ICOT Technical Memorandum: TM-0225

TM-0225

AI ワークステーション PSI の プログラミング環境

中澤　修，飯間　豊
(沖電気工業)

August, 1986

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

AIワークステーションPSIのプログラミング環境

沖電気工業(株) 総合システム研究所　　中澤 修
コンピュータシステム開発本部　飯間 豊

1．まえがき

近年，人工知能分野に関する研究開発が盛んとなり，人工知能用プログラミング言語としてLISP，Prolog等が広く用いられている。LISPについては，種々の計算機上で処理系が稼動しており，また専用ワークステーションも開発され，整った環境下での研究開発が可能となっている。しかしながら，Prologについては，LISPに比しての歴史の浅さもあり，実用的処理系がごく限られ，Prologで記述された実用的かつある程度の規模をもつプログラムは皆無に近かった。

Prologによる実用的プログラミングを可能とするため，沖電気は，第５世代コンピュータプロジェクトの一環としてPrologベースのAIワークステーションPSI(Personal Sequential Inference Machine)，及びPSI 上で稼動するオペレーティング／プログラミングシステムSIMPOS(Sequential Inference Machine Programming Operating System) の開発に参画した。

SIMPOSは，PSI 上に論理型言語ESP(Extended Self-contained Prolog) に基づく単一言語システムを提供し，個人用の高度なプログラム開発環境を実現するための基本ソフトウェアである。SIMPOSの開発は，58年度から新世代コンピュータ技術開発機構を中心にメーカ５社との共同で始まり，現在も高機能化に向け改良・拡張が行なわれている。

本文では，プログラミング環境を中心としたSIMPOSの概要，SIMPOSの記述言語であるESP 及びマンマシンインタフェースにおいて中心的役割りを果すウインドウシステムについて述べる。

2．システム概要

SIMPOS[1]は，PSI 上で稼動するオペレーティングシステム及びプログラム開発上必須となる各種ツールを提供するためのプログラミングシステムの総称であり，人工知能分野における本格的プログラムを作成するユーザを想定し，使い易くかつ柔軟性に富んだ環境を提供することを目的としている。SIMPOSの構成を図１に示す。

SIMPOSには，以下の主な特徴がある。

① シングルユーザ／マルチプロセス環境

シングルユーザ／マルチプロセスオペレーティングシステムであり，対話的プログラムの処理を主要なものとしている。

② オブジェクト指向構成

オブジェクト指向概念に基づいて構成されており，高度のモジュラリティを有す。保守性・柔軟性に優れ，既存プログラムの再利用も容易である。また，ユーザに対して簡明なインタフェースを提供する。

③ 論理型言語記述

システムすべてが論理型言語ESP で記述されており，読解性に優れ，信頼性向上につながる。

④ システムサービス

SIMPOSの豊富なシステムサービスは，オブジェクトという統一的枠組みでユーザに提供され，アプリケーションプログラムからの利用が容易である。また，ユーザによるシステムサービスの追加，変更も可能な柔軟なシステムである。

⑤ マルチウインドウ

並列実行可能な複数のプログラムを複数のウインドウを介すことにより，同時に操作可能である。また，それらの実行状態を視覚的に制御可能とする。さらに，マウス／メニュー指向のインタフェースを提供し，簡便な対話機能を実現する。

⑥ ネットワーク

LAN(Local Area Network) により，マシン間での資源や情報の共有及びファイル転送等が可能である。

3．記述言語　ESP

ESP [2] は，論理型プログラミングとオブジェクト指向プログラミングの両パラダイムを融合した言語である。すなわち，Prologをベースとした述語論理に基づく論理型言語に，オブジェクト指向概念を導入したものであり，システム記述等の大規模プログラムの開発に不可欠なプログラムモジュール化の機能を実現している。

論理型言語及びオブジェクト指向言語としての特徴は，以下の通りである。

① 論理型

ESP の論理型言語としての特徴は，PSI の機械語であるKL0 [3] の基となった標準的Prolog(DEC10-Prolog) [4]

とほぼ同じであるが，以下の機能が追加されている。
・遠隔カット等の実行制御機能
・恒久的データ保持のためのデータ構造
・ハードウェア制御用組込み述語
一方でESP には，DEC10-Prologの入出力組込み述語に類するものが存在しないが，これはESP がシステム記述用言語であるためである。
上記の様に詳細において相違点はあるものの，論理型言語としてのESP をPrologと考えて差し支えない。

