Vol. 30 No. 5　　情　報　処　理　　May 1989

解　説

TRONの思想と今後†

坂　村　健††

1. はじめに

コンピュータサイエンスの研究で，今後行われなければならないことは多々ある．その中でも，数千，数万という非常に多数のコンピュータを接続し，さまざまな相互関係をもちながら，個々のコンピュータがそれぞれの目的を同時並行的に遂行する，という協調型機能分散システムの研究は重要なテーマであると考えられる．

TRON (The Realtime Operating system Nucreus) プロジェクトは，この新しい考えに基づく協調型機能分散システム（超機能分散システム）構築のためのプロジェクトで，坂村により1984年に始められた[1]．

TRONには，分散環境におけるユーザのシステム操作方法を標準化するためのTRON作法の提供，システムもシリーズ化し階層構造をもたせ，さらにその仕様をオープンにすることにより，インプリメント独立でかつ自由なシステム接続を可能にするなど，従来のシステムにみられない数多くの特徴をもつ．

本稿では，TRONプロジェクトが目指している超機能分散システムについて述べることから始め，その実現のために考えられた，TRONアーキテクチャの特徴，ならびに将来への展望をもあわせて述べる．

2. 超機能分散システム

マイクロコンピュータの登場によりはじめて可能になった二つのコンピュータ応用分野がある．それは機器組込みの制御用コンピュータとしての応用と，個人が所有するコンピュータ，いわゆるパーソナルコンピュータとしての応用である．

2.1 インテリジェントオブジェクト

コンピュータがどんどん安く小さくなった結果，従来コンピュータを使っていなかった各種の機器にコンピュータを組込み，機能向上をねらった製品が数多く出てきている．たとえば，マイコン入りカメラ，マイコン入り電子レンジ，マイコン入り冷蔵庫，マイコン入り洗濯機，マイコン入り炊飯器などである．カメラでは焦点調節を自動化し，洗濯機や炊飯器ではでき上がり時間を指定すれば自動的に適当な時間に動作を開始する，といった機能の洗練を実現するためのコンピュータが使われている．

このように，安く小さくなったコンピュータやセンサやアクチュエータを組込み機能を洗練した「もの」をインテリジェントオブジェクトと呼ぶ．

2.2 コミュニケーションマシン

マイクロコンピュータのもう一つの代表的応用であるパーソナルコンピュータの将来について考えると，現在のような漠然とした形のパーソナルコンピュータではなく，はっきりとした使用目的を押さえた「コミュニケーションのための専用機」が重要となる．

コミュニケーションには「自分とのコミュニケーション」，「他人とのコミュニケーション」，「機械とのコミュニケーション」の三つのコミュニケーションがある．

自分の中にある何かモヤモヤしたものを，自分自身と対話しながらコンピュータの中に固定していく創造の過程を，自分とのコミュニケーションだと考える．この中に入る典型的なものがアイデアプロセッサやワードプロセッサであり，描画ソフトウェアである．

「他人とのコミュニケーション」というのは，単なる通信機能のことだけではない．異なる考えをもった人々の間でのコミュニケーションを助ける，さまざまな機能を指している．たとえばコンピュータグラフィックスやハイパーテキストなど，情報をいかに分かりやすく伝えるかといったプレゼンテーションのためのさまざまな機能を含む．また，異なる言語背景や身体条件をもつ人々の間の情報伝達を助ける自動翻訳やメディア変換の機能などが含まれる．

最後の「機械とのコミュニケーション」というの

† The Concepts Behind the TRON Project and Development by Ken SAKAMURA (Department of Information Science, Faculty of Science, University of Tokyo).

†† 東京大学理学部情報科学科

は，コンピュータのもつコントローラとしての機能のことである．インテリジェントオブジェクトが増え，それらがネットワークにつながれて協調動作する超機能分散システム環境では，インテリジェントオブジェクトからなる環境と人間との間の橋渡しをするコントローラとしての機能が重要なものとなる．

2.3 超機能分散システム

マイクロコンピュータの登場により，コンピュータは一部の専門家が特別な場所で使うものから，一般の生活の場面で使われるものに変わってきた．それにともない，環境中に存在するコンピュータの数は確実に増加してきている．この傾向が今後も続くならば，将来的には人間の環境を構成するすべての道具にコンピュータが入るという時代がくるだろう．

たとえば建物の制御を考えると，エレベータからエアコンまでの各種の機械，はては家具や壁にまでマイクロコンピュータとセンサが組み込まれてインテリジェントオブジェクト化する．各人は一人一台のコミュニケーションマシンをもち，情報に関係のある作業に気軽に利用する．工場でも工作機械は当然インテリジェントオブジェクトであり，マイクロコンピュータでは能力が不足な要求に備えて，要所には特殊目的のメインフレームが控え，必要に応じて高度なサービスを提供する（図-1）．

