Elements de specification pour l'interface utilisateur d'un systeme de calcul algebrique formel

Norbert Kajler

UNITÉ DE RECHERCHE
INRIA-SOPHIA ANTIPOLIS

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.:(1) 39 63 55 11

Rapports de Recherche

N° 1220

Programme 1
Programmation, Calcul Symbolique
et Intelligence Artificielle

ELEMENTS DE SPECIFICATION POUR L'INTERFACE UTILISATEUR D'UN SYSTEME DE CALCUL ALGEBRIQUE FORMEL

Norbert KAJLER

Mai 1990

RR_1220

Programme 1
Programmation, Calcul Symbolique et Intelligence artificielle

Eléments de spécification pour l'interface utilisateur d'un système de calcul algébrique formel

Specifying a Graphic User Interface for Computer Algebra Systems

Norbert Kajler
kajler@mirsa.inria.fr

INRIA Sophia-Antipolis
2004 route des Lucioles
06565 Valbonne Cedex, France

Résumé

Cet article décrit les problèmes liés à la réalisation d'une interface utilisateur ergonomique graphique paramétrable pour un système de calcul algébrique formel. Il propose une méthodologie basée sur l'utilisation des outils logiciels les plus adaptés, le but étant de générer des éléments essentiels de l'interface. Aprés avoir présenté les interfaces homme-machine des systèmes de calcul formel actuels, cet article expose les spécifications de l'interface souhaitée.

Abstract

This paper describes the specific problems of building an ergonomic parametrized user interface for a Computer Algebra System. A methodology based on the use of best suited software tools is presented. The aim is to generate essential parts of the user interface. After a presentation of the existing interfaces related to Computer Algebra System, this paper investigates possible specifications to provide the dreamed user interface.

Eléments de spécification pour l'interface utilisateur d'un système de calcul algébrique formel

Norbert Kajler
kajler@mirsa.inria.fr

INRIA Sophia-Antipolis
2004 route des Lucioles
06565 Valbonne Cedex, France

2 Avril 1990

Résumé

Cet article décrit les problèmes liés à la réalisation d'une interface utilisateur ergonomique graphique paramétrable pour un système de calcul algébrique formel. Il propose une méthodologie basée sur l'utilisation des outils logiciels les plus adaptés, le but étant de générer des éléments essentiels de l'interface. Après avoir présenté les interfaces homme-machine des systèmes de calcul formel actuels, cet article expose les spécifications de l'interface souhaitée.

1 Introduction

Une bonne interface homme-machine est désormais un élément essentiel pour un système algébrique de calcul formel (SCAF). Pour l'étudiant et l'utilisateur occasionnel, elle facilite grandement l'utilisation des principales fonctionalités; pour l'utilisateur averti, elle permet un accès exhaustif aux possibilités du système et facilite la programmation. Dans tous les cas, elle améliore grandement le confort visuel et offre de nombreuses possibilités d'édition. L'étude précise des besoins au niveau de l'interface homme-machine constitue l'objet de ce document et représente une étape importante, préalable à toute réalisation effective.

2 Description des interfaces homme-machine existantes

Du point de vue de leur interface homme-machine, les SCAF[1] se répartissent désormais en deux grandes familles.

[1] Des recopies d'écran (sur station de travail SUN3) correspondant aux systèmes MACSYMA, MATHEMATICA et MATHSCRIBE sont fournies en annexe.

La première fonctionne sur des terminaux traditionnels, les formules sont imprimées à l'aide de caractères alphanumériques, l'édition s'effectue en une dimension. Les systèmes MACSYMA [15], MAPLE [8] , REDUCE [16] et SCRATCHPAD II [2] [20,21] fonctionnent avec ce genre d'interface.

La seconde nécessite un terminal graphique haute définition doté d'un dispositif de pointage (une souris). Les formules sont dessinées en deux dimensions et l'édition peut s'effectuer à l'aide de la souris. Les SCAF disposant d'une telle interface sont encore peu nombreux. On peut toutefois citer[3] : GI/S [46], MATHEMATICA[4][45] et MATHSCRIBE [37,44]. Pour être précis, il faut distinguer MATHEMATICA, qui est un SCAF intégrant une interface graphique originale, de GI/S et MATHSCRIBE qui sont des interfaces utilisateurs évoluées construites au dessus d'un SCAF traditionnel (respectivement MACSYMA et REDUCE). Afin de disposer d'une référence pour la suite, nous allons présenter les principales caratéristiques de MATHSCRIBE.

