インタラクションの構成単位としてのアバカス

須崎 宏文[1,a)]

概要：計算を補助する装置としての歴史を持つアバカスはコンピュータの普及によって社会での役割を失っている．コンピュータは記号によって表現された情報を高速で処理するというアーキテクチャを持ち，計算を含む多くの情報処理を人間のために実現している．記号を用いることなく利用可能なアバカスと比べるならば，従来のコンピュータは記号の利用という制約を人間に与えることで発展を続けている．本稿では，アバカスをコンピュータとして扱う拡張を行い，記号の利用という制約のないコンピューティングを提案する．そして，アバカスを構成単位とするインタラクションが人類の知的活動にもたらす可能性について考察する．

1. はじめに

人類の知的活動は文字のような情報の表現を可能にする記号を基礎とし，記号によって知識の保存と伝達を可能にすることで，一人の人間では限られた発見を人類全体で共有することを可能にしている．記号による表現は法則の発見をもたらし，法則の体系化により科学の進歩がもたらされている．そして，法則を人為的に利用することで，設計者である人類よりも高速な記号による情報処理を実現するコンピュータが発明される．EDVAC[1] に関して考案されたコンピュータの基本的なアーキテクチャは，現代では入力装置，出力装置，記憶装置，制御装置，演算装置という 5 つの要素によって説明され，その機能は記憶装置に保存されている OS(Operating System) を，制御装置と演算装置を担う CPU(Central Processing Unit) が処理することでユーザに提供されている．

コンピュータが発明される以前から用いられているアバカス [2] は数字のような記号を用いることなく人間の計算の補助を行う装置である．コンピュータはアバカスよりも計算の速度や正確において優れているが，その処理はユーザが記号による表現を利用しなければ成立しないという制約を持つものである．この制約は当然のものとなっているが，仮に存在しないとすれば，コンピュータは異なる発展を遂げる可能性を持っている．本稿は，アバカスをコンピュータとして扱う拡張を行い，従来と異なるコンピューティングでのインタラクションにおける構成単位としてのアバカスを提案する．

[1] 個人事業主 Abacist
[a)] suzaki@homeodomain.net

2. アーキテクチャ

2.1 アバカスの拡張

CPU を用いずに計算を補助するアバカスは人間を CPU として扱う発明であり，人間はアバカスを，手に対する入力装置，計算の過程を保存する記憶装置，目に対する出力装置という 3 つの要素を統合した装置として扱っている．アバカスがコンピュータとなる拡張として内部に，データをファイルとして保存する記憶装置と，保存されたファイルがプログラムである場合に処理を担う CPU を追加する．そして，プログラムの処理は複数のアバカスをユーザが結合した場合に可能とする．アバカスの物理的な構造は図 **1**

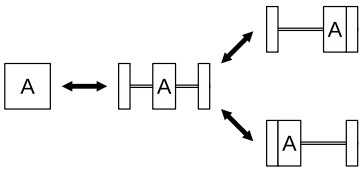

図 **1** アバカスの物理的な構造

のように左の初期状態となる立方体から，中央の状態へと変形し，右上と右下が示す 2 進数の選択が可能となるものとする．珠には内部のデータとは無関係のアバカスを呼称するための記号を便宜的に表記し，その移動が人間による場合を入力，CPU による場合を出力と定義する．

2.2 置き換え

記号を用いる代表的なコンピュータの操作方法である CUI(Character-based User Interface) をアバカスに置き換える方法を説明するため，図 **2** と図 **3** を例として示してい