そのときに重要となるのが，インテリジェントオブジェクトに内蔵されたコンピュータを含む数多くのコンピュータを結び，協調して働かせる技術である．コンピュータ同士で助けが必要なら助けを，妥協が必要なら会話して妥協を，といった技術の開発が必要だ．

一つのコンピュータだけでは分からないような全体像を，分散されたコンピュータそれぞれの情報をもち寄ることにより，明確に把握できなければならない．局所的な目標追求を行う数多くのインテリジェントオブジェクトが独立に環境を与え，予期せぬ相互干渉を起こし，不安定な状態をもたらすことになりかねないからである．

逆に，数多くのコンピュータを協調して働かせることができれば，その協力の結果として，個々の独立した機能以上の機能を全体として実現できるようになる．このようにインテリジェントオブジェクトあるいはコンピュータが有機的に連携して働くことにより，人間にとって最適の環境を生みだすシステムを「超機能分散システム」と呼ぶ．

3. 超機能分散システムの特質

超機能分散システムは，現在の分散処理システムと異なった特質をもっている．超機能分散システムの実現にはこれらの特質を実現できるネットワークモデルおよびその管理機構の開発が必要である．

3.1 ネットワークの量的な巨大さ

超機能分散システムにおいては，システムの構成要素となるコンピュータの数が極度に多い．現在の分散

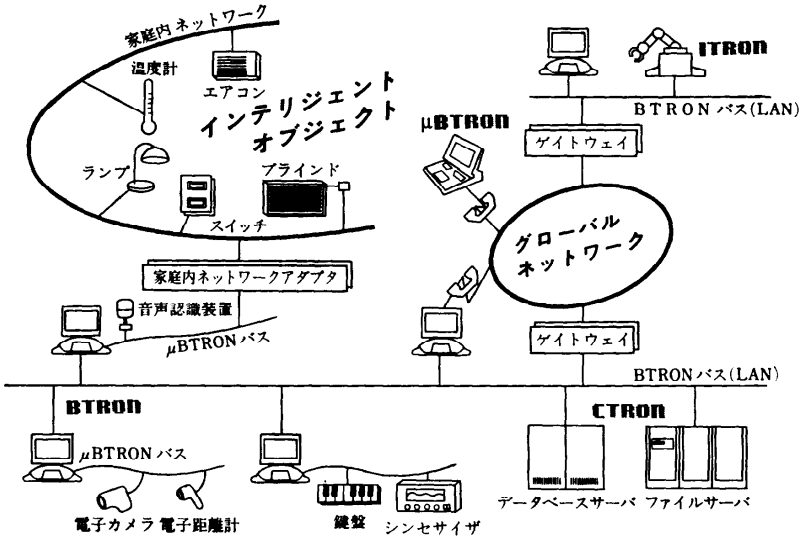

図-1 超機能分散システム

処理システムでは，定常的に相互関係をもつのは多くても数十台程度のコンピュータ（ワークステーション）からなるグループであるし，それらは作用を及ぼしあっているとはいえ，実際にはローカル処理が中心で，ネットワーク環境はあくまでも外の世界として考えられる場合がほとんどである．

これに対し超機能分散システムでは，定常的に相互関係をもつコンピュータの数は，住環境の制御で考えても，一部屋だけでも百を超えるということが普通で，建物全体では数千のオーダになるであろう．さらに建物レベルをサブシステムとして相互に結合されれば，都市ならば数百万，国ならば数億というオーダになる．

さらに地球的レベルでコンピュータ全部がつながるということまで考えると，将来的に必要なのは数千億，数百億という数のコンピュータが疎結合でつながった場合を考える必要がある．

3.2 構成要素の不均質さ

超機能分散システムにおいては，その構成要素がエアコンの制御コンピュータから巨大データベースサーバまでと多岐にわたる．このため必然的に，部屋にばらまいた温度センサとエアコンの間の関係から，それに指示する個人のコミュニケーションマシンとの関係，さらに空調機の要請で室内の流体力学モデルを作るスーパコンピュータとの関係など，その相互関係も多岐にわたる．

これら質の違う相互関係は，交信内容の複雑さやリアルタイム性など，多くの点で異なる要求をネットワークに対しもつ．超機能分散システムにおいては，これらが互いに重畳しながら存在することを考慮しなければならない．

3.3 物理環境に対する認識

超機能分散システムにおいて，その数量で考えるならば，構成要素の大部分はインテリジェントオブジェクトである．つまり，コンピュータの種別で考えるとコミュニケーションマシンや，サーバは相対的に少数で，大部分が機器組込みの制御用コンピュータである．

コンピュータが制御する機器は，同じ物理的環境に対する操作を行っている．特に物理的に近距離にある構成要素間では，ネットワークを介した通信と同様かそれ以上に，物理的空間での相互関係が重要である．現在の分散システムでは，位置に依存しないネットワーク透過性は重要であるが，超機能分散システムの要求としては適切なものではない．