② オブジェクト指向

オブジェクト指向[5],[6]とは，抽象データ型の概念の拡張であり，データとそれに対する手続きとを一体化したオブジェクトによるプログラムのモデル化を図るものである。

ESP は，クラス概念による高度なプログラムのモジュール化機能，差分プログラミングを実現するためのクラスの多重継承機能及びオブジェクトの動的状態を保持するためのスロットと呼ばれる変数を有す。また，オブジェクト指向の実行メカニズムであるオブジェクト間でのメッセージ通信は，メソッドと呼ばれる一種のPrologにおける述語呼び出しによって行う。
以下にESP プログラムの記述形式を示す。

```
class クラス名
   [継承クラス宣言]
   [クラススロット定義]
   [クラスメソッド定義]
instance
   [インスタンススロット定義]
   [インスタンスメソッド定義]
local
   [ローカルメソッド定義]
end.
```

ESP によるプログラミングを簡単に説明すると，プログラムのモジュール化は，オブジェクト指向概念のクラスにより行い，クラス内のプログラム記述は，Prologにより行うということになる。
プロセス管理を例とする。一般のオペレーティングシステムにおいては，プロセス管理はプロセス管理プログラムとプロセス管理テーブルにより構成される。一方，

オブジェクト指向におけるプロセス管理は，プロセスと呼ばれるクラスのみにより構成される。このクラス内に管理テーブルとしてのデータ構造及びそれを操作するプログラムが包含される。クラスの外からは，クラスのもつデータ構造は見えず，それを操作するためのメソッドのみが意識される。このため，クラス内を完全なブラックボックスとするモジュラリティの高いプログラミングが可能になる。

ESP は，上記の様に記述性・モジュール性に優れた言語であり，SIMPOSの記述言語であると同時に，アプリケーションプログラム記述のためのユーザ言語としても使用されている。

4．オペレーティングシステム

SIMPOSのオペレーティングシステム部は，資源管理，実行管理及び入出力メディアシステムにより構成されている。それぞれの構成要素は，以下の通りである。

① 資源管理
　プロセサ，メモリ，デバイス管理
　IPL プログラム
② 実行管理
　プロセス，ストリーム，プール，ワールド，
　ユーザ，時計管理
③ 入出力メディアシステム
　ウインドウ，ファイル，ネットワークシステム

上記の様にオペレーティングシステムの構成要素は，他のオペレーティングシステムと大きく変るものではない。SIMPOSの特徴は，オブジェクト指向を利用したその構成法にある。

一般のオペレーティングシステムにおけるシステムサービスは，システムコールあるいは，それに付加的機能を追加した手続きである。一方，SIMPOSにおいては，システム内の資源を論理的に表現するオブジェクト及びオブジェクトを動的に作り出す型紙であるクラスとして提供される。

SIMPOSのシステムサービスの一般的利用手順は，
1)クラスよりオブジェクトを作成
2)オブジェクトのメソッド呼び出し
3)オブジェクトの終了処理メソッドの呼び出し

により行う。
例えば，プロセスの作成，実行，終了は以下の様に行う。

1):create(#process,P)
クラスprocess からプロセスＰを作成

2):activate(P,Program)
プロセスＰにおいて，プログラムProgram を実行

3):terminate(P)
プロセスＰを終了

オブジェクトにデータ構造と処理がすべて含まれているため，ユーザが種々のデータ構造を意識する必要がなく，SIMPOSのシステムサービス機能は，少ない知識で容易に利用できるものである。

また，システムが提供するサービスに不足がある場合，システム提供クラスを継承して新たなユーザクラスを定義するという方法により，比較的簡単にシステムのカスタマイゼーションが可能である。

５．プログラミングシステム

ユーザが，プログラム開発を行う際に必要となる支援ツールの集合である。ツール個々の操作性，相互の統一性がプログラミング環境全体を左右する。このため，プログラミングシステム全体は，以下の方針により設計されている。

○マルチウインドウ環境下での使用
マルチウインドウ機能を利用したユーザインタフェースを提供し，複数のツールのウインドウを同時にディスプレイ上に表示することにより，相互に関連させながら並行して使用できる開発環境を実現する。

○メニュー指向
コマンド入力は，マウス選択方式によるメニュー指向とし，マニュアルレスオペレーションを可能とする環境を提供する。ただし，メニュー方式にも種々の問題があるため，キーボード入力を用いたコマンド入力等との協調を図る。