MATHSCRIBE est une interface homme-machine graphique pour le système REDUCE. Elle permet à l'utilisateur d'ouvrir des fenêtres de travail en cliquant sur les boutons d'un tableau de bord toujours présent à l'écran. Chaque fenêtre permet de manipuler des formules en deux dimensions. On dispose des possibilités suivantes : édition syntaxique en temps réel, couper-coller, barres de défilement, "undoing", substitutions, abréviations locales et globales, accès à certaines fonctionalités par menu. MATHSCRIBE permet également de visualiser les drapeaux de REDUCE dans une fenêtre, de tracer des courbes en deux ou trois dimensions à l'écran et de générer du code EQN ou TEX à partir d'une formule. MATHSCRIBE exige de l'utilisateur un usage intensif de la souris. Il est cependant possible d'utiliser des formes alternatives au clavier lors de l'édition de formule. Les messages d'erreur de l'interface apparaissent dans une fenêtre séparée qui peut également servir d'interface textuelle traditionnelle avec REDUCE. Le paramétrage de l'interface est possible, mais se limite à la modification statique de certains éléments de l'aspect visuel de l'interface (choix des couleurs, des polices de caractères et des dimensions des fenêtres). L'utilisation d'un gestionnaire de fenêtre (tel que *uwm*) est indispensable pour déplacer ou agrandir une fenêtre.

3 Problématique de l'interface homme-machine

MATHSCRIBE est le premier exemple d'une interface utilisateur graphique évoluée pour un système de calcul formel. Il facilite effectivement l'accès au système REDUCE et apporte un confort visuel non négligeable.

Toutefois, de nouvelles exigences apparaissent chez les utilisateurs de ces systèmes. Elles visent à améliorer la convivialité des interfaces afin de pemettre la diffusion du calcul formel dans les milieux scolaires, scientifiques et industriels. Or, les besoins en interface diffèrent sensiblement selon le type d'utilisateur concerné. De plus, certains utilisateurs, spécialistes de la mécanique quantique ou des mathématiques appliquées aux modèles économiques, souhaitent disposer des opérateurs spécifiques à leur do-

[2] Pour SCRATCHPAD II, une interface graphique, destinée à la version IBM PC/RT, est en cours de développement.

[3] MACSYMA sur machine Lisp étant un cas à part (interface de qualité bâtie sur le système de fenêtrage original des machines Lisp).

[4] Pour MATHEMATICA, seuls certains systèmes (dont MacIntosh et NeXT) bénéficient d'une interface homme-machine évoluée.

maines, ou d'opérateurs nouveaux qui leur sont propres. Ajouter de tels opérateurs au calculateur formel est désormais possible. Par contre, les introduire au niveau de l'interface (sous la forme de composants graphiques originaux) neccessite de modifier le code de l'interface. C'est pourquoi, une future étape consistera à proposer des interfaces adaptées à une classe d'utilisateur et à un domaine spécifique. Parallèlement, de nouvelles techniques se développent dans le domaine du génie logiciel et les outils élaborés se multiplient. C'est notamment le cas pour les éditeurs syntaxiques graphiques et les boîtes à outils destinées au concepteur d'interfaces. C'est pourquoi, une méthodologie nouvelle, utilisant les outils logiciels les plus performants du moment pour générer automatiquement certains éléments de l'interface utilisateur, est une alternative interressante à l'écriture directe d'une interface pour un SCAF donné (comme c'est le cas pour MATHSCRIBE).

3.1 Outils existants

Lors de la réalisation d'une interface pour un SCAF, on doit résoudre deux grandes difficultés : l'édition en deux dimensions des formules mathématiques et la création puis l'assemblage en un tout cohérent d'un grand nombre de composants élémentaires (éditeurs, tableaux de bord, menus, traceurs de courbes, etc). Pour traiter ces deux types de problèmes, plusieurs outils sont déjà disponibles. D'autres, plus ambitieux sont en cours d'achèvement. Enfin, les plus élaborés d'entre eux sont à la base des projets de génération d'environnement de programmation dont le développement est une priorité de la recherche en informatique fondamentale pour les dix prochaines années. Dans la suite de ce paragraphe, chaque classe d'outil sera présentée et illustrée par quelques exemples.