超機能分散システムの環境は，物理空間と情報のネットワークが重ね合わされ，人間を便利に快適にするためにネットワークが物理空間に積極的な働きかけをするというイメージである．このためリアルタイム性と物理空間の積極的認識が不可欠になる．

4. 超機能分散システムに求められるもの

次に，現在の分散処理システムにおいても重要であるが，超機能分散システムという枠組で考えた場合，特に必要とされる性質について述べる．

4.1 安 全 性

超機能分散システムが，現在の電話と同様に，多くの会社組織や一般の家庭を含むネットワークを想定していることを考えると，企業秘密や個人のプライバシを守るために，非常に高い守秘性をもたねばならない．たとえば，個人の健康データは，健康管理を委託された医師には送るが，それ以外の経路には絶対に流れないようにしなければならない．

また，超機能分散システム上の多くの設備を自動運転することを考えると，システムは高い耐故障性と災害に対する対処能力をもたなければならない．たとえば，故障により暖房が勝手に動きだして火事になるといったことはあってはならないし，逆に停電や瞬断や断線などにより，家の設備が動かなくなり閉じ込められるなどといったこともあってはならない．

4.2 リアルタイム性

超機能分散システムにおいてはすべてのコンピュータは，外部の環境と密接な関係をもっている．たとえば，システム中の他のコンピュータ，ヒューマンインタラクション，制御する機器の状態，物理的環境の変化などである．

外部の環境と密接な関係をもっている場合，コンピュータ側の状況と無関係に進行する外部の変化に対応できるリアルタイム性は必須となる．

4.3 互 換 性

システム中の他のコンピュータとの交信を確立するために，すべての構成要素が共通のインタフェース体系をもっていなければならない．ネットワークに属する構成要素の数が非常に多いため，インタフェースがバラバラであると，交信するマシン相互の交信法を個別に用意することは現実的に不可能であるからだ．

さらに，ある構成要素を新機種に置き換えた場合を考えると，ソフトウェアのバージョン間の互換性も重

要である．これは超機能分散システムがオープンシステムで，系全体でみれば常時構成要素の更新が行われていると考えられるからである．もしバージョン間互換性が弱かった場合，複数の構成要素の協力によって機能している応用は非常に頻繁に不具合を起こすことになってしまう．

4.4 ユーザビリティ

人間も超機能分散システムの構成要素であり，その意味で人間と機械のインタフェースであるヒューマンインタフェースの問題を避けて通るわけにはいかない．

超機能分散システムの時代には，いろいろな所に入っているコンピュータと会話ができるということが，生活に基本的に必要な能力となるからである．その時代には子供から老人まで，障害のある人でも，騒音，振動などの劣悪な環境の中でも容易にコンピュータが使えるようなインタフェースが用意されなければならない．

4.5 データ標準

さらに，異なるアプリケーション間のデータのやり取りにおいて情報ができるかぎり失われず，しかも効率よく交換ができる，すなわち専門化と汎用性の両立が可能な高水準のデータ標準が必要である．さらに機械からのメッセージ伝達や，人間間の情報伝達の基礎である言語についても標準的なデータ形式が望まれる．その場合，超機能分散システムは全世界をネットワークすることを最終ゴールとするため，あらゆる言葉や記号表現をサポートできる，多国語処理データ標準を用意する必要がある．

5. 超機能分散システム実現のための標準化

上記の超機能分散システムに求められるものはまず第一に情報を伝える基盤としてのインタフェース体系の確立にかかっており，標準化というものと大きな関係をもっている．このため，TRON アーキテクチャの標準化に対するコンセプトとして「弱い標準化」，「トータルアーキテクチャ」，「オープンアーキテクチャ」，「シリーズ化」の4つのコンセプトが立てられた．

5.1 弱い標準化

標準化といった場合，単に互換性にかかわる部分すべてを標準化するということではなく，どの部分で互換性をとるかが大事である．

TRON では，アーキテクチャ構築にあたり，複数の階層構造をもたせ，各階層のインタフェースは規定しているが，アーキテクチャとインプリメント手法とを分離させて，ある階層から下のインプリメンテーションが変わっても，上位の階層はそのまま使用できるようにした．

つまり，インタフェースは限定するが，従来の規格のような特定のハードウェアや特定のプログラムを前提とした強い標準化は行わず，実際の実現にはさまざまな手法のバリエーションを許している．

そして，このようなコンセプトを TRON では「弱い標準化」と呼び，TRON が特定のコンピュータハードウェアやソフトウェアのことではないことを示す言葉としている．

弱い標準化では各種の互換性のために標準化の必要な部分，主にインタフェース部分を網羅する一つの設計基準を作り，不特定多数の人がその基準に従って別途実現を行えるようになっている．このように，インタフェースと実現を分離しているため，インタフェースさえ満足していれば，実現についてはインプリメンタの自由競争部分となる．