以下に代表的ツールをあげる。

①コーディネータ
一般のシステムにおけるコマンドインタプリタに類するものである。メニュー指向により，システム提供またはユーザ作成ツールの起動，実行制御及び終了処

理を行う。また，UNIX風のシェルを有しており，これによるプログラム実行も可能である。

コーディネータは，常にバックグラウンドで実行されているプログラムであり，任意時点での呼び出し，実行中のプログラム強制終了等の制御が可能である。また，コーディネータ経由で呼び出されるウインドウマニピュレータにより，表示されている各種ツール用のウインドウの大きさ，位置等のレイアウト変更も容易である。

② エディタ

各種の計算機で広く使用され，使い易さで名高いスクリーンエディタEMACS に類似したエディタである。100種以上の豊富なコマンドを持ち，マルチプルバッファ機能，ユーザによるコマンドの追加，変更機能等を有す。また，カナ漢字変換方式による日本語入力，編集が可能である。

この他に，ESP プログラムの構文に従った自動的編集が可能な構造エディタも備えている。

③ ライブラリ

システム中に存在するすべてのプログラム（クラス）を一種のデータベースとして管理する。SIMPOSを構成するシステムクラス及びユーザ作成によるユーザクラス両者が全く同様に管理される。

ライブラリ操作を行う対話的プログラムをライブラリアンと呼ぶ。これは，ソースプログラムの解釈実行コードへの変換，コンパイル，ライブラリへの登録／削除，管理情報の検索等の機能を提供する。

④ デバッガ／インタプリタ

ESP プログラムのデバッグを支援するプログラムである。ライブラリに登録されているプログラムの実行，解釈実行形式のプログラムに対するデバッグ機能の提供を行う。ライブラリに登録されているプログラムには，機械語形式のものと解釈実行形式のものがあり，その形式に従ってハードウェアによる直接実行及びインタプリタによる解釈実行が自動的に選択される。また，解釈実行形式の場合，トレース，変数値の表示等のデバッグ機能，さらにオブジェクトの状態を調べるインスペクタ機能，性能評価用の性能測定機能を利用できる。

プログラム開発時には，エディタでソースプログラムの解析を行いつつ，デバッガを動かすという様に各ツールを同時に実行し，ツール間を飛び歩くという方法により効率的作業が可能となる。

6．ウインドウシステム

ウインドウシステムは，良好なプログラミング環境を実現するための重要な機能を提供するものとして，多くのワークステーション等で採用されている。

SIMPOSにおけるウインドウシステム[7],[8]の主要な目的は，以下の通りである。

○マルチプロセス環境に対応する端末の複数化

マルチプロセス環境下においては，ユーザは複数のプログラムを同時に実行可能である。これら複数のプログラムを同時に操作するために，論理的端末の複数化，すなわちマルチウインドウ化が必要となる。

これは，机上において多くの本や資料を拡げ，それらを互に参照しつつ作業する環境のワークステーション上への拡張である。

○入出力プログラムの簡単化

操作性を重視した対話的プログラムにおいては，端末に対する入出力がプログラムの多くを占めることとなる。一つのプログラムにおいて複数のウインドウの使用を可能とすることにより，入出力の簡単化を図る。

6．1　機能

本システムの主な機能は，以下の通りである。

○重ね合せマルチウインドウ

ウインドウ相互の重ね合せが自由であり，完全ビット単位の位置／大きさで論理的に無限個のウインドウ作成が可能。

○ウインドウの階層管理

ウインドウを階層的に管理することにより，ウインドウ内をさらにマルチウインドウ化することが可能。

○ボーダ，ラベル，マージン

ウインドウの視認性向上のためのボーダ，ラベル表示及びマージン設定が自由。

○文字，ビットグラフィック出力

ウインドウごとに漢字を含む異なるフォントによる

文字出力及び直線，円弧等の図形出力。

○ユーザ定義ウインドウ作成機能

システムで提供する部品クラスを組合せて継承することにより，容易に専用ウインドウクラスが作成可能。また，ユーザのプログラミングにより全く新しい属性を持つウインドウの作成も可能。

○ウインドウマニピュレータ

マウス及びキーボードによるウインドウ属性（位置，大きさ等）変更のための対話的操作機能。

○フォントエディタ

マウス指向の対話的文字フォント編集機能。

6．2　構成

本システムの構成を図２に示す。ウインドウマネージャは，独立したプロセス下で実行されるプログラムであり，主にユーザプロセス，デバイスハンドラプロセス間のプロセス間通信，同期を司どるものである。ウインドウの実体は，ウインドウクラスのインスタンスであるウインドウオブジェクトであり，ウインドウに対する入出力等の要求はすべてウインドウオブジェクト内で処理される。すなわち，ウインドウシステムには，単一の管理プログラムというものは存在せず，個々のオブジェクトに埋め込まれた管理機能が自律的にシステムとして統一のとれた管理を実現している。