Il existe désormais des éditeurs graphiques interactifs , tels que The Publisher ou Grif [34,33], spécialisés dans la manipulation des formules mathématiques. Il existe aussi, des générateurs d'éditeurs syntaxiques graphiques pouvant s'adapter à la manipulation d'expressions mathématiques. Un prototype en est Gigas [11] bâti au dessus du Cornell Synthesizer Generator [43,42]. Il permet de générer des éditeurs structurés graphiques en compilant une grammaire attribuée écrite dans le formalisme du Cornell Synthesizer Generator. Le dessin est le résultat d'un calcul incrémental d'attributs sémantiques fournis dans la grammaire. Le tout fonctionne dans l'environnement X Window System. Dans le même ordre d'idée, un système comme Centaur [22], destiné à la génération des environnements de programmation, devrait dans le futur intégrer un tel outil. Parmi les systèmes commercialisés, on peut aussi citer Graspin [4].

Aujourd'hui, la plupart des interfaces utilisateur sont écrites à l'aide de boîtes à outils. Une boîte à outils est un ensemble de fonctions de haut niveau, bâti au dessus des routines de bases fournies par un système de fenêtrage. Il existe désormais un très grand nombre de boîtes à outils. Certaines sont liées à un constructeur (par exemple : celle du MacIntosh [2], celle de la NeXT Machine, celle des stations SUN [40], ou celle des machines Lisp de Symbolics [38]), d'autres appartiennent au monde X Window System[5] (Athena Toolkit [26,41], OSF/Motif [29,31,30], Andrew Toolkit [1], InterView [25], CLUE [6][23] , ...) , d'autres enfin sont indépendantes de tout système de fenêtrage

[5]X [28,36] est en train de devenir le standard de l'industrie.

[6]CLUE (*Common Lisp User Interface Environment*) est un système de programmation orienté objet pour développper des interfaces utilisateurs en Common LISP [39] sous X Window System. Il est bâti au dessus de CLOS [3] (*Common Lisp Object System*) et de CLX [35] (*Common Lisp programmer's*

et peuvent être adaptées à des architectures variées (comme Aïda [18], construit au dessus du langage LELISP [7]). Ces boîtes à outils diffèrent également par leur langage de programmation (C, C++, Objective C, Smalltalk ou Lisp), leur conception générale (choix de la programmation orientée objet par exemple), et leur contenu (certaines proposent de nombreuses variantes des objets de bases comme les boutons et les menus, d'autres disposent de machines d'affichage spécialisées telles que des éditeurs d'arbres ou de vecteurs[7]).

Au delà des boîtes à outils, des outils de type éditeurs de tableaux de bord apparaissent [27]. Ils sont fréquemment rassemblés sous l'étiquette UIDS (*User Interface Design System*). Le précurseur en est SOS Interface [17] (en Lisp sur MacIntosh.). Depuis peu, un tel outil est commercialisé par la société Ilog (qui diffuse LELISP et Aïda) : Masaï [19]. Il s'agit d'un éditeur d'interfaces graphiques générant du code Aïda. Masaï possède lui-même une interface utilisateur graphique assez conviviale et permet de réaliser rapidement et interactivement des interfaces. De plus, le code généré est à la fois modifiable à la main par édition du programme Aïda et rééditable sous Masaï, ce qui facilite l'évolution des interfaces générées. Des outils équivalents existent aussi pour machine Lisp. Dans le monde X, de nombreux outils sont en cours de développement (par exemple : Egerie [5]) , sans oublier un outil déjà disponible pour NeXT : Interface Builder.

Il faut aussi signaler l'approche plus globale proposée par les spécialistes du dialogue homme-machine à travers le concept d'UIMS (*User Interface Management System*) [9,10,32]. Il s'agit cette fois de découper l'interface sous la forme de trois modules constitués en couches séparant fortement l'application de son interface. La communication entre une couche et la suivante s'effectuant de manière asynchrone par des appels de fonctions. Les trois modules ont des rôles précis : le premier est l' "*application interface*", son rôle est d'assurer les communications entre l'application et son interface. Dans un système comme Serpent [6], il consiste en une base de données partagée constituée des variables de l'application concernées par l'interface. Le second est le "*Dialogue Control*", il est responsable de toutes les interactions ne nécessitant pas de contrôle sémantique (comme les vérifications purement syntaxiques des données fournies par l'utilisateur), et impose le comportement de l'interface (le "*feel*"). Le troisième est le "*Presentation Manager*", il effectue les opérations d'affichage et gère l'aspect visuel de l'interface à l'écran (le "*look*"). Au concept d'UIMS correspondent des environnements de développement, intégrant des langages de spécification, qui permettent de générer des interfaces sur ce modèle. De tels outils complets et opérationnels sont rares et peu répandus, cependant de nombreux travaux sont en cours. A titre d'exemple, on mentionnera les UIMS Alberta [14], Serpent [6] et UIMX.