また，応用ごとに異なるインタフェース，たとえばアプリケーションソフトウェアのヒューマンインタフェースなどについては，設計ガイドラインといったレベルで既定し，応用に対応できる柔軟性を残している．

5.2 トータルアーキテクチャ

TRON ではリアルタイム性，標準化，高水準のユーティリティ，教育性の重視，設計基準と実現手法の分離といった要求を，組込み制御用のマイクロコンピュータからメインフレーム応用である大規模データベースまでというように，超機能分散システムに関係する広い応用分野全体にわたって満たそうと考えている．

またこのような応用分野に対する水平方向の広がりだけでなく，CPU の命令セットから始まりオペレーティングシステム（以下 OS）のシステムコール，高水準ユーティリティ群，文字コード，ヒューマンインタフェースなどの，コンピュータに関係するさまざまなインタフェース階層という垂直方向の広がりについても，一貫して設計しようとしている．

このようにトータルアーキテクチャを前提として統一的に再構築を行うため，複数のインタフェースを関連付けて更新することができる．これにより，多国語処理のように，従来一つのインタフェース階層のみの

機能拡張では理想的に対応できなかったような応用にも整合性をもって実現できる．事実，多国語処理を行うには文字コードだけ決めても何の解決にもならず，むしろ従来からの制御コードとの干渉のため多くのアプリケーションで不具合が起きるなど，現状の方法では OS レベルで使いものになる多国語処理の実現にはほど遠い状態である．

また，TRON ではトータルアーキテクチャとして計算機の各階層の再設計を同時に行うため，階層間にまたがるチューニングを行うことができる．たとえば OS でネックになる部分をプロセッサの命令で強化するなどといったことが可能になり，技術進歩を十分生かすことができる．また一貫した考えの下に設計されるので，効率がよく，しかも教育効果も高い．

TRON においてコンピュータの要素のほとんどを新規に作り直すというリスクの大きい方針で開発をすることとしたのは，次のような理由による．新しい応用に対応するために，個々のインタフェース階層で独立して対処するという方法を続けてきたため，規格が不統一となったと考えた．したがって本質的な対策は，すべてのインタフェースを統一的で発展的な考えのもとに新たに構築することである．

5.3 オープンアーキテクチャ

超機能分散システムにおいては構成要素間のインタフェースの互換性が維持されることが重要である．そうなってはじめて，その体系が広まり，本来の性能を出すことができるようになるからである．そのために TRON では先にあげた「弱い標準化」とともに，「オープンアーキテクチャ」という方針を取っている．

TRON で言うオープンアーキテクチャとは，標準化のために，TRON アーキテクチャのインタフェース仕様を TRON プロジェクトの主旨に賛同する全世界の人々に無償で公開するということである．

重要なことは，このようなアプローチにより，TRON の体系は一部企業に限られることなく広まっていくという利点をもつ．なぜならば，標準化されていることと同時に自由競争を維持しながら多くの企業が大きな負担なく参加できるようになっていることが，広まるために有利であるからである．そして，そのような公開された互換性の基盤の存在が，結果として市場を広げると考えるからである．

5.4 シリーズ化

現実にトータルアーキテクチャの実現を考えた場合，超機能分散システムに含まれる，機器制御から大規模データベースまでの広い応用範囲では，コンピュータに対するその要求には大きな違いがある．

たとえば，リアルタイム性の重視といっても，インテリジェントオブジェクト応用では数ミリ秒程度の応答時間が要求される．コミュニケーションマシン応用でも，ユーザビリティの重視という点から数秒以上待たせるべきではないが，たとえば，画面内の人間が注目している部分だけ更新するといった心理的な方法が使用でき，単に処理速度を向上する以外にも，いろいろな手法が取れる．

一方処理の規模からみても，インテリジェントオブジェクト応用の場合は，多くの処理を同時にこなしながら非常に高速の応答を要求されるが，処理一つ一つの規模は小さい．これに対しコミュニケーションマシン応用では，多くの場合一時に一つの処理しか要求されないが，その処理の規模が大きい．また処理の質も異なり，インテリジェントオブジェクト応用での処理の大部分が数値処理であるのに対し，コミュニケーションマシン応用では本来コンピュータが不得意とする文字処理や図形処理である．

このように要求の異なる応用について同じ OS で対処することは現実的ではない．

そこで，このような応用から要求される必要の違いを考慮し，TRON は，ITRON[2] というインテリジェントオブジェクト実現のための TRON，BTRON[3] というコミュニケーションマシン実現のための TRON，CTRON[4] といって ITRON と BTRON からなるネットワークの中核部分となる大型計算機やサーバなどに使う TRON，MTRON[5] といってネットワーク全体の調整を取る TRON の 4 つのシリーズに分け，TRON の全体コンセプトの下に独立したサブプロジェクトとして研究開発されている．

