xv6の構成要素の継続の分析

清水 隆博[1,a)]　河野 真治[2,b)]

概要：OS 自体そのものは高い信頼性が求められるが、 OS を構成するすべての処理をテストするのは困難である。テストを利用して信頼性を高めるのではなく、 OS の状態を状態遷移を基本としたモデルに変換し形式手法を用いて信頼性を高めたい。
状態遷移単位での記述に適した言語である CbC を用いて、小さな unix である xv6 kernel の書き換えを行っている。このためには現状の xv6 kernel の処理がどのような状態遷移を行うのかを分析し、継続ベースでのプログラミングに変換していく必要がある。本稿では xv6kernel の構成要素の一部に着目し、状態遷移系の分析と状態遷移系を元に継続ベースで xv6 の再実装を行う。

1. OS の信頼性

様々なアプリケーションは OS の上で動作するのが当たり前になってきた。アプリケーションの信頼性を向上させるのはもとより、 土台となる OS 自体の信頼性は高く保証されていなければならない。OS そのものも巨大なプログラムであるため、 テストコードを用いた方法で信頼性を確保する事が可能である。しかし並列並行処理などに起因する動かしてみないと発見できないバグなどが存在するため、 テストで完全にバグを発見するのは困難である。また、OS を構成する処理も巨大であるため、 これら全てをテスト仕切るのも困難である。テスト以外の方法で OS の信頼性を高めたい。

数学的な背景に基づく形式手法を用いて OS の信頼性を向上させることを検討する。OS を構成する要素をモデル検査してデッドロックなどを検知する方法や、 定理証明支援系を利用した証明ベースでの信頼性の確保などの手法が考えられる。形式手法で信頼性を確保するには、 まず OS の処理を証明などがしやすい形に変換して実装し直す必要がある。これに適した形として、 状態遷移モデルが挙げられる。OS の内部処理の状態を明確にし、 状態遷移モデルに落とし込むことでモデル検査などを通して信頼性を向上させたい。既存の OS はそのままに処理を状態遷移モデルに落とし込む為には、 まず既存の OS の処理中の状態遷移を分析する必要がある。分析の結果を定理証明支援系などによって証明を行うか、 仕様記述言語で再実装することで仕様の整合性を検証する事が可能である。しかしこれらの方法では、 実際に動く OS と検証用の実装が別の物となってしまうために、 C 言語などの実装の段階で発生するバグを取り除くことができない。実装のソースコードと検証用のソースコードは近いセマンティクスでプログラミングする必要がある。

実装用の言語と証明用の言語の両方に適した言語として Continuation Based C(CbC) がある。現在小さな unix である xv6 kernel を状態遷移を基本とした単位でのプログラミングに適した言語、 Continuation Based C を用いて再実装している。再実装の為には、 既存の xv6 kernel の処理の状態遷移を、継続を用いたプログラムに変換していく必要がある。

[1] 琉球大学大学院理工学研究科情報工学専攻
[2] 琉球大学工学部工学科知能情報コース
[a)] anatofuz@cr.ie.u-ryukyu.ac.jp
[b)] kono@ie.u-ryukyu.ac.jp

参考文献

[1] Knuth, D. E.: *Fundamental Algorithms*, Art of Computer Programming, Vol. 1, chapter 2, pp. 371–381, Addison-Wesley, 2nd edition (1973).
[2] Schwartz, A. J.: Subdividing Bézier Curves and Surfaces, *Geometric Modeling: Algorithms and New Trends* (Farin, G. E., ed.), SIAM, Philadelphia, pp. 55–66 (1987).
[3] Baraff, D.: Curved Surfaces and Coherence for Non-penetrating Rigid Body Simulation, *SIGGRAPH '90 Proceedings* (Beach, R. J., ed.), Dallas, Texas, ACM, Addison-Wesley, pp. 19–28 (1990).
[4] Nakashima, H. et al.: OhHelp: A Scalable Domain-Decomposing Dynamic Load Balancing for Particle-in-Cell Simulations, *Proc. Intl. Conf. Supercomputing*, pp. 90–99 (online), DOI: http://doi.acm.org/10.1145/1542275.1542293 (2009).
[5] Adobe Systems Inc.: *PostScript Language Reference Manual*, Reading, Massachusetts (1985).
[6] 山下義行：文脈自由文法への否定の導入，修士論文，筑

波大学大学院工学研究科 (1989).
[7] Weihl, W.: Specification and Implementation of Atomic Data Types, PhD Thesis, MIT, Boston (1984).
[8] Institute for New Generation Computer Technology: *Proc. Intl. Conf. on Fifth Generation Computer Systems*, Vol. 1 (1992).
[9] Aredon, I.: TEX 独稽古，Seminar on Mathematical Sciences 13, Department of Mathematics, Keio University, Yokohama (1989).
[10] 情報処理学会：コンピュータ博物館設立の提言，情報処理学会（オンライン），入手先 ⟨http://www.ipsj.or.jp/03somu/teigen/museum200702.html⟩ (参照 2007-02-05).
[11] 情報処理学会論文誌編集委員会：「情報処理学会論文誌 (IPSJ Journal)」原稿執筆案内，情報処理学会（オンライン），入手先 ⟨http://www.ipsj.or.jp/08editt/journal/shippitsu/ronbunJ-prms.pdf⟩ (参照 2010-10-28).
[12] Kay, A.: Welcome to Squeakland, Squeakland (online), available from ⟨http://www.squeakland.org/community/biography/alanbio.html⟩ (accessed 2007-04-05).
[13] Nakashima, H.: A WEB Page, Kyoto University (online), available from ⟨http://www.para.media.kyoto-u.ac.jp/ nakashima/a.web.page.of.long.url/⟩ (accessed 2010-10-30).
[14] Nakashima, H.: Another WEB Page, Kyoto University (online), available from ⟨http://www.para.media.kyoto-u.ac.jp/ nakashima/a.web.page.of.much.longer.url/⟩ (accessed 2010-10-30).