知識工学と人工知能　49－5
（1986. 11. 11）

ＡＳ３０００用Ｋｙｏｔｏ　Ｃｏｍｍｏｎ　Ｌｉｓｐ

山本孝志　　星野洋　　才所敏明
（株）東芝　総合情報システム部

京都大学が開発したＫｙｏｔｏ　Ｃｏｍｍｏｎ　Ｌｉｓｐ（ＫＣＬ）を当社のＥＷＳ（ＡＳ３０００シリーズ）へ移植し，ＡＩ応用システム開発用言語としての性能と機能を評価して，機能拡張及びベンチマークを行ったので報告する。

Ｃｏｍｍｏｎ　ＬｉｓｐはＬＩＳＰ言語特有の高い記号処理能力と豊富な関数が用意されて記述力に優れ，一方，ＥＷＳはマン・マシン・インタフェースを重視し高度のユーザ・インタフェース開発環境を備えている。本報告は，ＫＣＬとＥＷＳとのインタフェース・プログラムを開発して実現したグラフィクスやウインド機能，及び，情報処理学会記号処理研究会主催のＬＩＳＰコンテストで使用されたベンチマーク・プログラムによる性能評価結果について述べる。

Ｋｙｏｔｏ　Ｃｏｍｍｏｎ　Ｌｉｓｐ　ｆｏｒ　ＡＳ３０００

Takashi YAMAMOTO , Hiroshi HOSHINO , Toshiaki SAISHO
Total Information & Systems Division,TOSHIBA Corporation
72,Horikawa-cho,Saiwai-ku,Kawasaki,210,Japan

This paper describes a extended function and performance of KCL/AS3000. We port Kyoto Common Lisp (KCL) , developed by Kyoto Univ., to AS3000 TOSHIBA engineering workstation. With a graphics library and window management library, KCL/AS3000 has a facility to develop an AI applied system. By WGSYM benchmark test, we also find KCL/AS3000 has a well performance for application.

A library inclusion method and a benchmark result are discussed in this paper.

１．はじめに

Ｋｙｏｔｏ　Ｃｏｍｍｏｎ　Ｌｉｓｐ（ＫＣＬ）は，京都大学・数理解析研究所の湯浅太一氏，萩谷昌己氏によって開発されたＬＩＳＰ処理系であり，ＥＷＳや中型機などのＵＮＩＸマシン上で動作する。当社では，社内ＥＷＳ（ＡＳ３０００シリーズ）上の標準ＬＩＳＰ言語の候補としてＫＣＬを導入して，ＡＩ応用システム記述言語としての適用性を検討した。

ＫＣＬの主な特徴は次の通りである。

（１）　ＬＩＳＰの国際標準案Ｃｏｍｍｏｎ　Ｌｉｓｐ準拠

（２）　Ｃ言語とＫＣＬ自身で処理系が記述され移植性に優れる

（３）　Ｃソース・プログラムの組み込み，ＵＮＩＸコマンドの呼び出し可

また，ＡＩ応用システム記述言語の要件として次のものがある。

（１）　高い生産性

（２）　外部プログラムとの接続

（３）　高い実行効率

ＡＩ応用システム開発ではプロトタイピングにみられるように短期間で柔軟で拡張性あるシステムを作成する。このため記述言語に高い生産性が要求される。ＬＩＳＰ言語には高度な記号処理能力が備わっており，Ｃｏｍｍｏｎ　Ｌｉｓｐに豊富な機能が取り入れられ，ＫＣＬは高い生産性を発揮する。

（２）については，既存のデータ処理システムにＡＩを取り組む際に必要となる。ＫＣＬにはＣプログラムを組み込む機能が備わっており，Ｃプログラムを介して，外部プログラムを呼び出すことができる。しかし，計算機システムが提供するさまざまなライブラリと接続する際も，毎回ユーザがＣでインタフェース・プログラムを記述していたのでは，ユーザの負担が大きい。これに関する機能拡張について，第二章で述べる。

