小規模組込みシステム向けFRP言語への文脈指向プログラミング機構の導入

渡部 卓雄[1,a)]

概要：小規模組み込みシステム向けに設計された関数リアクティブプログラミング (FRP) 言語 Emfrp のための文脈指向プログラミング機構を提案する．提案方式では時変値の値にもとづく層の活性化機構を導入し，実行時文脈の変化に伴う振る舞いの動的な変更を FRP の枠組み内で宣言的に記述できるようにしている．これにより，小規模組込みシステムにおける適応動作のモジュール化が可能になる．

キーワード：関数リアクティブプログラミング, 文脈指向プログラミング, 小規模組込みシステム

Introduction of a Context-Oriented Programming Mechanism to an FRP Language for Small-Scale Embedded Systems

Takuo Watanabe[1,a)]

Abstract: This paper briefly describes the design of a simple context-oriented programming extension to Emfrp, a purely functional reactive programming language for resource-constrained embedded systems. The proposed extension provides a layer mechanism along with a a feature to describe events along with the layer activation. An example shows that the extension can eliminate typical cross-cutting code that often appear in plain Emfrp programs.

Keywords: Functional Reactive Programming, Context-Oriented Programming, Small-Scale Embedded Systems

1. はじめに

本研究は，小規模組込みシステム向けに設計された関数リアクティブプログラミング言語への文脈指向プログラミング機構の導入が，(a) 適応動作記述におけるモジュール性の改善，および (b) 適応動作の安全性・信頼性向上に寄与することを明らかにすることを目的とする．現在，上記目的 (a) に向けて筆者らが開発している FRP 言語 Emfrp への文脈指向プログラミング (COP) 機構の導入を提案している [2]．本ポスター発表では，提案した COP 機構について概略[*1]を述べ，デモを行う．

関数リアクティブプログラミング (Functional Reactive Programming, FRP) は，**時変値** (Time-Varying Value) と呼ばれる抽象化機構によって時間と共に変化する値とその関係を宣言的に記述することで，組込みシステムに代表されるリアクティブシステムの効果的な記述を支援するプログラミングパラダイムである．

著者らは，マイクロコントローラに代表される小規模組込みシステムを対象とした純粋 FRP 言語 Emfrp [1] を設計・実装し，いくつかの例題を通してその有用性を明らかにしてきた．Emfrp はリソースの限られた環境で実行することを前提として設計されている．具体的には静的型付けに加え，時変値を一級データではなく名前で参照することや再帰の禁止といった言語設計上の制約を設けている．これにより，プログラムが利用する記憶領域の大きさは静的に決定でき，また push 型と呼ばれる実行方式の採用により生成されるコードサイズを抑えている．これらの設計により表現力を大幅に落とすことなく小規模組み込みシステム記述に適した言語機構を提供しているが，実行時情報にもとづく柔軟な振る舞いの表現に制約が生じる．例えば組込みシステムの実行において重要な，変化する環境への適応動作を適切にモジュール化して記述することが難しい．

[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
[a)] takuo@acm.org

[*1] 詳細については [2] および登壇発表予稿 [4] を参照されたい．

このような問題に対処することを目的として，筆者らは Emfrp における自己反映計算 (reflection) の機構を提案し，ロボットの適応動作記述への応用等を通してその有効性を示した [3]．これによって適応動作の宣言的な記述が可能になったが，その記述はメタレベルモジュール内で行うため，オブジェクトレベルで記述されるアプリケーション動作とは分離される．しかし必ずしも汎用的ではない，アプリケーションに特化した適応動作を記述したい場合，アプリケーションと完全に分離した記述が必要になることで却って記述性や可読性の低下につながることがある．

適応的な動作のモジュール化を促進するプログラミングパラダイムとして**文脈指向プログラミング** (Context-Oriented Programming, COP) がある．一般的な自己反映計算機構とは異なり，文脈（実行時情報）に依存した処理のモジュール化に特化した言語機構を導入することで，適応動作を見通しよく記述することを可能にする．多くの COP 言語は，Java などの既存の言語に文脈に依存する処理を分離して記述するための**層** (layer) と呼ばれる言語機構，および実行時情報に依って適切な層を活性化する実行時機構によって実現されている．