ウインドウクラス群は，図形出力あるいは文字入力といった単機能について記述する多数の部品クラスと部品クラスのいくつかを選択的に継承することで一つのウインドウとしてまとまった機能を提供する完成品クラスとから構成されている。

以上の様な構成により，柔軟性に富み，システムとしての機能拡張が容易であり，またユーザによる機能追加も簡単である。

6．3　ウインドウ管理

(1) ウインドウ階層

作成されたウインドウは，スクリーンと呼ばれる端末画面全体を表すシステムウインドウをルートとする階層を構成する。図３は，ウインドウの階層の例であり，図４は，その時の表示画面である。スクリーンの一部を占めるウインドウA及びBは，スクリーン直下の階層に置

かれる。複数のウインドウの相互関係に依存する管理は，それらウインドウの上層に位置するウインドウにより行われる。図において，ウインドウAとBとの重なりは，スクリーンにより管理され，またウインドウCとDとの重なりは，ウインドウAにより管理される。

この様にウインドウを階層管理し，重なり管理をその上層のウインドウに受け持たせることにより，何段階にも及ぶ再帰的なマルチウインドウ化が容易に実現できる。

(2) 出力管理

他のウインドウにより隠されていないウインドウのイメージは，ビットマップディスプレイ装置が有するフレームメモリ上に直接作成可能である。一方，隠されているウインドウのイメージは，その全体または隠されている部分をいずれかの場所に退避しておく必要がある。PSIのビットマップディスプレイ装置には，フレームメモリの数～数十倍の容量のビットマップメモリが搭載されており，各ウインドウのイメージは，ビットマップメモリ上に割り当てられるセーブエリアに退避される。

ウインドウ相互の重なりの変更は，図5に示す通りセーブエリアとフレームメモリとの間のメモリ転送により実現される。

ウインドウに対する文字，図形の出力は，セーブエリアないしはフレームメモリの書き換えで実現され，これはウインドウが隠されているか否かによらず可能である。すなわち，隠されていないウインドウへの出力はフレームメモリの書き換えにより，隠されているウインドウへの出力はセーブエリアへの書き込みにより実現される。

しかし，ユーザの立場に立つと，隠されたウインドウへの出力はユーザに見られずに終わる可能性があるという問題がある。そこで，隠されたウインドウに対する出力処理方式のオプションを設け，プログラムの種別によりこれを任意に選択可能とした。

出力処理オプションには，以下の4通りがある。

① 出力要求を発行したプロセスを待ち状態とする。当該ウインドウが隠されていない状態となって初めて出力処理が行われ，待ち状態が解除される。オプションのデフォルト状態である。

② そのままセーブエリアへの出力を行う。この時，当該ウインドウの隠されていない部分への出力は，スクリーン上に正しく反映される。

③当該ウインドウを強制的に隠されていない状態とした後，出力を行う。
④隠されたウインドウへの出力要求があることをユーザに通知するための一時的ウインドウを表示する。ユーザは，マウス操作により，①または③と同様の処理を選択できる。

この様な出力処理オプションを用いることにより，当面使わないツールのウインドウを隠すことにより，そのツールの実行を最初の表示時点で停止させることができ，視覚に対応したプログラムの実行制御が可能となる。

(3) 入力管理

入力管理の主な役割は，キーボード及びマウスからの入力の送り先ウインドウを決定することである。キーボードとマウスは，異なった性格を持つ入力装置であるので，その決定法も全く相違している。

1)キーボード

キーボード入力は，ウインドウ階層中に常にただ一つ存在するセレクテッドウインドウと呼ばれるウインドウに送られる。セレクテッドウインドウは，キーボード入力機能を持つウインドウであり，ウインドウ相互の重なりの最上層に位置するものである。

セレクテッドウインドウは，ユーザのマウス操作により自由に変更可能である。

2)マウス

マウスは，基本的にウインドウ内の位置を入力する装置であり，マウスを移動することにより，種々のウインドウへ入力を与えられるという特性を有している。このため，キーボードの様にウインドウ階層の最上層のウインドウにのみ入力を行うという方式とは，異なった制御が必要となる。そこで，ウインドウシステムでは，マウスマーカがどのウインドウ上にあるかを常に監視し，マウス入力をマウスマーカの下にあるウインドウに送るという方式を採用している。

