ソフトウェア基礎論 22－7
（1987. 10. 1）

汎用の論証支援システムの構想とその実現法

沢村 一　　南 俊朗

富士通㈱国際情報社会科学研究所

これまで基礎論理系が固定されている特定目的のための論証支援システム（証明チェッカー、コンストラクタ）については多くの研究がなされてきている。このような研究の動向とは対照的に、本論文では、それぞれの対象ごとにより適した記述形式・推論形式を定義することを許し、さらにその下での論証による問題解決を可能にする汎用の論証支援システムを考察する。

論文の前半では、「すべての議論領界はそれぞれの論理構造をもつ」という仮説を背景に、このような汎用の論証システムの意義、必要性を論ずる。後半では、汎用の論証支援システムの構想の実現に向けて、主として次の4つの特徴をもつ論証システムの提案を行う：(1)論理記述言語、(2)論理的思考（計算）のための作業シートに基づく証明方法論、(3)種々の理論間の関係依存性の保持、(4)論証システム向きインタフェース。

Conception of general-purpose reasoning assistant system and its realization method

Hajime Sawamura and Toshiro Minami

International Institute for Advanced Study of Social Information Science (IIAS-SIS), FUJITSU LIMITED, 140 Miyamoto, Numazu, Shizuoka 410-03, Japan

Much work has been done on special-purpose reasoning assistant systems whose underlying logics are fixed. In contrast with such a research trend, this paper is devoted to general-purpose reasoning assistant system which will allow us to define our own logical systems relevant for the objects in the problem domains and to reason about them.

In the first half of the paper, the thesis that every universe has its logical structure and our conception of general-purpose reasoning assistant system are described. In the latter half, a feasible scheme to realize it is proposed, placing emphasis on the following four points; (1) logic description language, (2) proving methodology based on working sheets for logical thought (or logical calculi), (3) maintenance of a relational dependency among various theories, (4) human-computer interface for the reasoning system.

1. はじめに

我々は、人間の知的活動は記号活動であるという認識から、人間の知的な記号活動を計算機の支援の下で行うための原理、および方法論に興味がある。我々は、問題を認識したとき、それを記述し、解決するためにある種の論証を行う。それはある種の記号システムにおいてなされるのが普通である。したがって、そこにおける記号およびそれらを操作するための基本となる記号操作とは何かを明らかにすることは問題解決の第一歩ともなる。そして、それらが計算機の中に実現されるとき初めて、十分な支援を計算機より受けることができる。

これまで、計算機科学と論理学の接点において、問題の論理学的認識による形式化と、解決が与えられてきた。そこでは、問題ごとに自然な論理が対応していた。他方、すべての問題に対して普遍な記号および論理システムというものを求めることは不可能なことのように思われる。したがって、各自の認識を自由に記号システムとして表現でき、その記号システムの下で記号および論理操作を支援してくれる、論証支援システムというものが強く求められる。

これまでの証明チェッカー（例えば、LCF/ML、PL/CV2、MIZAR 、FOL 、HOL 、AUTOMATH、EKL 、CAP-LA、BOYER-MOORE 定理証明機、BERTIE- IIなど）は、それらのいくつかは高階論理表現を可能にするものの基本的には定められた論理系の下で、計算機に証明をチェックさせることを主目標としていた。これらの有利な点は論理系が固定されているが故の強力な証明チェック、構成能力にあるといえる。ここで、記号系あるいは論理系そのものも変えることができ、ユーザに推論の自由度を与えることができるならば、さらにその利用度が高まるのみならず、いろいろな応用にも耐えられるようになるであろう。本論文では、ユーザによって対象に関する記号系と推論系が定義されると、その下での論証の構成を支援する汎用の証明構築支援システムの構想〔1〕について述べ、それを実現するに当たって重視した4つの特徴的な側面を説明する。