（３）については，会話型プログラムでは重要な要因となる。ＫＣＬの性能を評価した結果については，第三章で述べる。

本報告では，ＫＣＬの機能と性能を評価し，機能拡張およびベンチマークを行った結果について報告する。

２．機能拡張

２．１　機能拡張の概要

Ｃｏｍｍｏｎ　Ｌｉｓｐは，米国のＧ．Ｓｔｅｅｌｅが中心となりＬＩＳＰの国際標準化をめざして設計された。１９８４年に発表された言語仕様は，

①　機種依存性のない共通の仕様　②　多くの機種で実行するための移植性

③　コンパイラとインタプリタの同一性　などの基本方針で設計され，豊富な機能が盛り込まれた。

一方，ＥＷＳはマン・マシン・インタフェースを重視して設計され，ハードウエアとしてビットマップ・ディスプレイやマウス，ソフトウエアとしては，ウインド管理システムやグラフィクスを備えている。Ｃｏｍｍｏｎ　Ｌｉｓｐでは，グラフィクスなどの機能は複数の案が提案されたことから見送られたが，ＥＷＳ環境で応用システムを開発する際は次の機能が重要であると判断し，ＫＣＬに拡張機能として取り入れることにした。

（１）　マルチウインド利用環境
（２）　グラフィクス利用環境
（３）　日本語機能

２．２　実現方式

ＡＳ３０００のＯＳ／ＡＳ上では，①　マルチウインド管理システム　ＳｕｎＶｉｅｗ　②　ＳＩＧＧＲＡＰＨのＣＯＲＥ仕様のグラフィクス　ＳｕｎＣｏｒｅ　③　ＡＴ＆Ｔパシフィク社のＵｎｉｘ拡張コードを採用した日本語環境　が提供されている。

これらのシステム・ライブラリ（ＳｕｎＶｉｅｗ，ＳｕｎＣｏｒｅ）とＫＣＬとのインタフェース・プログラムを，ＫＣＬが持つＣプログラム組み込み機能を利用して開発し，ＫＣＬ上にユーザ・インタフェース開発環境を実現した。インタフェース・プログラムの実現方法としては，次の方法が考えられる。

（１）　システム・ライブラリとインタフェース・プログラムは，ＫＣＬと別プロセスにおく
（２）　システム・ライブラリとインタフェース・プログラムは，ＫＣＬの作業領域におく
（３）　システム・ライブラリはＫＣＬの処理系の本体部におき，インタフェース・プログラムは，ＫＣＬの作業領域におく。

（１）の方式は，ＫＣＬとは別プロセスでウインドやグラフィクスを操作する方式である。ＫＣＬ側には，プロセス間通信モジュールをおき，パイプを経由して，外部プロセスを操作する。プロセス間通信のためには，データの受け渡しやライブラリの呼び出しのためのインタフェースを設計する必要がある。

（２）の方式は，ＫＣＬの本体の外部の作業領域にライブラリとインタフェース・プログラムをおいて，処理系とライブラリとの独立性を保つ方式である。この方式では，処理系の作業領域がふくらみ，ガーベジ・コレクションの時間がかかる。

（３）の方式は，ＫＣＬの本体とシステム・ライブラリを合体して新しいＫＣＬの処理系を作成しインタフェース・プログラムは作業領域におく方式である。システム・ライブラリがＫＣＬの作業領域から省かれたことによって，作業領域の有効利用およびガーベジ・コレクションによる実行時間の短縮をはかることができる。

記憶域の大きさは，（１）～（３）のいずれの方式も格差がないと思われるが実行効率の面では，（１）の場合はプロセス間通信とプロセス間のデータの引渡しに時間がかかり，（２）の場合はＫＣＬの作業領域内にシステム。ライブラリが格納されることによってガーベジ・コレクションの時間がかかる。