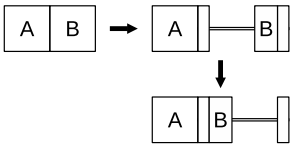

図 **2**　アバカスによるファイルの削除の例

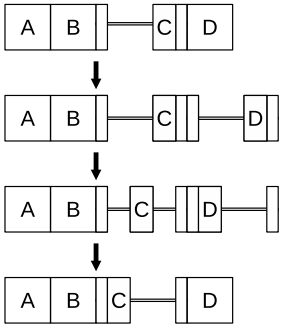

図 **3**　アバカスによるファイルのコピーの例

る．図 2 ではアバカス A は 1 つのファイルを保存し，アバカス B はファイルの有無を判定するプログラムを保存している．アバカス A がファイルを保存していなければ，アバカス B は反応せず，左上の状態から変化しないが，保存していれば，アバカス B は右上の状態へ変化する．ここで，ユーザがアバカス B を右下の状態へと操作によって変更することで，アバカス A のファイルを削除することが可能となる．図 3 ではアバカス A とアバカス B に異なるファイルが保存され，アバカス C はこれら 2 つのファイルが同一であるかを比較するプログラムを，アバカス D はアバカス A からアバカス B へとファイルをコピーするプログラムを保存している．一段目では，アバカス C はアバカス A とアバカス B が保存する 2 つのファイルを比較し，異なるために珠を右に移動させる．二段目では，アバカス D がアバカス C の状態を参照して起動する．三段目では，アバカス D をユーザが操作することでコピーを実行し，アバカス C はコピーの進捗を示すインジケータとなる．四段目では，コピーが完了し，アバカス C は 2 つのファイルが同一であることを示し，アバカス D が操作不可能な初期状態に戻る．

2.3　長所と短所

アバカスを利用することで，記号によって表現されるコマンドやデータを保存するパス，ファイル名を記憶することのないコンピューティングは実現されるが，次のような長所と短所を生じる．長所は，ユーザ自身が CPU となるために OS を必要とせず，CPU はアバカス内の特定のプログラムを処理するだけの簡素なものであるため，作成が容易になるという点である．短所は，データを分散処理することに特化した UI(User Interface) であり，ディスプレイの利用を前提として成立しているプログラムの作成が困難となる点である．よって，アバカスによって記号を利用する方法が，アバカスの発展には必要となる．

3.　コンピューティング

3.1　ハードウェア

多くのソフトウェアを利用可能にする OS はコンピュータの入力装置としてキーボードとマウスを前提に開発されている．キーボードはキーを押した状態と押していない状態という 2 つの状態から記号を入力する機能を持つため，入力する記号を珠に表記したアバカスによって代用可能である．マウスはディスプレイを表す平面上の座標を入力する機能を持つため，垂直方向と水平方向についての 2 つのアバカスの珠のアナログな位置を入力することで代用可能となる．よって，ハードウェアとしてのキーボードとマウスはアバカスの一種として作成可能であり，アバカスがプログラムの作成に必要な従来のソフトウェアを利用する前提が成り立つ．

3.2　ソフトウェア

出力装置としてのアバカスは文章や画像を表示するために用いられるディスプレイと同等の機能を持つことは困難であるが，CUI と同様に GUI(Graphical User Interface) に対しては，マウスで操作するコンポーネントをアバカスに置き換えることは可能である．すなわち，ディスプレイにはアバカスでは表現できない情報だけを表示し，メニューやボタンはデジタル入力を行うアバカスに，スクロールバーはアナログ入力を行うアバカスに置き換えるというようなソフトウェアごとの操作に適した利用方法を提案可能である．この提案は従来の図 **4** のように 1 つの CPU を利

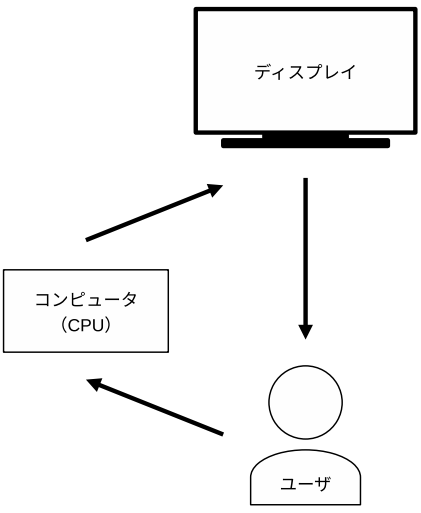

図 **4**　アバカスを用いないユーザによるコンピューティングの例

用するコンピューティングとは異なる図 **5** のような複数の CPU を利用するコンピューティングを新たにもたらす．これらを比較した場合における矢印によって示される情報の流れの違いとは，図 4 では 1 つのコンピュータ内部の CPU を中心に生じる情報の流れが，図 5 ではユーザを中