3.2 Méthodologie

Depuis la conception de MATHSCRIBE, les techniques ont donc largement progressé dans le domaine des interfaces homme-machine. C'est pourquoi, une nouvelle approche semble aujourd'hui souhaitable pour réaliser des interfaces homme-machine répondant aux impératifs d'évolutivité et de portabilité.

Le premier objectif consiste en la séparation du SCAF et de son interface, ce qui

interface to X).

[7]Par contre, aucune boîte à outils ne propose d'éditeur de formules mathématiques.

implique d'une part une répartition précise des fonctionalités entre les deux programmes et d'autre part la mise en œuvre d'un mécanisme de communication adéquate. Ainsi, un programme comme IRIS [24], conçu initialement pour le système MAPLE, propose un protocole de communication entre le noyau purement algébrique d'un SCAF et ses "périphériques" (interface utilisateur textuelle ou graphique, traceur de courbes et autres interfaces).

Ensuite, au lieu de concevoir l'interface homme-machine sous la forme d'une extension graphique d'un SCAF donné, on peut partir des besoins des utilisateurs pour réaliser une spécification précise de l'interface souhaitée. Cette spécification a alors l'avantage d'être largement indépendante d'un système particulier. La seconde étape, consiste alors à réaliser une interface répondant à ces spécifications pour le SCAF visé. Elle s'effectue en mettant en œuvre le maximum d'outils logiciels disponibles. Ces outils sont nombreux et doivent être choisis en fonction des besoins exprimés dans les spécifications. De plus, ils évoluent très rapidement. Séparer l'interface du SCAF permet donc de bénéficier des progrés réalisés par ces outils.

Les avantages apportés par l'utilisation d'outils générateurs sont bien connus : gain de productivité important lors du développement de l'interface, fiabilité du code produit, et surtout, possibilité de faire évoluer rapidement l'interface en effectuant les modifications en amont de l'outil générateur. En contrepartie, il faut mentionner des risques de perte de performances et de gaspillage d'espace mémoire, lorsque les outils choisis ne sont pas optimisés. Toutefois, l'évolution favorable des stations de travail (en termes de puissance et d'espace mémoire disponible) et la réalisation prochaine de versions industrielles de ces outils rendent prévisible le succés de cette démarche.

Enfin, il faut signaler que ces outils sont eux mêmes, pour la plupart, en cours de développpement. Or, les besoins relatifs aux interfaces des SCAF sont suffisament complexes pour intéresser les concepteurs de ces outils et participer à leur mise au point. Il est donc important de disposer rapidement de spécifications précises, couvrant l'ensemble des besoins des interfaces dans le domaine du calcul formel pour faire évoluer les outils à venir dans le sens des besoins ainsi exprimés.

Ainsi, le développement d'une interface homme-machine pour un SCAF pose une nouvelle problématique. Dans un premier temps, elle consiste à définir clairement les besoins pour l'interface utilisateur. C'est l'objet de la suite de cet article. Puis, dans une seconde étape, il s'agira d'étudier en détail les différents outils, collaborer éventuellement à leur mise au point, et effectuer des choix en vue de la réalisation effective de l'interface répondant aux spécifications énoncées.

4 Spécification des besoins pour l'interface utilisateur

4.1 Manipulation élémentaire de formules

L'utilisateur dispose de fenêtres d'édition dans lesquelles il frappe au clavier ou assemble à l'aide de la souris des formules destinées à être fournies au calculateur formel. L'édition s'effectue de manière interactive, en deux dimensions et en temps réel sur l'écran, tandis que la transmission a lieu sur requête explicite de l'utilisateur.

- **Visualisation des formules**
  Il s'agit de tirer parti de l'écran graphique haute définition pour visualiser les formules de la façon la plus agréable possible pour l'utilisateur, sans pénaliser ce

dernier par des contraintes supplémentaires ou des temps d'affichage trop longs. Un affichage de qualité implique en particulier l'utilisation de jeux de caractères de tailles différentes (caractères plus petits pour les indices par exemple), et le positionnement précis des composants graphiques. Ceci devra être pris totalement en charge par le programme, et ce, de manière tout à fait transparente pour l'utilisateur.
De même, lors de l'édition, certaines modifications entrainent des bouleversements dans le dessin de la formule. Chaque caractère frappé entraine donc automatiquement une révision à l'écran du dessin de la formule, sans autre intervention de l'utilisateur.