そこで，実行効率を考慮して（３）の方式で実現することにした。

次に，インタフェース・プログラムには，次の機能が必要となった。

（１）　データ構造の変換
（２）　割り込み処理
（３）　メモリ管理

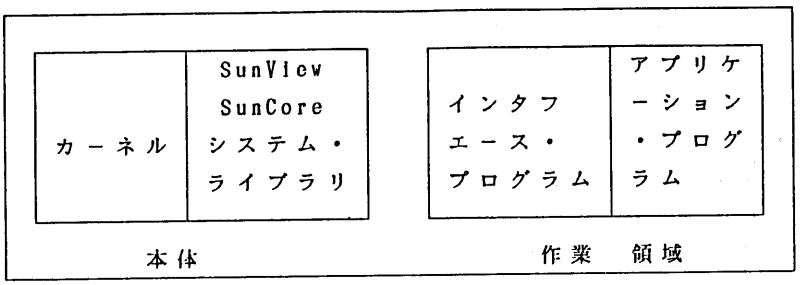


図１　　拡張ＫＣＬの構造

（１）のデータ構造の変換は，ＬＩＳＰのデータ構造をＣのデータ構造に変換することを意味する。システム・ライブラリはＣ言語とのインタフェースを持ち，配列や構造体を関数の引数として使用している。そこで，ＫＣＬからライブラリを自然な形で呼び出せるように配列や構造体に対応するストラクチヤを設計して，ＬＩＳＰのストラクチヤをＣの配列や構造対へ変換するモジュールを作成した。特に，ＳｕｎＶｉｅｗでは，フレームやキャンヴス，ｔｔｙなどのウインドとウインド属性の為に，データ抽象型の構造体を用いているため，ＫＣＬ上でオブジェクト指向風のストラクチヤを作成した。これによって，ＫＣＬからウインドの作成・移動・削除を行うためのプログラミング環境を整えた。

（２）の割り込み処理は，ＫＣＬ処理系が提供する割り込み処理とＳｕｎＶｉｅｗが用意しているノーテファイア機構の両立を意味する。ノーテファイア機構では，利用者が割り込み処理を行う際に，割り込みルーチンをノーテファイア機構に登録することを，義務ずけている。そこで，ＫＣＬ本体が行っている割り込み処理を調査してノーテファイア機構に登録した。

（３）のメモリ管理は，システム・ライブラリの管理下のメモリを，ＫＣＬ処理系管理下に置くことを意味する。システム・ライブラリでは，実行時に動的にメモリを獲得したり解放したりする。一方，ＬＩＳＰ処理系では，使用するメモリは作業領域において，処理系でメモリを管理しなければならない。そこで，メモリを確保するルーチンを置き換えて，ＫＣＬ下でメモリ管理することにした。

２．３　日本語の扱い

当社の日本語ＵＮＩＸ（ＯＳ／ＡＳ）は，ＡＴ＆Ｔパシフィク社が推奨するコード体系を採用している。ＡＴ＆ＴのＵｎｉｘ拡張コードは，ＪＩＳ漢字や半角片仮名を２バイトで表現するが，２バイト目は常に先頭のビットがＯＮとなる。これによって，他のコード体系の様に２バイト目に括弧やシングル・クオートといったＬＩＳＰで特別の意味をもつコードがなくなり，処理系に手を加えることなく日本語の取り扱いが可能となる。

今回，ＫＣＬ処理系では特別の日本語処理を行わずに，ＡＳ３０００が提供する日本語環境（ＪＩＳキーボード，日本語化ツール，仮名漢変換日本語入力）で，日本語入出力できることを確認して，ＬＩＳＰの文字型として日本語を扱うことにした。

２．４　実現機能

ＡＳ３０００は，１１５２×９００画素の解像度を持った１９インチ型ディスプレイを備えモノクローム及びカラーの表示が可能である。ＯＳ／ＡＳには，ウインド管理及びインタフェース開発のためのシステムＳｕｎＶｉｅｗ，Ｃｏｒｅ仕様のグラフィクスＳｕｎＣｏｒｅが組み込まれている。