2. Emfrp への COP 機構の導入

Emfrp [1] は小規模組み込みシステム向けに設計された関数リアクティブプログラミング言語である．本研究では，層 (layer) を記述する構文を Emfrp に追加することで，文脈指向プログラミング (COP) 機構の導入を行う．

図 1 は COP 拡張 [2] を施した Emfrp のプログラム例である．これは温度・湿度センサによるファンの制御プログラムであり，不快指数が規定値以上になっている間ファンのスイッチを ON にする．この基本機能に加え，(a) 不快指数 `di` の値が閾値 `th` 近辺で変化した際にファンのスイッチが頻繁に ON/OFF を繰り返さないようにするヒステリシス制御，(b) ファンが現在までに ON になった合計時間を LCD に表示する機能，および (c) ファンのモーターを保護するため連続して 2 時間以上 ON になった際に強制的に 30 分間 OFF にする追加機能が導入されている．

各追加機能は層として定義されている．具体的には (a) は層 `HYSTERESIS`（20-22 行目）で，(b) は層 `CTIME`（25–27 行目）で，(c) は層 `OTIME` と `SHUTDOWN`（30–39 行目）でそれぞれ実現されている．それぞれの層は個別に導入や削除が可能であり，層の独立性は高い．また基本機能の記述である 10–17 行目については変更の必要がない．

COP 機構を用いずに同等の動作を記述すると，一般に実行時文脈（例えばファンが ON になっている文脈やモーター保護タイマーが有効である文脈等）に依存する動作は条件式（`if`式）を伴って記述される．しかし文脈に依存する範囲が広くなることで，同様の条件を表す式がプログラム中に散在することになる．このような状況は横断的関心事とみなすことができ，プログラムのモジュール性・可読性を低下させる．提案手法では，文脈に依存した振る舞いの定義を層としてモジュール化することで，実行時文脈に依存した動作の宣言的かつ簡潔な記述を可能にしている．

```
1  module FanController2  # module name
2  in  tmp   : Float,     # temperature sensor
3      hmd   : Float,     # humidity sensor
4      clock : Int        # POSIX time system clock
5  out fan   : Bool,      # fan switch
6      ctime : Int        # LCD for time
7  use Std                # standard library
8
9  # discomfort (temperature-humidity) index
10 node di =
11     0.81 * tmp + 0.01 * hmd * (0.99 * tmp - 14.3) + 46.3
12
13 # fan switch
14 node init[False] fan = di >= th
15
16 # threshold
17 node th = 75.0
18
19 # hysteresis behavior
20 layer HYSTERESIS where fan
21   enter node th = @proceed - 0.5
22   exit node th = @proceed + 0.5
23
24 # cumulative operating time
25 layer CTIME where fan
26   enter node init[0] ctime0 = clock - ctime@last
27   retain node init[0] ctime = clock - ctime0
28
29 # continuous operating time
30 layer OTIME where fan
31   enter node init[0] otime0 = clock
32   retain init[0] node otime = clock - otime0
33
34 # stop the fan for 30 minutes
35 # after 2 hours continuous operation
36 layer SHUTDOWN where (otime > 7200 || timer > 0)
37   enter node init[0] timer0 = clock + 1800
38   retain node init[0] timer = timer0@last - clock
39   node fan = False
```

図 1 COP 拡張を用いたファン制御プログラム

参考文献

[1] Sawada, K. and Watanabe, T.: Emfrp: A Functional Reactive Programming Language for Small-Scale Embedded Systems, CROW 2016, ACM, pp. 36–44, doi:10.1145/2892664.2892670 (2016).
[2] Watanabe, T.: A Simple Context-Oriented Programming Extension to an FRP Language for Small-Scale Embedded Systems, COP 2018, ACM, doi:10.1145/3242921.3242925 (2018).
[3] Watanabe, T. and Sawada, K.: Towards Reflection in an FRP Language for Small-Scale Embedded Systems, LASSY 2017, ACM, doi:10.1145/3079368.3079387 (2017).
[4] 渡部卓雄: 小規模組込みシステム向け FRP 言語への文脈指向プログラミング機構の導入, 組込みシステムシンポジウム (ESS 2018), 情報処理学会 (2018).