- **Curseur et sens de parcours d'une formule**
  Le curseur est un symbole désignant à l'écran l'emplacement de la formule susceptible de recevoir la prochaine insertion (il existe un curseur par fenêtre d'édition). Contrairement aux éditeurs traditionnels, le curseur n'effectue pas un simple balayage de gauche à droite et de haut en bas. Son déplacement dépend de la structure syntaxique de la formule et correspond au sens de parcours naturel d'une formule mathématique (notation infixée).

- **Edition à la souris**
  La souris est destinée aux opérations globales : elle facilite les manipulations portant sur des blocs dans une formule.
  Les fonctions d'édition suivantes doivent pouvoir être effectuées facilement et avec précision à la souris : déplacement du curseur à la position de la souris, marquage d'un bloc, effacement d'un bloc, insertion d'un bloc (couper-coller).

- **Notion de bloc marqué**
  La notion de bloc marqué est locale à l'interface (le SCAF n'en a pas connaissance). Pour l'utilisateur, il s'agit de désigner à l'aide de la souris une sous-formule dans une fenêtre d'édition mathématique. Une fois marqué, le bloc apparaît dans un rectangle de couleur et son contenu sert de paramètre implicite pour de nombreuses opérations de l'interface (destruction, copie, transmission, impression, ...).

- **Edition au clavier**
  Le clavier est l'outil le plus adapté pour la manipulation des unités lexicales élémentaires, il permet de déplacer le curseur à travers la formule et d'y effectuer les destructions et insertions de texte.
  Un maximum des fonctionalités élaborées de l'interface doit pouvoir être accessible à partir du clavier. Ainsi, les commandes d'édition, habituellement réalisées à la souris, pourront être effectuées à partir du clavier en frappant une combinaison particulière de touches (du genre (ESC) (X) ou (CTRL) (N)). Les liaisons "commande d'édition / équivalent clavier " seront modifiables, et pourront être visualisées à la demande de l'utilisateur.

- **"Undoing"**
  Au cours de l'édition d'une formule mathématique, il est souhaitable de pouvoir revenir pas à pas en arrière, afin de retrouver la formule telle qu'elle était à chaque étape de l'édition. Ce dispositif devra fonctionner de manière autonome à l'intérieur de chaque fenêtre d'édition et indépendament de l'historique (qui concerne les requêtes et les résultats du SCAF).

- **Transmission de la formule au programme de calcul formel**
  Elle s'effectue sur demande explicite de l'utilisateur, et porte sur l'expression éditée dans la fenêtre active.
  L'édition étant dirigée par la syntaxe, l'expression ne sera transmise au calculateur formel que si elle est syntaxiquement correcte. L'expression retournée par le calculateur pourra être affichée soit au dessous de la requête (comme c'est le cas dans tous les SCAF traditionnels); soit dans une seconde fenêtre dédiée aux réponses du système et couplée latéralement à la première.

4.2 Manipulation de formules complexes

Les formules manipulées par un SCAF sont souvent volumineuses (plusieurs milliers de caractères), les afficher telles quelles se révèle inutile et nuisible dans la plupart des cas. C'est pourquoi, l'interface devra particulièrement s'attacher à faciliter la manipulation de ces formules complexes, afin de les rendre plus aisément appréhendables à l'utilisateur.

- **Affichage des formules "longues"**
  Les expressions d'un encombrement latéral supérieur à la largeur de la fenêtre d'édition pourront être soit découpées selon des règles de césures inspirées de MACSYMA; soit affichées partiellement, la fenêtre étant alors couplée à un ascenseur horizontal (solution choisie par MATHSCRIBE).

- **Réductions et expansions**
  Il s'agit de substituer à un bloc, dans une formule, un objet graphique rappelant l'objet initial. L'icône créée comportera un texte généré automatiquement d'après la nature de l'expression substituée (Sum[100], Cos(...) ou Matrix[20,20] par exemple). L'expansion pourra être effectuée directement (par désignation à la souris) ou indirectement (à l'aide de la loupe ou du zoom).

