小規模組み込みシステム向けFRP言語とその自己反映機構

渡部 卓雄[1,a)]

概要：マイクロコントローラ等の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) 言語とそのための自己反映計算機構を提案する．提案方式では，FRP 言語の特徴である時変値を介した実行系内部へのアクセスを可能にすることで，自己反映計算もリアクティブな操作として実現されている．本機構の導入により，小規模システムでの実行を考慮して静的に実現されている言語の実行系に，ある程度の柔軟性と適応性を与えることが可能になる．

キーワード：関数リアクティブプログラミング, 組み込みシステム, 自己反映計算

An FRP Language for Small-Scale Embedded Systems and its Reflection Mechanism

Takuo Watanabe[1,a)]

Abstract: This poster presentation will describe a simple reflection mechanism for a functional reactive programming language designed for resource-constrained embedded systems. The proposed mechanism allows accessing the internal structure of the runtime system of the language via time-varying values. This means that reflective operations are also reactive. The mechanism introduces a certain degree of flexibility and adaptability to the statically designed runtime system of the language.

Keywords: Functional Reactive Programming, Embedded Systems, Reflection

リアクティブシステムとは，離散的なイベントや連続的な外界の状態変化などの入力に対して，応答の生成および自身の状態の更新をし続けるシステムである．GUI や組み込みシステムはリアクティブシステムの典型例である．

一般にリアクティブシステムに対する入力は，与えられる順序およびタイミングはあらかじめ予測できない．そのため，リアクティブシステムのプログラミングでは，コールバック，イベントループ，ポーリング等が用いられる．これらは一般的な手続き型プログラミング言語においても実現可能である一方，コードの細分化を招き可読性低下の原因となる．

リアクティブプログラミング *(Reactive Programming)* は，**時変値** *(time-varying value)* やイベントストリームなどの抽象化機構を用いて入力およびそれらに依存する値を表現することで，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 [1]．特に**関数リアクティブプログラミング** *(Functional Reactive Programming, FRP)* は，関数プログラミングに時変値を導入することでリアクティブシステムの宣言的な記述を可能にする．

時変値は時間と共に連続的に変化する値を抽象化したものである．歴史的には，時変値の概念は Haskell の対話型アニメーションライブラリ Fran[5] において導入された．そしてこれを皮切りに，例えば [10] や [7] 等，FRP の研究は主に Haskell 上のライブラリ（あるいは内部 DSL）の研究開発に関連して行われてきた．これらにおいては，時変値を Haskell のデータとしてナイーブに表現した場合に発生する時間漏洩 (time leak) や空間漏洩 (space leak) といった現象を回避する必要があり，そのためにアロー (arrow) と呼ばれるデータ型 [8] の導入等が行われてきた．

アローにもとづく FRP (Arrowized FRP) では，時変値自体は隠蔽し，その代わりに時変値から時変値への関数（シグナル関数）をアロー型のデータとして導入する．そしてシグナル関数を合成するコンビネータ群を用いること

[1] 東京工業大学 情報理工学院 情報工学系
Department of Computer Science, Tokyo Institute of Technology
[a)] takuo@acm.org

で，時間・空間漏洩を回避しつつ時変値間の依存関係の記述を可能にする．

一方，時変値の型を（ユーザ定義型ではなく）組み込みの型とすることで，アローにもとづく FRP と同等の記述力を持ち，かつ簡潔な記法を持つ言語を実現できる．その手法をとった言語の一例として GUI 記述のための言語 Elm[3] がある[*1]．Elm では型 t の値をもつ時変値は型 Signal t を持つ．これは組み込み型であり，引数の型 t として時変値および時変値を含むデータ型は用いることはできない．

FRP の初期の研究においてロボットへの応用 [7], [10] がなされたことが示すように，FRP は組み込みシステムの記述にも適用可能である．そして現在までに，特にマイクロコントローラ等で実現される小規模組み込みシステムのための FRP 言語として E-FRP[13], Flask[9], Céu[11], Juniper[6] などが提案されている．

著者らは，いくつかのマイクロコントローラをターゲットとした純粋関数型言語である Emfrp を設計・実装し，例題を通してその有用性を明らかにしている [12]．Emfrp では時変値を一級データとして扱わず，必ず名前によって参照するというプログラミング上の制約を設けている．また一般的な再帰呼び出しおよび再帰的なデータ構造の利用を禁止する代わりに，任意の時変値の直前値を参照できる言語機構を導入している．以上によってプログラムが利用する記憶領域の大きさを静的に確定できるようになっている．また実行系としてシンプルな push 型 [4] を採用しているため，生成されるコードサイズも小さい．