2. 汎用の論証支援システムの意義

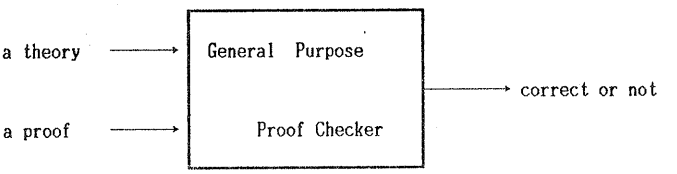
汎用性の意義を、いくつかの必要性を示唆する事項を取り上げて説明する。

(1) 計算機科学や人工知能研究における論理学的方法論の立場から

計算機科学や人工知能研究の初期からすでにこれらに対して論理学的方法論の重要性と有効性は認識されてきた。論理学的方法論というのは大雑把に言えば、まず対象世界を公理的に捉えることに始まり、そして問題を解く（計算する）ということをその公理系の下での論証によって行おうとするものであると言える。しかしながら、そこで要求される論証の形態は問題の性質によって異なってくることは避けられない〔2〕。

(2) 計算機支援プログラミング (Computer Aided Programming)

プログラミング方法論の一つに、今抱えている問題に適切な理論（定義、公理、定理等）をまず与え、次にそれをインプリメンテーションにより適した具体的対象に適用することによってプログラムの作成を行う方法論があり、いくつかの変種が知られている。この場合、問題によってその背後にある理論は問題毎に一般に異なる。したがって、このようなプログラミングを支援するには、おのずとさまざまな理論が扱える汎用の証明チェッカー(general-purpose proof checker)が必要になってくる〔3〕（汎用の証明チェッカー自身を計算機支援プログラミングで構成するには、種々の理論を扱う理論の理論(a theory of theories)が必要になる〔4〕）。すなわち、汎用の証明チェッカーは、問題を規定する理論とそこにおける主張の証明を入力として、その正当性を検証するものである（下図参照）。



(3) 論理モデルとその構築

上記(1)で述べたことは、計算機科学や人工知能の研究に限定されたことではない。どの分野であれ、我々の知識がまとまって体系化されたときは一つの公理系を形づくると思われる。あるいはむしろ、知の体系化として公理系という形への体系化を指向しているといった方が正確かもしれない。実際、公理化が最も顕著に現れる数学は言うまでもなく、他の科学、例えば、言語学、生物学、社会科学、物理学、音声学、さらには論理とは異質の思考が要求されると言われる音楽、芸術の分野に対してさえそれぞれの対象に関して公理的な把握、すなわち論理モデルの構築が試みられているのである。論理モデルの構築に当たっては、観察された事実を説明する公理、さらには推論の規則も不確かな状況の中いで、その観察された事実を説明する理論／モデルを作り上げることになるため、公理、推論方法を自由に変えられることが必要になる。我々の汎用の論証支援システムによる論理モデルの構築支援は知の体系化へのこのような過程の支援をも意図している。

以上述べてきた事柄の立脚点は、「すべての議論領界はそれぞれの論理構造をもつ」〔5〕という考えにあり、それを計算機の世界に実現するものが汎用の論証支援システムであると言える。

3. 汎用の論証支援システム（メタシステム）の構想

本節では、汎用の論証支援システム実現の核となる部分の説明を与える。それは次のようにまとめられる。

メタシステムの機能：

1. 記号解釈系

   メタシステムは入力を手続き的に解釈し、内部に論理系（より広義には記号系）を理解する能力を具象化する。例えば、

   ① 論理Lで定められた記号、その記号で書かれた「項」、「論理式」、「公理」、「推論規則」、「証明」等の概念を理解する。

   Syntax checker の生成

   ② このとき、論理Lでの論証を支援できるように、公理、定理、証明戦略（証明テンプレート、書換規則）等のデータベース、知識データベースの枠を準備する。

   データベースの生成

   ③ 論理Lの下で論証が正しく行われたかを検証する。

   Proof checker の生成

2. 記号操作系

   メタシステムは基本的記号操作機能を有する。

   ① 置き換え操作、② 代入操作、③ 合成操作、④ 名前換え操作、⑤ 分離操作、等

この他に、汎用の論証支援システムは少なくとも次の二つの機能は具備すべきものと考えられる：① 証明戦略記述言語（例えば、MLのような）、② メタ証明のための自然演繹法（図1参照）。

ここで記号解釈系とは、論理系記述言語で書かれた論理系定義を手続き的に解釈し、これらの記号を理解できるシステムを作り出す働きをもつ。例えば、一階論理の項の定義を次のように、適当な論理系記述言語で与えたとしよう。