この分類は，個々の構成要素が直接「付き合うのはだれか」ということを基準にした分類である．つまり ITRON は機械と，BTRON は人間と付き合う．そして CTRON は周囲の環境と付き合わずに ITRON，BTRON と付き合う．そしてそれらすべてからなるシステムを包含するのが MTRON である．

また，これ以外に超機能分散システムの構成要素としての応用を特に重視して，システムの中核をなすパーツとして，まったく新しく独自に VLSI マイクロプロセッサを設計するサブプロジェクトを起こした．この成果が TRON チップ[6]である．

6. 超機能分散システムの実現

最後に，TRON における超機能分散システム実現のための開発の現状について述べる．TRON ではそのための要として，プログラマブル・インタフェースというコンセプトを導入することとした．

6.1 プログラマブル・インタフェース

超機能分散システムは数百，数万以上という大量の，しかも機器組込みコンピュータからデータベースサーバまでの多様なレベルのコンピュータを疎結合でつなぐネットワークである．しかも，このネットワークは開かれたネットワーク——常にどこかでノードが取り去られたり，追加されたり，取り換えられたりするオープンネットワークである．

このネットワーク内で個々の機器間のコントロールインタフェースを考えた場合，データベースからエアコンの制御にまで適用できるインタフェースというのは仕様が巨大すぎて現実的ではない．

また，弱い標準化というコンセプトによるものであっても，長期的に考えた場合に，インタフェースの仕様自体が技術進歩や新しく生まれる応用に対して最適なものでなくなることは否定できない．そのような場合には，当然 TRON においても仕様のバージョンアップが行われるべきである．

従来のシステムでは，このようなバージョンアップが行われた場合，アプリケーションが使用できなくなるなど，インタフェースの不整合による不具合が起こることは黙認されてきた．しかし，超機能分散システムにおいて考えた場合，その不整合は予想できない広さで影響をおよぼす可能性があるし，OS がバージョンアップされたために，ビルや家庭での多くの機器が動かなくなるとしたら，このような不具合は容認できるものではない．

つまり，超機能分散システム実現のためには，「インタフェースの互換性を取りながら，応用に適した拡張が可能で，技術の進歩に応じた成長が可能なインタフェース」の実現が TRON に求められているのである．また，このような性質はヒューマンインタフェースをはじめとする，TRON に存在する多くのインタフェース仕様において有用な性質である．

このようなインタフェースを実現するのが，動的にアルゴリズム記述をともなう仕様の拡張もしくは変更ができるプログラマブル・インタフェースである．各種のインタフェースを，プログラマブル・インタフェース化することにより，インタフェースの仕様自体を実行時に柔軟に変更し，インタフェースを介して接する二つのシステム間での最適な負荷分散が行える．さらに，将来的な技術進歩に対する適応とインタフェースの互換性を両立することが可能になる．また，デザインガイドラインとインプリメンテーションの分離による多様性と，互換性を両立することがシステムレベルで可能になる．

これにより，最終的な MTRON 環境である超機能分散システム構築が可能になる．

6.2 TULS

このプログラマブル・インタフェースはプログラミング言語によって実現される．このためのプログラミング言語の基本仕様となるのが TULS (TRON Universal Language System)[7] である．

TRON では最終的には，意味記述，データ記述，ヒューマンインタフェース，ネットワークプロトコル，ユーザプログラミング，アプリケーションプログラミング，システムプログラミングなどを TULS で記述する．

TULS プログラムの評価はマクロの展開という形で理解することができる．記憶領域としては，ローカルな揮発性記憶である辞書と，グローバルな永続性記憶である実身をもち，辞書の中でマクロの定義が行われている．

ここでいう実身とは従来のシステムにおけるファイルに当たるものであり，BTRON における永続性記憶の蓄積および検索のためのモデルである実身/仮身モデルにおける用語である*．

マクロ呼び出しは特別なコードを含む一連の文字列からなるが，それを置換するマクロ本体は任意のデータが許され，変数も関数もマクロの展開として実現できる．マクロ本体が評価され，その中のマクロ呼び出しがすべて置換されてから，全体が最初のマクロ呼び出しと置換される．

マクロ呼び出しとマクロ本体の組を集めたものが，辞書であり，呼び出しが発生した場合，辞書を検索し対応するマクロ本体を評価し結果を返す．評価の連続がプログラムの実行に当たる．この評価の流れは，同

* BTRON においては，永続性のあるデータのまとまりを実身と呼び，それを仮身という特別なタグにより参照できるようになっている．仮身はデータとして文字や図形と混在できるため，同じ実身をデータとしても，ファイルのディレクトリとしても使用できる．また一つの実身を複数の仮身で参照できるため，実身の参照関係は，従来のファイルシステムのようなツリー状でなくネットワーク状になる．詳しくは文献 3) 参照．