今回の拡張でＫＣＬに実現した機能は次のとおりである。

（１）ウインドウ利用環境

・ウインドウ・タイプ

フレーム（ウインドウの外枠）

キャンバス（グラフィクス表示用ウインドウ）

ｔｔｙ（端末エミュレータ）

・ウインドウ付属オブジェクト

メニュー，スクロール・バー，カーソル，アイコン

＊上記３種類のウインドウの生成・削除・移動と４種類のウインドウ付属オブジェクトの付加が可能

（２）グラフィクス利用環境

・２次元・３次元の座標系

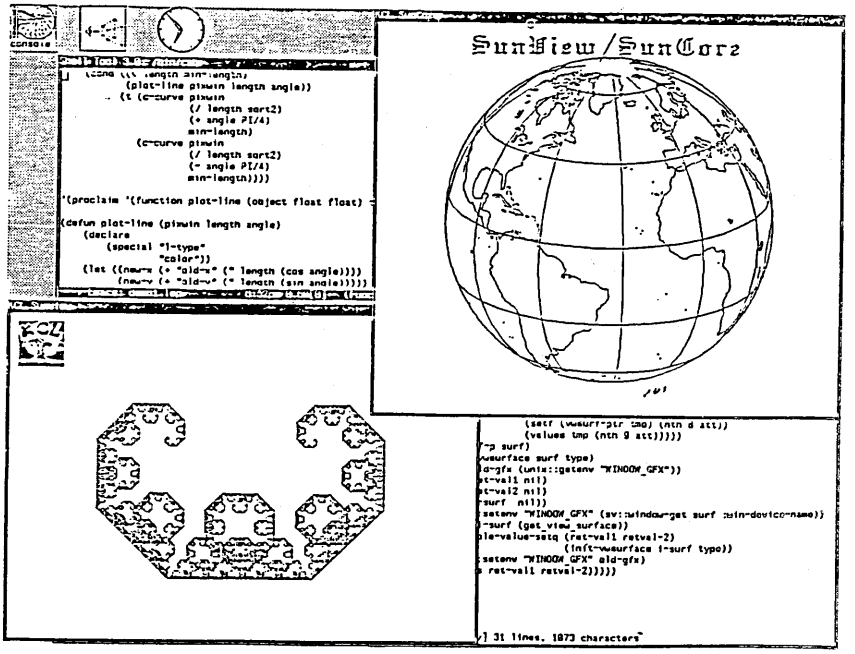
・セグメントによる図形表現

・ベクトル，マーカ，テキスト，ラスタ基本出力要素

・基本出力要素に対するカラーや模様などの属性付与

また，ＫＣＬからシステム・ライブラリを呼び出す際，システム・ライブラリと同名の関数が使用でき，引数としてストラクチャが提供され，利用者の便をはかっている。

２．５　実行例



3．ベンチマーク結果

3．1　ベンチマーク方法

情報処理学会記号処理研究会主催の第三回ＬＩＳＰコンテストで使用されたベンチマーク・プログラムを用いて性能を評価した。

使用マシンはＡＳ３０００，実記憶４Ｍバイト，浮動少数点アクセラレータなしの環境で評価した。

3．2　ＫＣＬ／ＡＳ３０００のベンチマーク結果

ベンチマーク	インタプリタ (ms)	コンパイラ (ms)	ベンチマーク	インタプリタ (ms)	コンパイラ (ms)
[1-1] Tarai-4	24016	81.7	[8-1]Sort-50	72.67	71.5
[1-2] Tarai-5	649850	2183	[8-2]Sort-10	5.84	5.83
[1-3] Tarai-6	———	78883	[8-3]Sort-100	90	883
[1-4] Tak-18-12-6	123680	450	[8-4]Sort-1000	1350	1300
[2-1] List-Tarai-4	39617	2150	[8-5]Sort-10000	26900	25933
[2-2] Srev-5	1483.3	16.7	[8-6]Rsort-10	9	8.33
[2-3] Srev-6	5267	67	[8-7]Rsort-100	143.3	140
[2-4] Qsort-50	1088.3	26.7	[8-8]Rsort-1000	1900	1867
[2-5] Nrev-30	818.3	16.7	[8-9]Rsort-10000	32133	32233
[2-6] Rev-30	0.33	0.67	[9-1]TPU-1	11017	1867
[2-7] Drev-30	0.67	0.266	[9-2]TPU-2	48983	6597
[2-8] Rev-10	0.67	0.267	[9-3]TPU-3	19100	2150
[2-9] Rev-100	11.84	2.67	[9-4]TPU-4	24367	4183
[2-10]Rev-1000	25	26.7	[9-5]TPU-5	3117	350
[2-11]Rev-10000	1267	267	[9-6]TPU-6	105783	11600
[2-12]Drev-10	0.33	0.083	[9-7]TPU-7	20200	1917
[2-13]Drev-100	0.85	0.75	[9-8]TPU-8	20067	1383
[2-14]Drev-1000	7.34	7.24	[9-9]TPU-9	13400	984
[2-15]Drev-10000	73.67	73.83	[10-1]Pri-Rev-15	———	983
[3-1]String-Tarai-4	34850	11450	[10-2]Pri-Sort-20	———	1417
[4-1]Flonum-Tarai-4	26400	2017	[11-1]Diff-3	293.3	8.3
[4-2]Binnum-Tarai-4	26183	2683	[11-2]Diff-5	21500	600
[5-1]Bubble-50	3281.7	411.7	[12-1]Boyer	1409867	42850
[6-1]Seq-100	500	150			
[7-1]BITA-5	180	10			
[7-2]BITA-6	735	31.7			
[7-3]BITB-5	61.6	3.5			
[7-4]BITB-6	168.3	11.7			