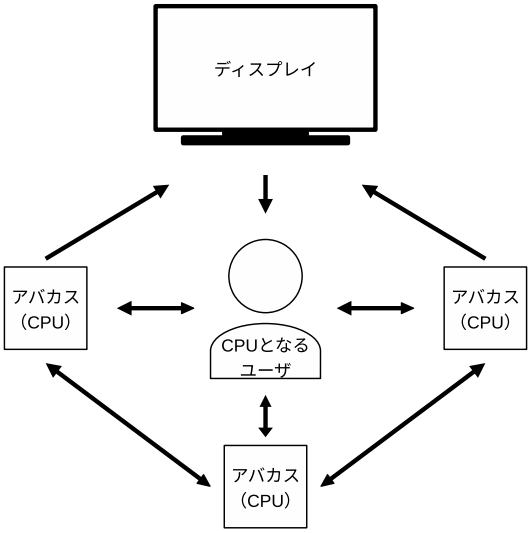

図 **5** アバカスによりユーザを CPU とするコンピューティングの例

心に生じている点である．また，図 4 では OS を前提としてソフトウェアの実行が実現されるために，1 つのソフトウェアしか利用しないとしても OS を起動しなければならないという点やソフトウェアが OS に対応して作成されていなければ実行できないという点をユーザに制約としてもたらしているが，このような制約は図 5 では生じることはない．なぜならば，アバカスには CPU が内蔵され，個々の CPU がソフトウェアの実行を独立して担うことができるからである．ただし，OS によって管理されているソフトウェア間の情報共有が不可能であるため，左のアバカスと右のアバカスを下のアバカスがコピーのような処理で結ぶというような OS の役割をユーザが担わなければならないという新たな制約を生じる．

3.3 サイクル

コンピューティングに不可欠なプログラムの作成とは過去での入力を原因，未来での出力を結果とする因果関係を記号によって記述することである．従来のコンピュータでは，OS は開発者に依存した記号が保存されたものであるために，ユーザは OS を作成する必要がない反面，個々の OS の使い方を学ぶ必要がある．アバカスによってユーザが CPU になるということは，ユーザがアバカスに思考上のプログラムを読み取らせ，操作を導くとも解釈できるため，OS のようなプログラムを作ることなく実行可能な状態を生じる．そして，従来のコンピュータと異なり，アバカスは思考上のプログラムの実在を問題とせず，ユーザを中心としたサイクルが成立することを問題としている．よって，アバカスは過去から未来への因果関係ではなく，思考上のプログラムの読み取りという現在だけを基準にしたサイクルをコンピューティングにもたらす発明となる．

3.4 実世界

アバカスによりユーザを CPU とするコンピューティングはプログラムが実在しないために，プログラムを継続的に改良するという発展性に欠けている．ここで珠への入出力という機能以外の新たな機能をアバカスに与えることを提案するならば，その実体を部品として組み合わせることで机や椅子のような構造物を作るという新たな発展性を生じる．すなわち，プログラムではなくアバカスの物理的な構造を開発の対象とした場合ではコンピューティングに限らず，実世界におけるアバカスによる置き換えが成立する．ROS(Robot Operating System)[3] や LRP(Live Robot Programming)[4] によって制御可能なロボットの部品をアバカスによって置き換えるならば，ユーザを CPU としない制御とユーザを CPU とする制御が両立する未知の発展性を持ったシステムを導く．これは実世界でのインタラクションにおける構成単位としてのアバカスを考察する意義をもたらす．

