプログラミング（*Functional Reactive Programming, FRP*）[1], [2] は時間によって変化する値（**時変値**）を取り扱うプログラミングパラダイムであり, GUI や組込みシステムなどのリアクティブシステムを取り扱うのに適している．Emfrp[2] は組込みシステム向けに設計された FRP 言語であり, メモリサイズが小さいマイクロコントローラ等でも動作可能である．一方, Emfrp は副作用をもたない純粋な関数型言語であり, 状態は時変値の直前値を用いて表現される．そのため, 後述するボタン操作の例のような, 時間制約のある動作の列として表される処理（本研究では**イベント**と呼ぶ）の記述や変更が煩雑になる．本研究ではこのような問題を解決するイベント記法の Emfrp への導入とその実現方法を提案する．

提案手法では，時変値とそれに関する時間制約の列としてイベントを記述するための構文を導入する．そしてその構文で書かれたイベントを, **時間オートマトン** [3] による表現を経由して Emfrp コードに変換する手法を提案する．これにより, 組込みシステムで用いられるような各種イベントの簡潔でモジュール性の高い記述が可能になり, システムの漸進的な開発が容易になる．

[1] 東京工業大学情報理工学院情報工学系
Department of Computer Science, School of Computing, Tokyo Institute of Technology, Meguro, Tokyo 152–8552, Japan
[a)] hori@psg.c.titech.ac.jp
[b)] chiguri@acm.org
[c)] takuo@acm.org

2. 提案するイベント記法

2.1 例題：クリックの検出

コード 1 は，1 個のボタンのクリックのされ方を検出する Emfrp プログラムの一部である．ボタンの状態は，押されている間のみ `True` となる `Bool` 型の時変値 `button` で表す．ここでは 4 種類のクリック（シングルクリック, ダブルクリック, トリプルクリック, ロングクリック）を区別して検出したい．時変値 `singleclick` の値はシングルクリックの検出を表している．これは，ボタンを押してから離すまでの時間が 500ms 以内で，離してから 500ms 以内に再びボタンを押さなかったときに `True` となる．時変値 `t0`, `t1`, `t2`, `t3` はボタンを押した時刻の記録に用いられている．

コード 1　クリック検出

```
....
node init[(0, 0, 0, 0)] (t3, t2, t1, t0) =
  if !button@last && button
  then (millis, t3@last, t2@last, t1@last)
  else (t3@last, t2@last, t1@last, t0@last)
....
node init[False] singleclick =
  (t4 - t3 >= 500) && (t4 != t4@last) &&
    (t3 - t2 >= 500) && (t5 >= t3) && (t5 - t3 < 500)
....
```

2.2 提案手法による例題の記述

コード 1 のように記述した場合, 検出したいイベントの

コード 2　提案手法による記述

```
1 event singleclick[1, 500, button]
2 event doubleclick = singleclick + [1, 500, button]
3 event tripleclick = doubleclick + [1, 500, button]+
4 event longclick = singleclick - [1, 500, !button]
```

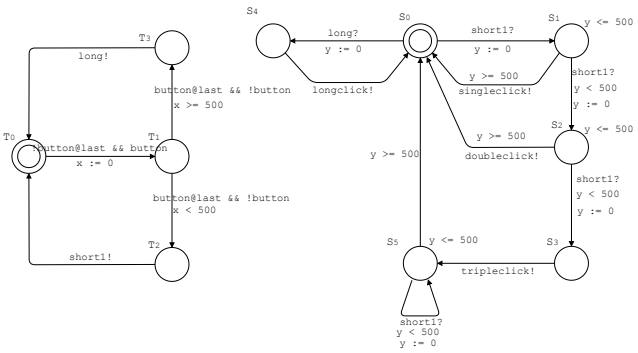

図 1　クリック例題のオートマトン変換

数に応じて時間記録のための時変値が増え, 各イベント検出の条件も増加していく. また, 異なるイベントの検出を追加する際, 既に記述したイベントに対しても変更を要する.

本研究ではこのような問題を解決するイベント記述方式を提案する．提案方式によって例題を記述したものがコード 2 である.

イベントはキーワード event によって Bool 型の時変値として定義される. コード 2 の 1 行目はイベント singleclick の定義である．定義内の [k, t, c] はイベントの発生条件を表しており，c が時変値に関する制約条件を，t と k は最大継続時間と回数をそれぞれ指定する．1 行目の [1, 500, button] は button が 500 単位時間 (ここでは 500ms) 以下の間 True であることを表している．+は続くイベントが t 以内に発生することを表し, -は t 以上経過してから発生することを表す. 直前のイベントで指定した c に対して!c を指定した場合, c が t 以上継続することを意味する（コード 2 の 4 行目 longclick の例）.

この記述方法を利用すると, イベント追加時に既に記述されていた部分の変更は不要になる.

3. 実装手法

提案手法による記述を時間オートマトンへの変換を通じて Emfrp コードに変換する.

3.1 時間オートマトンへの変換

コード 2 を時間オートマトンに変換すると, 図 1 のようになる. 4 つのイベント記述を, ボタンが押されている時間を計測するためのオートマトンと, それを用いてイベントの種類を検出するためのオートマトンの 2 つのオートマトンに変換する.

コード 3　時間オートマトンから Emfrp コードへの変換

```
1  ....
2  node init[0] state =
3    if (state@last == 0) && (state_d@last == 2) then 1
4    else if (state@last == 0) && (state_d@last == 3) then 4
5  ....
6  node init[0] y =
7    if (state@last == 0) && (state_d@last == 2) then 0
8    else if (state@last == 1) && (y@last < 500)
9      && (state_d@last == 2) then 0
10   ....
11   else y@last + millis - millis@last
12 ....
13 node init[False] singleclick =
14   (state@last == 1) && (y@last >= 500)
15 ....
```

3.2 Emfrp コードへの変換

2 種類のオートマトンはどちらも同じように Emfrp コードに変換される. 現在のロケーションとクロック変数を表すための時変値を用意する. 時間オートマトン生成時にはロケーションに固有の番号が割振られる. この番号を用いて, ロケーションを Int 型で表している. これらの時変値は, 生成された時間オートマトンの遷移条件にしたがって変化する. 各時変値の直前値を比較することでイベントを検出する.

図 1 を Emfrp コードに変換したものの一部がコード 3 である. state_d, state は図 1 の各オートマトンのロケーションを, y は右のオートマトンのクロック変数を表している.

4. 結論

Emfrp における, 時間制約のある動作の列として表されるようなイベントの記述方式を提案し，イベント記述から時間オートマトンを介して Emfrp コードへ変換する手法を示した．提案方式により，イベントの簡潔かつモジュール性の高い記述が可能になる．提案手法にもとづく変換系の実装は今後の課題である.

謝辞　本研究の一部は JSPS 科研費 21K11822 および 19K20245 の助成を受けている.

参考文献

[1] Elliott, C. and Hudak, P.: Functional Reactive Animation, *2nd ACM SIGPLAN International Conference on Functional Programming (ICFP 1997)*, ACM, pp. 263–273 (1997).
[2] Sawada, K. and Watanabe, T.: Emfrp: A Functional Reactive Programming Language for Small-Scale Embedded Systems, *Companion Proceedings of the 15th International Conference on Modularity*, ACM, pp. 36–44 (2016).
[3] Alur, R. and Dill, D. L.: A Theory of Timed Automata, *Theoretical Computer Science*, Vol. 126, No. 2, pp. 183–235 (1994).