- **Abréviations locales**
  Il s'agit de substituer, localement à une formule, toutes les occurences d'une expression par une abréviation. L'expression est celle correspondant à la zone marquée. L'abréviation peut être frappée au clavier dans une fenêtre de dialogue, ou prendre une valeur par défaut (LOCAL_i, si i est la i-ème abréviation locale demandée pour la formule courante).

- **Abréviations locales simples**
  Elles fonctionnent de la même manière que les abréviations locales, excepté que seule la zone marquée est substituée par l'abréviation. La valeur par défaut est : $\mathsf{LOCAL}_i^{\wp}$, le symbole $\wp$ (pour partiel) rappelant que la substitution ne porte pas sur toutes les occurences.

- **Abréviations globales**
  Le principe est le même que pour les abréviations locales, excepté que le couple abréviation/contenu est défini et disponible pour chaque fenêtre d'édition. La valeur par défaut est ABBREV_i, la numérotation des indices étant également globale pour l'application.
  Deux commandes, disponibles par menu, effectuent des abréviations globales : la première consiste à abréger la zone marquée (ce qui définit un nouveau couple

abréviation/contenu); la seconde consiste à réaliser à l'intérieur de la formule tous les remplacements de sous-expressions possibles en fonction des abréviations déjà définies.
L'ensemble des couples correspondant aux abréviations globales définies, peut être affiché dans une fenêtre en cliquant sur un bouton du tableau de bord.

- **Abréviations globales simples**
  Elles fonctionnent de la même manière que les abréviations globales, excepté que le sujet de la substitution est la zone marquée exclusivement. La valeur par défaut est : $\mathsf{ABBREV}_{\mathsf{i}}^{\mathsf{p}}$.

- **La loupe**
  La loupe permet, lorsqu'on la positionne sur une icône relative à une réduction ou à une abréviation, de faire apparaître dans une fenêtre le contenu correspondant. Cette fenêtre comportera le nom de l'abréviation concernée et une indication sur la fenêtre d'édition dont elle est issue. Elle pourra soit disparaître lorsque le bouton de la souris est relaché, soit être épinglée sur l'écran.

- **Le zoom**
  Il s'agit d'effectuer le passage d'une formule complexe vers une sous-formule en simulant l'effet d'un zoom. L'action du zoom est multiple : utilisation d'une police de caractères de plus grande taille, élimination des réductions réalisées, et remplacement de toutes les abréviations par leurs valeurs.

4.3 Manipulations globales

- **Existence d'un "plan de travail"**
  On souhaite disposer à tout moment d'une fenêtre intitulée "plan de travail", vide lors du lancement de la session, et destinée à recevoir des morceaux de formule, non obligatoirement corrects d'un point de vue syntaxique, que l'on pourra entreposer et utiliser librement.

- **Gestion des blocs précédemment marqués**
  La plupart des fonctions évoluées de l'interface portent sur l'expression contenue dans le bloc marqué. Il serait donc utile de conserver les dernières expressions marquées à la disposition de l'utilisateur en fournissant un composant graphique facilitant leur recherche et leur manipulation.
  On peut ainsi imaginer deux fenêtres acollées latéralement, présentes à tout moment sur l'écran. Celle de gauche comporterait cinq boutons ayant pour image les icônes (au sens de la réduction) correspondant aux cinq derniers blocs marqués; les boutons, accessibles à la souris, permettant de visualiser les expressions correspondantes dans la fenêtre de droite sous forme expansée. Cliquer dans cette seconde fenêtre permettrait de recopier dans une fenêtre d'édition ou dans le plan de travail, l'expression ainsi selectionnée.
  Ce dispositif étant commun à toutes les fenêtres d'édition , les icônes seront accompagnées d'une indication sur la fenêtre dont elles sont issues.
  De plus, un ascenseur, couplé à la fenêtre de gauche permettrait de faire défiler les blocs marqués plus anciens.
  L'avantage d'un tel mécanisme est double : faciliter le travail de l'utilisateur qui disposera ainsi de cinq blocs actualisés automatiquement; et améliorer la sécurité

lors des manipulations globales (un bloc effacé reste ainsi disponible un certain temps).