(1) T : var -> T : term

(2) T : const -> T : term

(3) T , ・・・ ,Tn : term, F : func -> F(T , ... ,Tn) : term

すると、論証支援システムは次のようなsyntax checkerを構成する（本論文では手続き表現に、Prologを採用することにする）。

```
term(T) :- var(T).
term(T) :- atomic(T).
term(T) :- T =.. [F |Args] , term1(Args).
term1( [] ).
term1( [H |T ] ) :-  term(H), term1(T).
```

また、推論規則を前提と結論の手続き的な関係とみなすと、例えば、 推論規則「 P(S), S = T |- P(T)」は、

```
inference-rule-1(Term1, S = T,Term2) :- term(Term1), replace(Term1, S, T, Term2), term(Term2).
```

と解釈される。ここで、P(S), P(T)はS, Tが出現するタームの上を動くメタ変数であることが認識されているものとする。また、規則「 P, Q |- P ∧ Q」は、

```
inference-rule-2(Form1, Form2, Form1∧Form2) :-  formula(Form1), formula(Form2).
```

と解釈される。ここで、P, Qは論理式の上を動くメタ変数であるとする。また、「replace 」は論証支援システムがあらかじめ持っている基本的記号操作機能の一つとする。さらに、「term」, 「formula 」は論証支援システムによってすでに生成されたsyntax checkerの一部である。

このとき、証明の概念を次のような形式をもつ行のリストとして定義するならば、

```
<number> |- <formula> by <justification>.
```

proof checker の概形は次のようになるであろう。

```
proof-checker( [] , _ ).
proof-checker( [H |T ] , Stack) :-
    (axiom(H) ; inference(H, Stack)), proof-checker(T, [H |Stack ] ).
```

ただし、「axiom 」、「inference 」は論理系記述言語で記述された公理、推論規則によって定まる述語である。

4. 実現に向けての方策

第2節(2)で述べた理論からのプログラミング構成論にしたがって汎用の論証支援システムを構成することが考えられる。すなわち、まず理論の理論 (a thoery of theories) あるいは証明の理論(a theory of proofs)を定義し、それを基に汎用の論証システムを構成するという道筋である。実際、この方向での試みは論文〔3〕に、関連する理論は〔4〕に見受けられる。本論文では、第3節で述べた構想の実現に向けてより具体的なアプローチを取り、さらに以下の4.1 ～ 4.4節で述べる4つの特徴をもつ汎用の論証支援システムの構成を考える。まず、論理系を定義するのに必要となる概念を整理しておく。

4．1 論理系とその記述言語

4．1．1 論理系

ほとんどの場合、論理系は以下の項目のいくつかによって定まるものと考えてよい。

(1) 型の定義、(2) 対象記号(with or without type)：①変数記号、②定数記号・関数記号（アリティと結合順位付）、③等号記号・述語記号（アリティ付）、④論理記号（結合順位付）、⑤補助記号、(3)メタ記号(with or without type)（(1)および(2)の①、②、③に対するメタ記号）、(4)項・論理式の帰納的定義、(5)定義される記号、(6)省略記法の規約、(7)公理、(8)推論規則、派生規則、(9)書換規則、(10)証明の定義、等。

また、論理系に関して成立するメタ定理が知られている場合、それをも列記した方が論理系の定義としては完結する。この他に、プラグマティックな観点から、論理系固有の証明戦略とか証明テンプレート、証明手続きなどを論理系の定義に付随させておくと利用に当たって都合がよい。

4．1．2 論理操作のための基本的記号操作機構

論理操作を行うためには、まず記号を認識するメカニズムとそれらの記号を操るための原始的な記号操作機構を明らかにしなければならない。

〔1〕記号認識機構

記号認識機構で最も重要になるのは次の三つである：①オブジェクトとメタの区別、②オペレータの優先順位の記述と識別、③パターンマッチメカニズム。①は、例えば、定理を表現するときそれがオブジェクトとしての定理なのか、定理図式なのかを区別するときに必要になる。②は、補助記号としての括弧によってオペレータ優先順位のあいまいさを避けることができるが、式を読み易くするためには必要になる。③は、すべての記号処理に必ず必要になるものである。メタ的な論証システムでは、任意に与えられた記号のセットに対して適用可能な、抽象的パターンマッチングなるものが必要になってくる。これは、記号どうしの代入可能性を定義することにより解決されることになる。