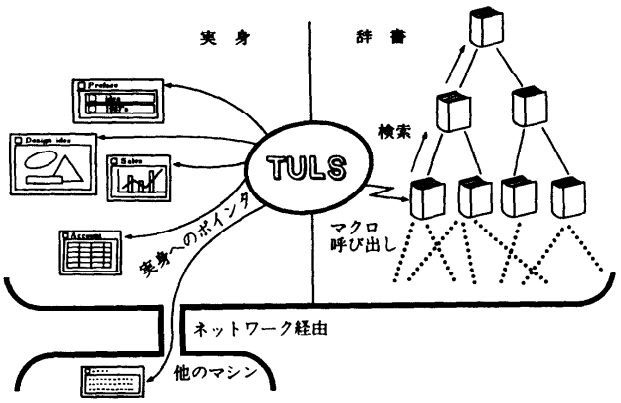

図-2 TULS における実身と辞書

時に複数存在しうる.

辞書は動的にいくつでも作ることができ，この生成関係により親子関係をもつ．マクロの評価時はある辞書中で定義を捜してなかった場合，その親の辞書へさかのぼるというようにして，最終的にはルートの辞書までさかのぼるというサーチルールを取る．この辞書間の関係により，実質上のインヘリタンス（継承性）が実現できる(図-2).

TULS はインタプリタを主とし動的に必要機能を部分コンパイルできる対話型言語[9]である．関数単位でコンパイル可能であるため，リアルタイム要求の高いインタフェースでは，効率を維持するために主要機能をネイティブコードで実現できる．コンパイルした関数は，ルートの辞書で実現された TULS 本来のプリミティブマクロと同様に使用できる.

TULS の特徴は以下のようにまとめることができる.

—対話型言語（実行時部分コンパイル可能なインタプリタ言語）である
—マルチコントロールフローをもつ
—型をもたずマクロにより何でも抽象化できる
—データとプログラムに区別がない
—実身と辞書の二系統の記憶を使用する

TRON の各インタフェース仕様は，TULS に仕様やハードウェアに合わせた関数ライブラリを追加することにより実現される．このように応用に特化して仕様拡張された TULS も当然それ自体対話型言語としての性質をもち，プログラマブル・インタフェースである.

TULS で記述していないプログラムも，ネイティブコードの形で，TULS にプリミティブマクロとして取り込むことができる．現在，TULS を使わずに作られている各 TRON OS のコールも，将来的には TULS のライブラリと考えることができる．この場合，将来的な技術進歩により，TULS で記述しても効率上の問題がないようになれば，現在のコールを TULS で実現することでエミュレーション可能であり，バージョンアップ時にも仕様的に無理なく継承される.

このようなことが可能なのは，TRON でプログラマブル・インタフェースを採用し，互換性の最低保証とするだけでなく，そのプログラマブル・インタフェースで実現する個々の具体的インタフェース仕様も決めているからである．そのため，現在の TRON の固定的インタフェース仕様も将来のプログラマブル・インタフェースと矛盾しないように考慮されて設計されている.

6.3 TACL

TACL (TRON Application Control-flow Language) はヒューマンインタフェース応用に特化して仕様拡張された TULS である．TACL が人間と BTRON のインタフェースをつかさどる言語であるということは，BTRON とユーザとのすべての対話に TACL インタプリタが介在するということである.

ユーザからの入力をイベントとして実行中のアプリケーションに伝えることから，BTRON 上のアプリケーションが処理結果をユーザに示す行為までの一連の処理の流れは，すべてが TACL の実行の結果として捉えることができる（アプリケーションは巨大な TACL プリミティブとして呼び出すと考える）.

アプリケーションの実行も TACL から行われるということにより，適当な TACL プログラムを書くことで，一連のアプリケーションの自動実行を行わせることもできる.

TULS の特徴に加え，TACL の特徴は以下のようにまとめることができる.

—一般のデータ中のどこでも TACL のマクロを挿入することができる
—ユーザのイベントも，内部イベントもマクロとして扱う

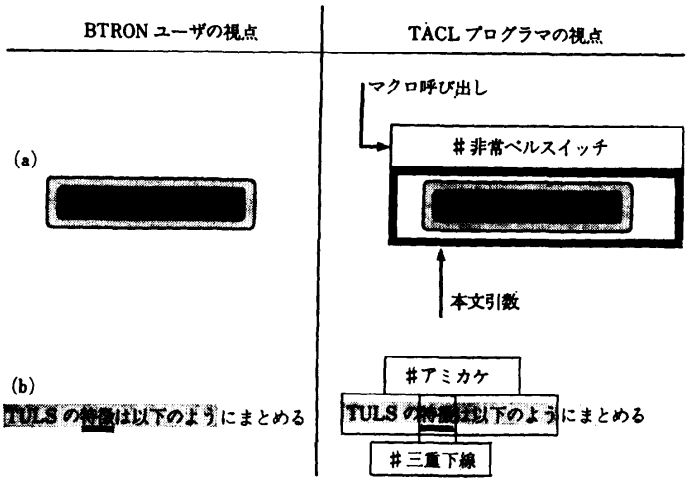

図-3 本文引数をもつマクロ

―マクロへの引数として本文データの一部分を扱うことができる.