3．3　ＫＣＬ／ＡＳ３０００と他処理系の実行時間の比較

処理系	インタプリタ			コンパイラ		
項目	Utilisp OS/MVS	VAXLISP VAX/VMS	KCL/SUN2 OS/UNIX	Utilisp OS/MVS	VAXLISP VAX/VMS	KCL/SUN2 OS/UNIX
[1] 関数呼出の手間	0.011	0.79	2.77	0.21	3.96	3.80
[2] リスト処理	0.01	2.28	2.15	0.05	1.54	6.80
[3] 文字列の処理	0.01	3.15	3.47	0.02	0.16	3.37
[4] 浮動少数点計算	0.01	2.7	2.9	0.02	0.28	1.04
[5] 配列処理	0.02	1.31	2.54	0.01	0.62	2.40
[6] 属性リスト処理	0.02	6.82	3.17	0.04	0.90	3.11
[7] マップ関数	0.01	3.45	2.88	0.04	2.07	3.22
[8] ソート	0.02	0.82	2.74	0.02	0.67	2.27
[9] 応用ＴＰＵ	0.01	2.65	2.44	0.04	1.23	3.11
[10]応用Prolog						
[11]応用微分	0.01	2.70	1.40	0.07	1.52	2.52
[12]応用Boyer	0.01	3.31	3.45	0.03	1.47	3.04
平均	0.01	2.73	2.72	0.05	1.30	3.15

＊各欄の数値は，ＫＣＬ／ＡＳ３０００の実行時間を１としたときの，それぞれの処理系での実行時間

＊各項目および平均は算術平均値

[1] Function call and return Tarai and Tak
[2] List Processing Tarai with list data Reverse and Nreverse
[3] String processing Tarai with string data
[4] Floating and Bignum Tarai with floating number Tarai with bignum
[5] Array munipulation Bubble sort
[6] Property list manipulation Calculate by property list
[7] Mapping function Tree construction
[8] Sorting
[9] TPU
[10] Portable Prolog
[11] Data-oriented programming Differentiation
[12] Boyer's Teorem Prover

4．おわりに

京都大学で開発されたＫＣＬの実システムへの適用性を当社のＥＷＳ（ＡＳ３０００シリーズ）上で評価した。

実システム開発のために必要なグラフィクス，マルチウインドウの機能を追加して，ユーザ・インターフェイス開発環境を整えた。また，ベンチマークを行い，性能的には満足することを確認した。

今後は，ウインドウ・システムとしては，Ｘｗｉｎｄｏｗ，グラフィクスとしては，ＧＫＳをサポートしていく予定である。なお，ＫＣＬの使用を許可してくださった京都大学数理解析研究所の湯浅太一，荻谷昌己先生に感謝する。

＜参考文献＞

［１］湯浅太一，荻谷昌己："Ｋｙｏｔｏ Ｃｏｍｍｏｎ Ｌｉｓｐ Ｒｅｐｏｒｔ"，帝国印刷，１９８５．

［２］Ｇｕｙ Ｓｔｅｅｌｅ Ｊｒ："Ｃｏｍｍｏｎ Ｌｉｓｐ Ｔｈｅ Ｌａｎｇｕａｇｅ"，Ｄｉｇｉｔａｌ Ｐｒｅｓｓ，１９８４．

［３］奥野博：第三回ＬＩＳＰコンテストと第一回ＰＲＯＬＯＧコンテストの課題案，情報処理学会記号処理２８－４（１９８４）

［４］奥野博：Ｔｈｅ Ｒｅｐｏｒｔ ｏｆ Ｔｈｅ Ｔｈｉｒｄ Ｌｉｓｐ ｃｏｎｔｅｓｔ ａｎｄ Ｔｈｅ Ｆｉｒｓｔ Ｐｒｏｌｏｇ ｃｏｎｔｅｓｔ，情報処理学会記号処理３３－４（１９８５）