7．あとがき

以上述べてきた様にSIMPOSは，論理型言語＋オブジェクト指向という新しいアプローチを採用して実現したものであるが，充分な処理速度が要求される下層のソフトウェアも，また操作性の良さが要求される上層のソフト

ウェアをも完全に実現することができた。

現在のSIMPOSの規模は，ESP の記述で

ソースプログラム	約２３０，０００行
クラス総数	約１，４００個
メソッド総数	約１８，０００個

となっている。

SIMPOSは，現在第 2.5版がリリースされており，新世代コンピュータ技術開発機構をはじめとし，メーカ８社及び大学等の研究機関で使用されている。しかしながら，マンマシンインタフェースに係わるソフトウェアについては，実際的環境下での使用経験に基づく継続的な改良が必要である。今後は，SIMPOSをより使用環境の良いシステムとしていくために，ユーザからのフィードバックを反映した一層の改良・拡張を行う予定である。さらに，ESP に関しては，言語仕様の見直しを行い，汎用性の高いプログラミング言語として成長させていく予定である。

最後に本システムの開発に当たり，御指導いただいた新世代コンピュータ技術開発機構第４研究室の研究員各位に厚く感謝の意を表します。

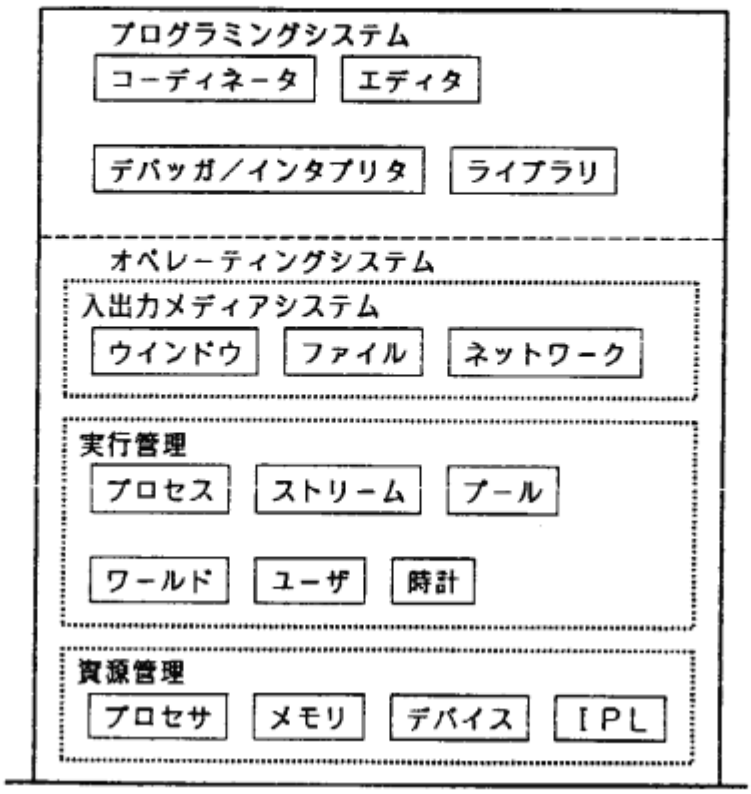

図1　SIMPOSの構成

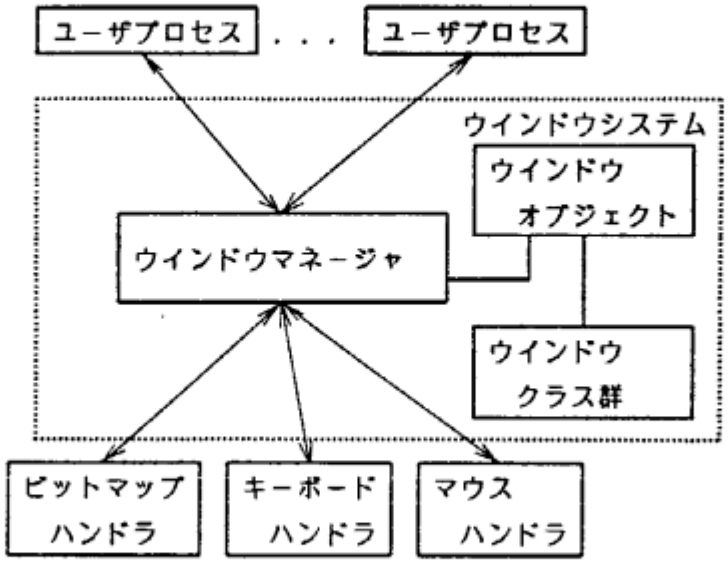

図2　ウインドウシステムの構成

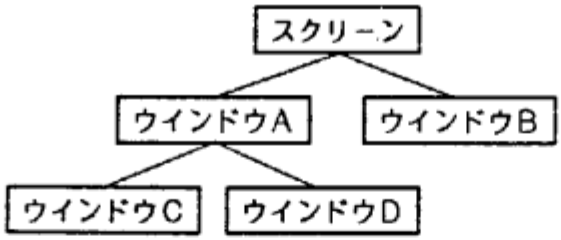

図3　ウインドウの階層例

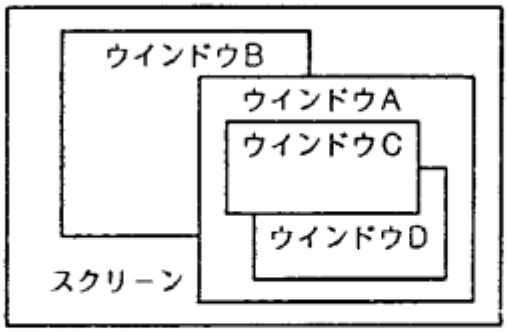

図4　表示画面例

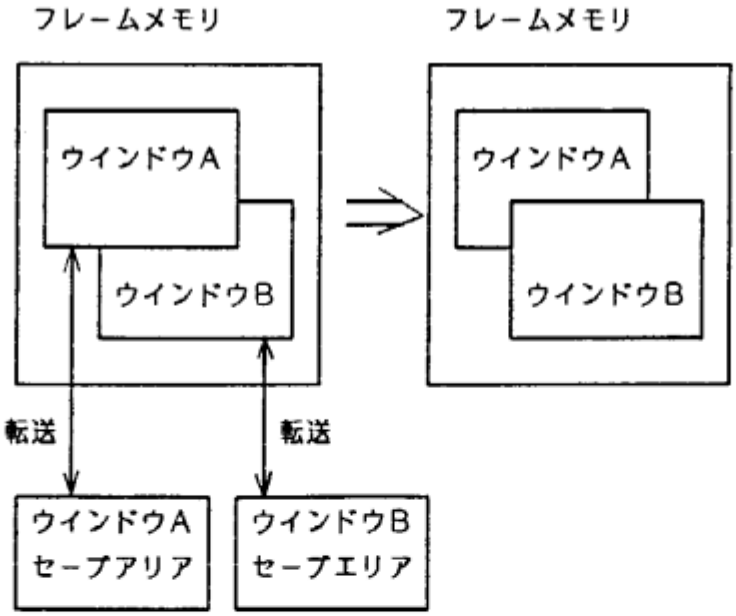

図5　ウインドウ間の重なりの変更

書式－4

参考文献・引用文献用原稿用紙

沖時報編集委員会