本文データの一部分をマクロへの引数としてローカルな辞書を作ることにより，その部分でのポインティングデバイスの操作により発生するイベントを，本来のアプリケーションに渡す前に TACL で評価し，異なる反応を起こさせることが可能になる．たとえば図形エディタで表示されている図形の一つでポインティングデバイスをクリック（押し離し）することにより，音楽がなるといった反応をさせることも可能になる（図-3(a)）.

また，ウィンドウの再表示イベントを，本文データの一部分でだけ特別に評価することにより，3重下線など本来の仕様にない表示を行わせることができる（図-3(b)）.

BTRON では基本的な機能ライブラリが用意されており，この環境でユーザプログラミングを行える．ライブラリとしては以下のような分野のものが提供される.

―基本操作
―計算
―変換
―制御
―入出力
ビジュアルエフェクト/サウンドエフェクト/ウィンドウ制御/ダイアログパネルデザイン
―作用
―特殊参照/設定
―リスト処理
―その他

これ以上に，マクロはそれぞれが独立した機能のインタフェースを表現しているので，アプリケーションをインストールして，新しい機能をもったマクロを追加することで簡単に機能の追加ができる.

TACL によるプログラミングの特徴は，通常の意味でのメインプログラムというものが存在しないことである．TACL の評価は常に，動作中の BTRON 環境の中で行われる．そのため，一連の評価の連鎖を始めるためのイベントは，なんらかの形で外部から与えられると考えてよい.

ユーザが指示して TACL プログラムを含む実身の評価を行わせることにより，プログラムが起動される．しかし一般の対話型プログラムでは，辞書のセッティングがおわるとそのプログラム自体は終了する．セットされた辞書は残り，ユーザイベントを待つ状態で待機している.

この辞書はユーザイベントによるマクロ呼び出しによって動作するいくつかの小プログラムからなっており，セッティング後にユーザがなんらかのイベントを発することにより，対応するプログラムが働く.

このように，TACL は，画面上の表示領域に密着したイベント評価機能をもつイベントドライブ言語であるということができる．ヒューマンインタフェースの基本は人間の動作に対して反応し，それがさらに人間の動作を促すという，このイベントドライブメカニズムであり，TACL ではヒューマンインタフェースの記述が特にやりやすくなっている.

6.4 TAD

TAD (TRON Application Data-bus) は TRON におけるデータ互換性の基盤となる規約である．TAD は TRON で扱う文字や図形などの基本的な要素データ（これを TRON ではセグメントと呼ぶ）の表現仕様を規定し，さらにそれらを組合せる方法を定めている．基本的なデータを組合せることにより，さまざまなマルチメディアデータが実現できる.

TAD の基本的な要素データとしては，以下のデー

タがある.

0. 音声セグメント
1. 文章セグメント（1次元データ）
2. 図形セグメント（2次元データ）
3. 物体セグメント（3次元データ）

音声セグメントは，一単位の音を規定する．音の高さ，音色，アタックタイム，エンベロップなどのパラメータで表現された音データ，発音記号，声の高さ，声色，などのパラメータで表現された声データからなる．このような抽象化された表現とならび，PCM 録音された生の音データも使用できる．

文章セグメントは一連の文字コードからなるデータである．先頭の言語指定により，以下の文字コードをどの言語として評価するかが決まる．また一つの言語でも，複数の表現形式をもつ場合も考慮にいれられ，表現の指定ができる．たとえば，漢字仮名交じり文による日本語とローマ字による日本語は表現は異なるが同じ日本語であり，ローマ字がアルファベットを使っていても英語とは違うと認識される．

図形セグメントは長方形，円，折れ線，曲線…といった一つの図形を表現する．図形を，座標位置・属性，図形を塗り潰すための模様のパターンなどのパラメータで表現した抽象化された図のデータと，色の点の格子状の集まりで規則性のない絵を表現するピクセルマップデータがある．

物体セグメントは，一つの3次元物体を表すもので，物体の形状，材質，質量などのパラメータをもつソリッドモデル，複雑な表面形状を表すワイヤフレームモデル，点の立体格子状の集まりで規則性のない物体形状を表現するボクセルマップデータなどいくつかの表現形式を定める．