〔2〕基本的記号操作機構

これをすべて枚挙することは難しいが、科学における記号操作は論理学が教える記号操作・論理操作に含まれてしまうと考えるならば次のもので十分であるかもしれない：①置き換え操作（replacement）、②代入操作(substitution)、③記号合成操作、特に、記号の並置操作（juxtaposition)、④分離操作（detachment）、⑤具象化操作(instantiation)（代入の特別の場合）、⑥単一化(unification)(結果的には、代入の特殊ケース）、⑦名前換え(renaming)（代入の特殊ケース）など。

これらの記号操作を任意にあたえられた基本記号の下で定義することは簡単なことではないが、少なくともそれらを実現するための抽象的なスキーマによって可能である。このとき、注意を払わなければならない、論理系固有の概念には次のようなものがある：(1)自由・束縛、(2) B is free for x in A(x)、(3) alphabetic variant 。(1)と(2)については式の帰納的定義にしたがって定義されるのが普通である。

4．1．3 論理系記述言語

我々は様々な論理（これには既存の論理に加えこれから形式化されるであろう論理も含む）を扱えるシステムを考えている。これは言い換えると、メタ論理と言ってもよいものであろう。すなわち、各種の具体的な論理を上から眺めて、それらの論理の外形的な共通性を一つの記述系で捉え、操作する方法を研究するというものである。ここでは、そのようなメタ論理のための基礎となる記述体系をまず考察する。論理系記述言語に最も強く要請されることは、言語の記述能力と記述された表現から手続きへの変換が可能なものでなければならないことである。このような言語の候補として、確定節文法(DCG)，Wijngaardenの2-レベル文法、UNIX上の字句解析機・構文解析機ジェネレータであるLex,Yacc、SmullyanのEFS 、Affix grammar 、Indexed grammar 、属性文法等が上げられるが、ここでは記述能力（特に文脈依存性）、記述のし易さ、読み易さ、構文解析機ジェネレータの生成可能性等を考慮し、しかもPROLOG処理系で利用できる確定節文法〔6〕に注目し、論理定義文法およびProlog言語への変換方法の研究が進められている。以下に、確定節文法による内包論理の定義例を示す。

〔内包論理の定義例〕型付の内包論理の構文を確定節文法で表現すると次のように定義できる。型の整合性のチェックの記述にはPrologの手続きを用い、これによって構文の文脈依存性を表現している。その他の部分は本来の内包論理の項の定義形式そのままである。

型の構文

```
type(e) → [e ] .
type(t) → [t ] .
type((X,Y)) → [( ] , type(X), [, ] , type(Y), [ )] .
type((s,T)) → [( ] , [s ] , [, ] , type(T), [ )] .
```

定数記号の構文

```
const(T) → const-sym, [: ] , type(T).
const-sym → [c1] ; [c2] ; ・・・; [cN] .
```

変数記号の構文

```
var(T) → var-sym, [: ] , type(T).
var-sym → [v1] ; [v2] ; ・・・; [vN] .
```

型がTの項、term(T)

```
term(T) → var(T) ;
          const(T) ;
          [ [ ] ,term(R),term(S) [ ] ] , [: ] , type(T), {R = (X,T),S =X} ;
          [lambda] , var(S), [. ] , term(R), [: ] ,type((S,R)) ;
          term(S), [= ] , term(S), [: ] , [t ] , {T = t } ;
          [ext ] , term(S), [: ] ,type(T), {S = (s,T) } ;
          [int ] , term(S), [: ] ,type(T), {T =(s,S)} .
formula → term(t).
```

4．2　論理的思考（計算）のための作業シートを用いる証明方法論

論証の過程は、初めから完全に行われるものではなく、仮定を設けたり、公理を引用したり、以前既に証明した結果を探索したりしながら、試行錯誤的に行われるものである〔7〕。また、証明は仮定や既に得られた結論などを組み合わせて新しい結論を導くという、いわゆるボトムアップ的な接近法、逆に目的とする式をそれを導くための十分条件に分解するトップダウン的な方法、それらを両方向から組み合わせる前進後退法〔8〕、役に立ちそうな中間結果を出しておく方法など多くの方法が混在するものである。証明のための作業シート（あるいは、落書帳〔9〕）(working sheet)とは、まさにこのような書いては消し、加えては修正するといった、言わばメモ用紙を使うような動作を支援するための編集環境であり、複数の作業用紙、計算用紙の比喩として生まれた証明方法論を実践するためのものである（図1参照）。さらに、複数の作業シートによる証明方法論は、次のような科学方法論をも計算機上において実践することを可能にするであろうことは見逃せない：