4. 思考実験

4.1 シミュレーション

人類は記号によって表現可能な法則を利用した計算をコンピュータ上で行い，実世界で未来に起こる現象のシミュレーションが可能である．シミュレーションはコンピュータを利用する目的の一つであり，天気予報のような実用的な情報をもたらしている．天気予報は利用者の判断に用いるべき情報であるが，風雨を避ける手段が存在しないならば，どれだけ正確であっても十分な価値を持たず，建物の中で活動する人間にとっては必ずしも必要とされない．よって，天気予報のようなシミュレーションの実用性は，風雨に強い建物と作るという実世界での現象との関係によって決定される．建物の代わりに利用できる大型のコンピュータに自己複製をもたらす技術が存在すると仮定するならば，天気予報より自己複製を優先することで，社会から天気予報の必要性が失われることは明らかである．すなわち，実体あるコンピュータの複製のサイクルは，記号表現に基づくシミュレーションでは解決できない問題の解決方法を導く確かな手段である．

4.2 自己複製

実現する具体的な方法は明らかとはなっていないが，生命に見られる自己複製を前提とした進化がコンピュータのような人工物であっても可能であることは証明されている [5]．コンピュータにおけるシミュレーションの進歩とは CPU の複雑化によって達成され，複雑な CPU は複製を困難なものとするため，自己複製の実現からは遠ざかる進歩となる．対して，アバカスは分散処理を前提とし，CPU が簡素で複製が容易であるために，実世界での自己複製を論

じる場合では有望な出発点となりえる．自己複製の実現を前述したロボットを構成するアバカスについて適用するならば，アバカスが CPU となるユーザに従い，部品として変形しながら組み合わさることでアバカスを再生産する建物を実体として建築するという事例が類推により導かれる．

4.3 考察と展望

上記は実世界でのインタラクションにおけるアバカスの構成単位としての有用性を論じるための思考実験であって，自己複製の実現方法について言及するものではない．DNA(Deoxyribonucleic Acid)[6] の発見によって生物の自己複製は説明され，その仕組みは複製する対象を選択できないという点と，利用可能な物質が限られるという点を制約としていることが知られている．人類の文明の発展も人工物の複製を基礎するが，上記のような制約は存在しない．人間の脳が複雑化を複製にもたらす原理は解明されていないが，自己複製を前提とする進化とは異なる基準で成立していることは明らかである．プログラムが脳の活動の産物であることに着眼するならば，ユーザを CPU として扱うことで思考上のプログラムを読み取り，実行するというアバカスの発展を論じることは重大な手がかりを導く可能性を持っている．この可能性を究明するために開発したソフトウェアを本稿の著者は https://github.com/metafiction/Protagonist にて公開している．

5. おわりに

実体あるアバカスを利用せずに，思考上の記号を利用して計算することは科学の原点である．記号の利用は法則の発見による進歩をもたらしたが，コンピューティングにおけるインタラクションにおいては記号が進歩をもたらすとは限らない．そのため，本稿は再びアバカスを利用して，ユーザが CPU となるコンピューティングが成立することを説明した．これは示唆に富むが，記号を用いないという仮定によって成り立つ人間とコンピュータの関係の発展を提案したものに過ぎず，アバカスの物理的な構造や従来のコンピュータとの関係は十分に論じられてはいないために，その検証が今後の課題となる．

謝辞　本稿の執筆を温かく見守って下さった皆様に感謝を申し上げます．

参考文献

[1] Neumann, J. V.: First draft of a report on the EDVAC, *IEEE Annals of the History of Computing*, Vol. 15, No. 4, pp. 27–75 (1993).
[2] Ifrah, G.: *A universal history of computing: from the abacus to the quantum computer*, John Wiley (2001).
[3] Quigley, M., Conley, K., Gerkey, B. P., Faust, J., Foote, T., Leibs, J., Wheeler, R. C. and Ng, A. Y.: ROS: an open-source Robot Operating System, *ICRA 2009* (2009).
[4] Campusano, M. and Fabry, J.: Live Robot Programming: The language, its implementation, and robot API independence, *Science of Computer Programming*, Vol. 133, pp. 1 – 19 (2017).
[5] Neumann, J. V.: *Theory of Self-Reproducing Automata*, University of Illinois Press (1966).
[6] Watson, J. D. and Crick, F. H.: Molecular structure of nucleic acids: A structure for deoxyribose nucleic acid, *Nature*, Vol. 171, pp. 737–738 (1953).