TAD の特徴は，これらのデータを自由に混在させることができるということである．同じ種類のセグメントを集めたものは和音，文章，図形…である．また異なるデータを組合せれば，たとえば，文章データの中に図形データを入れれば，文章の自由な位置に絵を埋め込むことができる．文章・図形データの中に音データを挿入すれば，表示時に音を鳴らすことができる．

TAD では，このほかに，音声，文章，図形，物体のセグメントを時間軸上に並べるための実時間制御セグメントをもつ．実時間制御セグメントは単独では使用されず，常に他の一連のデータを時間軸上で配置するのに使われる．

たとえば，音データと実時間制御セグメントを一緒に用いると，音楽演奏データを記述することができる．また，図形データを実時間制御セグメントとともに用いると，動画を記述することができる．

実時間制御セグメントの中には，各種のデータを並列に再生するための構造をもっている．たとえば，図形データと文章データと音データの3種のデータを同期させながら再生するように記述すると，動画を描画しながら，その解説をテロップで挿入し，台詞を読み上げるといった状況を再生することができる．

各種のデータ間ではデータ変換が可能な場合がある．たとえば，文章を朗読してその結果を音データに変換することを考える．この場合，音データへの変換には，文章データの他に声質，抑揚，強弱などの多くの付加データが必要となる．このように，データ間の変換は，付加情報の削除や追加の操作が介在する．TAD ではこのような付加情報は付箋として元のデータの中に挿入され，データの可換性を高めている．

一つ一つの TAD セグメントは，なんらかの表現を産む TULS のマクロとして考えることができる．これにより，TAD もアプリケーション間をつなぐプログラマブル・インタフェースとして機能する．

したがって，TAD の仕様自体も実行時に柔軟に変更できるようになる．また，バージョンが古いとか，アプリケーション間でデータのやり取りが想定されていない場合でも，相手の環境に，必要な辞書環境を渡すことにより，互換性が保証される．このようなインタフェース間でのバージョン調整は自動的に行われる．

7. おわりに

現在のコンピュータ体系のほとんどすべてを見直すということは，大きなリスクをともなうという声も聞かれる．新しいコンピュータ構築には多大のコスト，日数がかかり，過去の資産を生かすことをまず考えるべきだという理由からである．

しかし，現在のコンピュータ体系を維持しようとすることが逆に，新しいシステム開発に際して多くの問題を生み，個々のレベルでの局所的対応が新システム構築を邪魔する結果となっている．もはや部分的改良による発展法は限界に近づいており，このままではコンピュータの発達の次の段階である協調型分散システム構築は困難である．

いま努力を集中して，これまでの成果をもとに，新

しいマイクロエレクトロニクス技術を前提に新しいコンピュータ体系を一貫して構築すること．それが，結局は超機能分散システムを構築するための近道であると考える．

TRON では差し当たりの目標は 90 年台に個々の実用システムを広めることであるが，90 年台の終わりから 21 世紀にかけて，それら個々のシステムを統合し有機的に協調動作できるようにしていき，最終的には超機能分散システムを構築することを目標としている．

さらに，本稿では触れなかったが，そのために，コンピュータそのものだけでなく，それを使った電気製品から住宅，ビルまで含めた具体的なインテリジェントオブジェクトの設計も行っている．コンピュータの能力を，老齢者や障害者を含む，どんな人でも使えるようにするには…，今まで考えていなかったような場面でコンピュータの能力を使えるようにするには…，独立にシステムを開発しても最終的にドッキングできるようにするには…．そのときにはどういう枠組が必要か——そういったことを確立しようとしているのが TRON である．この枠組というのがまさに概念であり，もっとも根本的な意味でのソフトウェアであると考えている．

参 考 文 献

1) Sakamura, K.: The Objectives of TRON Project, TRON PROJECT 1987, Springer-Verlag, pp. 3-16 (1987).
2) Sakamura, K.: ITRON: An Overview, TRON PROJECT 1987, Springer-Verlag, pp. 29-34 (1987).
3) Sakamura, K.: BTRON: An Overview, TRON PROJECT 1987, Springer-Verlag, pp. 75-82 (1987).
4) Sakamura, K.: CTRON: An Overview, TRON PROJECT 1987, Springer-Verlag, pp. 153-156 (1987).
5) Sakamura, K.: Design of MTRON: Construction of the HFDS, TRON PROJECT 1988, Springer-Verlag, pp. 21-32 (1988).
6) Sakamura, K.: TRON VLSI CPU: Concepts and Architecture, TRON PROJECT 1987, Springer-Verlag, pp. 199-238 (1987).
7) Sakamura, K.: TULS: TRON Universal Language System, TRON PROJECT 1988, Springer-Verlag pp. 3-19 (1988).
8) Atkinson, M. P. and Buneman, O. P.: Types and Persistence in Database Programming Languages, ACM Computing Surveys, 19, 2.

(平成元年 3 月 15 日受付)