(1) 我々の数学、科学に対する知識は推測であり試行錯誤の過程を通じて成長するというLakatos〔10〕、Popperらの数理哲学の実践。

(2) 一つの理論に対しても複数の公理システムがあって（公理の相対化）、それら複数公理集団からの演繹結果の比較・検討を通して一つに体系化されていくというような生成主義、あるいは相互規定の相対論理（Relativistic logic by mutual specification）〔11〕の実践。

現在、証明断片を表す複数作業シートの分離、結合オペレーションが検討されており、結合オペレーションとして

完全一致の他、包含関係、具象関係、類比的関係などによる結合オペレーションが試みられている。

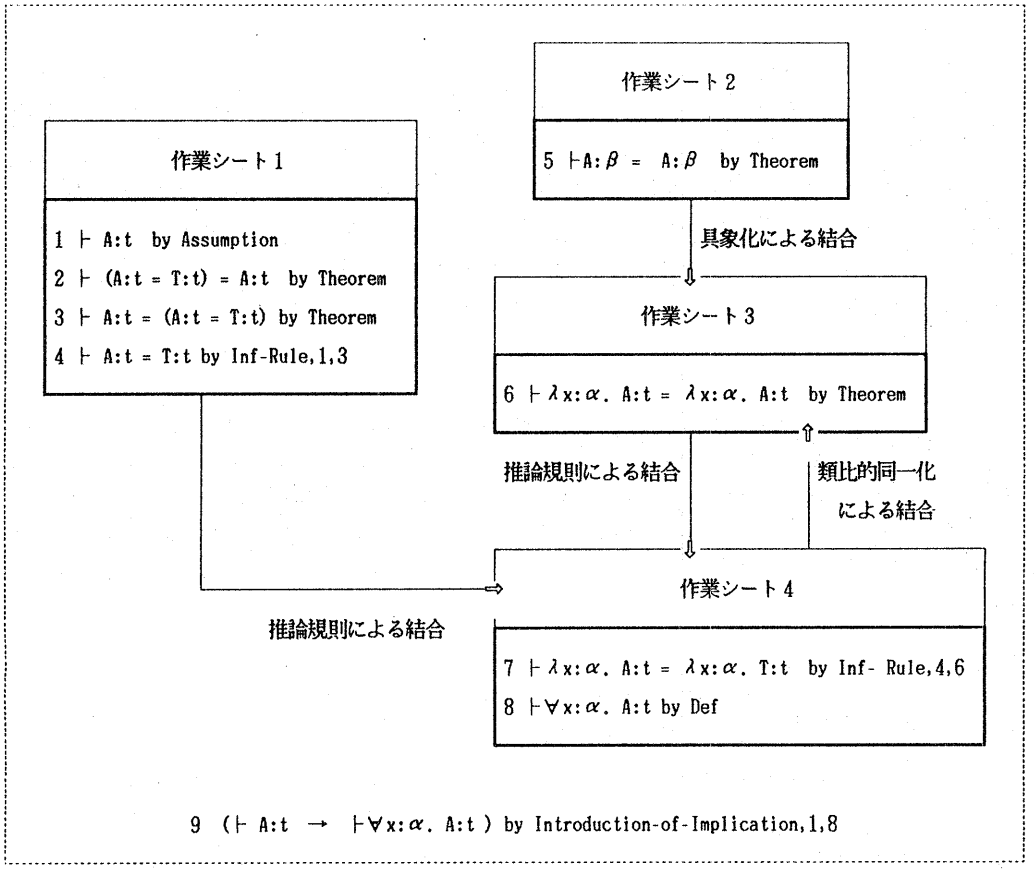


図1．内包論理の派生規則「⊢ A:t → ⊢∀x:α. A:t 」の作業シートによる証明例

4．3　種々の理論間の関係依存性の保持

我々の汎用の論証支援システムは、種々の記号系と論理系を許しているので様々な理論が理論データベースに存在することになる。そして、そのある理論の記号系あるいは公理系の変更に伴なって、理論に関する非単調性問題が必然的に現れてくることになる。ある理論が他のどの理論に依拠しているかとか、理論間の証明可能性等に基づく半順序関係、階層関係の保持が重大な問題として起こってくる。これに対処する方法の一つに、理論の変更によって新たな理論が形成されたものとみなし、消滅がなく単調に増加する理論群を保持するという考え方がある〔12〕。