文献番号	執筆者名	〔雑誌〕論文題目 〔単行本〕書名 版・巻	出版者	雑誌名 章・節	巻 Vol.	号 No.	ページ P. P.	発行年 (西暦)
1	石崎，近山 他	SIMPOS（逐次型推論マシンPSIのOS）		情報処理学会 コンピュータシステム・シンポジウム			107～113	1985
2	T. Chikayama	ESP Reference Manual		ICOT Technical Report		44		1984
3	M. Yokota, A. Yamamoto et.al.	The Design and Implementation of a Personal Sequential Inference Machine : PSI	OHMSHA	NEW GENERATION COMPUTING	1	2	125～144	1983
4	D. L. Bowen	DEC system-10 Prolog User's Manual	Univ. of Edingburgh					1983
5	A. Goldberg, D. Robson	Smalltalk-80 The Language and Its Implementation	Addison-Wesley					1983
6	D. Weinreb, D. Moon	Flavors : Message Passing in the Lisp Machine		MIT. A. I. Memo		602		1980
7	J. Tsuji, T. Kurokawa, Y. Iima, O. Nakazawa et.al.	Dialogue Management of Personal Sequential Inference Machine (PSI)		Proceedings acm'84 Annual Conference The Fifth Generation Challenge			314～322	1984
8	飯間，中澤 他	SIMPOSのウインドウ・サブシステム		情報処理学会全国大会予稿		4E-6	295～296	1984