このように Emfrp ではプログラムの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に適した言語機構を提供している．しかしその一方でプログラムの実行方式に関する柔軟性が不足しており，例えば pull 型の実行方式を採用すれば避けられるような不要な計算を行ってしまう場合がある．

このような問題は push 型と pull 型の実行を混在させることで解決できるが [4]，そうすることで実行系が複雑になりサイズが肥大化するという問題が生じる．筆者らは，簡単な自己反映計算機構を Emfrp の実行系に導入することを提案し，この問題を解決可能であることを示した [14]．この自己反映機構では，Emfrp のモジュール（プログラムの構成単位）とその実行を表現するモジュール（メタモジュール）を導入している．そして時変値を介した自己反映動作によって実行系の動作をカスタマイズし，アプリケーションに応じた適切な実行を可能にしている．

本ポスター発表では，[14] において示された自己反映機構と，その実装について概略を述べる．

[*1] バージョン 0.17 以降では時変値の利用をやめている [2].

参考文献

[1] Bainomugisha, E., Carreton, A. L., Van Cutsem, T., Mostinckx, S. and De Meuter, W.: A Survey on Reactive Programming, *ACM Computing Surveys*, Vol. 45, No. 4, p. 52 (online), DOI: 10.1145/2501654.2501666 (2013).

[2] Czaplicki, E.: A Farewell to FRP: Making signals unnecessary with The Elm Architecture (2016).

[3] Czaplicki, E. and Chong, S.: Asynchronous Functional Reactive Programming for GUIs, *34th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2013)*, ACM, pp. 411–422 (online), DOI: 10.1145/2499370.2462161 (2013).

[4] Elliott, C.: Push-Pull Functional Reactive Programming, *Proceedings of the 2nd ACM SIGPLAN Symposium on Haskell*, pp. 25–36 (online), DOI: 10.1145/1596638.1596643 (2009).

[5] Elliott, C. and Hudak, P.: Functional Reactive Animation, *2nd ACM SIGPLAN International Conference on Functional Programming (ICFP 1997)*, ACM, pp. 263–273 (online), DOI: 10.1145/258949.258973 (1997).

[6] Helbling, C. and Guyer, S. Z.: Juniper: A Functional Reactive Programming Language for the Arduino, *4th International Workshop on Functional Art, Music, Modelling, and Design (FRAM 2016)*, ACM, pp. 8–16 (online), DOI: http://dx.doi.org/10.1145/2975980.2975982 (2016).

[7] Hudak, P., Courtney, A., Nilsson, H. and Peterson, J.: Arrows, Robots, and Functional Reactive Programming, *Advanced Functional Programming*, Lecture Notes in Computer Science, Vol. 2638, Springer-Verlag, pp. 159–187 (online), DOI: 10.1007/978-3-540-44833-4_6 (2003).

[8] Hughes, J.: Generalising monads to arrows, *Science of Computer Programming*, Vol. 37, No. 1–3, pp. 67–111 (online), DOI: 10.1016/S0167-6423(99)00023-4 (2000).

[9] Mainland, G., Morrisett, G. and Welsh, M.: Flask: Staged Functional Programming for Sensor Networks, *International Conference on Functional Programming (ICFP 2008)*, ACM, pp. 335–346 (online), DOI: 10.1145/1411204.1411251 (2008).

[10] Pembeci, I., Nilsson, H. and Hager, G.: Functional Reactive Robotics: An Exercise in Principled Integration of Domain-Specific Languages, *4th International Conrefernce on Principles and Practice of Declarative Programming (PPDP 2002)*, ACM, pp. 168–179 (online), DOI: 10.1145/571157.571174 (2002).

[11] Sant'Anna, F., Ierusalimschy, R. and Rodriguez, N.: Structured Synchronous Reactive Programming with Céu, *14th International Conference on Modularity (Modularity 2015)*, ACM, pp. 29–40 (online), DOI: 10.1145/2724525.2724571 (2015).

[12] Sawada, K. and Watanabe, T.: Emfrp: A Functional Reactive Programming Language for Small-Scale Embedded Systems, *Modularity 2016 Constrained and Reactive Objects Workshop (CROW 2016)*, ACM, pp. 36–44 (online), DOI: 10.1145/2892664.2892670 (2016).

[13] Wan, Z., Taha, W. and Hudak, P.: Event-Driven FRP, *Practical Aspects of Declarative Langauges*, Lecture Notes in Computer Science, Vol. 2257, Springer-Verlag, pp. 155—172 (online), DOI: 10.1007/3-540-45587-6_11 (2002).

[14] Watanabe, T. and Sawada, K.: Towards Reflection in an FRP Language for Small-Scale Embedded Systems, *Programming 2017 Workshop on Live Adaptation of Software SYstems (LASSY 2017)*, ACM, (online), DOI: 10.1145/3079368.3079387 (2017).