4．4　論証支援システムのための固有のインタフェース

これまでの証明チェッカーとか証明機の研究ではそれらの機能面の研究のみに焦点が置かれていたために、論証のために適したインタフェースの研究は軽視されがちであった。我々が目的とするような論証支援システムでは、インタフェースは論証方法論とも密接に関連するものとして重視している。論証支援システムのためのインタフェースといっても、論証のための小道具としてのデスクアクセサリ(DA)から論証の発想を促すのに助けとなるような美学的な側面〔13〕まで非常に多岐にわたっている。4.3 節で述べた複数作業シートは論証のための発想の道具としての性格ももっ

ているものである。このとき、それらをどのようにレイアウトすれば証明の断片から我々の意図している証明の構造が見えてくるかといったことは論証の認知科学的な問題として興味深い〔14〕。

5. おわりに

本論文では、なぜ今汎用の論証システムなのかについて説明し、それを実現するに当たって要請される事柄につい述べてきた。ここにおける汎用性には二つの意味があることを注意しておきたい。一つは、問題領域ごとに論理を定義することができなくてはならないという意味での汎用性であり、他の一つは、でき上がった知の体系上での論証の支援のみならず、公理は一つの理論体系を組み立てるときの前提となる過程であり、論理モデルの構築過程においては、それは絶対的なものでなく、創造、選定、棄却が繰り返されるものであるから、そのようなダイナミニズムも扱うことができなければならないという意味での汎用性である。現在、以上に述べてきた4つの特徴をもつ汎用の論証支援システムの部分機能が実現されつつある。

謝辞

日頃御指導、御鞭撻をいただく北川敏男会長、および榎本肇所長に感謝いたします。北川会長には、論証システムに関連して、御自身の相互規定の相対論理についてお教えいただいたことに深く感謝の意を表します。論証支援システムの共同研究者であり、ソフトウェアの開発の面から御協力いただいている富士通研究所のハイ東善、佐藤かおる、土屋恭子さんにも感謝したい。本研究の一部は第五世代計算機プロジェクト研究の一環としてＩＣＯＴの依託で行ったものである。

参考文献

〔1〕南、沢村：論証支援システムの一構成、ソフトウェア科学会第3回大会、1986, pp. 273-276.

〔2〕伊藤、松山：推論ソフトウェアの構成［I］、電子情報通信学会誌 3/'87, pp. 268-274.

〔3〕J. A. Abrial : The mathematical construction of a program, Science of Computer Programming, Vol.4, 1984, pp. 45-86.

〔4〕J. A. Goguen and R. M. Burstall : Introducing institutions, LNCS, Vol. 164, 1983, pp. 221-256.

〔5〕S. K. Langer : A set of postulates for the logical structure of music, Monist 39, 1925, pp. 561-570.

〔6〕F. C. N. Pereira and D. H. D. Warren : Definite clause grammars for langauge analysis - A survey of the formalism and a comparison with augmented transition networks, Artificial Intelligence, Vol. 13, 1980, pp. 231-278.

〔7〕銀林 浩：証明の心理・納得の論理、数学セミナー、昭和61年、pp. 13-17.

〔8〕ダニエル・ソロー著、安藤他訳：証明の読み方・考え方、共立出版、昭和60年.

〔9〕南、沢村：落書帳からの証明 —計算機による論証支援のための証明方法論の一考察—、情報処理学会第35回全国大会、1987.

〔10〕I. Lakatos著、佐々木訳：数学的発見の論理、共立出版、昭和60年.

〔11〕北川敏男：統計情報論Ｉ、共立出版、昭和62年.

〔12〕南、沢村：理論関の関係構造、情報とサイバネティックス・シンポジウム、1987.

〔13〕J. Gait : An aspect of aesthetics in human-computer communications : Pretty windows, IEEE Tran. on Software Engineering, Vol. SE-11, No. 8, 1985, pp. 714-717.

〔14〕K. L. Norman, et al. : Cognitive layouts of windows and multiple screens for user interface, Int. J. Man-Machine Studies, Vol. 25, 1986, pp. 229-248.