- **Utilisation et manipulation de l'historique**
  Le concept d'historique est sensiblement le même dans les systèmes de la génération MACSYMA / REDUCE et dans un système comme MATHSCRIBE, il offre à l'utilisateur la possibilité de récupérer la n-ième expression fournie au système.
  Pour généraliser et faciliter l'emploi de ce mécanisme, il semble souhaitable d'aller au delà du rappel de la dernière expression fournie au système et de la référence explicite d'une expression par son numéro d'ordre dans l'historique.
  On peut ainsi imaginer les améliorations suivantes : permettre la recherche dans l'historique par "*pattern-matching*"; offrir à l'utilisateur la possibilité d'insérer dans l'historique des commentaires utilisables comme clés de recherche.
  Par ailleurs, l'historique doit pouvoir être couplé à un objet graphique de type barre de défilement.

- **Existence d'ascenseurs**
  Chaque fenêtre mathématique dispose de deux ascenseurs : le premier est horizontal, et permet de parcourir une expression latéralement; le second est vertical, et permet de remonter dans le passé de la session.
  On souhaite laisser le choix de coupler l'ascenseur vertical avec les expressions fournies au système, les résultats rendus, ou les deux à la fois (comme c'est le cas dans MATHSCRIBE). Ceci est rendu nécessaire par le fait que les utilisateurs ont des habitudes et des besoins différents : lorsque les résultats sont longs (plusieurs lignes de résultat pour une requête d'une ligne est fréquent) la recherche d'une requête par un ascenseur classique se révèle fastidieux; inversement, la recherche d'un résultat de calcul est facilité si l'ascenseur est couplé avec les seuls résultats.

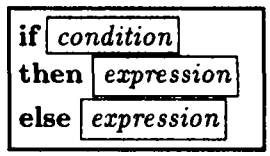
- **Gestion de commentaires**
  En dehors de l'historique, il peut être utile d'attacher des commentaires à des objets, notamment à l'intérieur du "plan de travail". Ces commentaires devront apparaître dans une police de caractères particulière. Ils pourront être crées, modifiés ou détruits librement par l'utilisateur. De plus, ils pourront également être momentanément cachés (un symbole graphique comme † rappelerait l'existence d'un commentaire relatif à une expression).
  Dans tous les cas, les commentaires (liés à une expression ou à une ligne de l'historique) sont gérés par l'interface. Ils ne sont pas transmis au calculateur formel et n'apparaîtront donc pas dans les lignes de réponses du SCAF.

4.4 Utilisation du langage de commande

Tout SCAF possède un langage de commande qui donne accès à l'ensemble des possibilités du système et permet l'écriture de programmes. L'interface homme-machine devra donc proposer des outils pour faciliter l'utilisation de ce langage de commande.

- **Edition syntaxique**
  Dans les fenêtres mathématiques, l'édition des expressions sera dirigée par la syntaxe du langage de commande (y compris les formules mathématiques).
  Pour l'interface, cela implique une connaissance précise de la syntaxe de la forme

manipulée et un niveau d'abstraction supérieur à celui d'un simple éditeur de caractères, le tout permettant l'application de traitements puissants au texte édité (tel que la vérification dynamique de la validité syntaxique).
Pour l'utilisateur, cela signifiera que l'éditeur tentera pour chaque unité lexicale reconnue, de faire correspondre le texte frappé à un modèle syntaxiquement correct. Ainsi, l'éditeur pourra refuser des unités lexicales non acceptables dans le contexte courant, ou les compléter par l'insertion de blocs vides que l'utilisateur devra remplir avant de demander la transmission de l'expression au calculateur formel. De plus, l'éditeur se servira de sa vision syntaxique pour anticiper la frappe de l'utilisateur. Par exemple, après la frappe au clavier du caractère (^), le curseur montera et attendra l'entrée d'un exposant. De même, une fois le mot clé if reconnu, l'éditeur génèrera automatiquement le bloc :

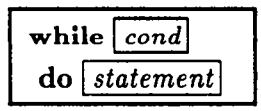


Dans un tel moule syntaxique, les mots clés (imprimés en gras) fournissent un cadre immuable pour l'insertion et la modification des unités lexicales manquantes.
Le rôle de l'éditeur incluera donc : l'assurance de la validité syntaxique des expressions, l'indentation automatique des programmes et l'anticipation de la frappe de l'utilisateur par le placement correct du curseur.

- **Génération de moules syntaxiques**
Des menus permettront de selectionner une unité syntaxique à compléter (moule syntaxique) et de la positionner à l'emplacement du curseur.
Voici sur des exemples, l'aspect de ces moules à l'écran :

pour une intégrale indéfinie

ou,



pour l'instruction **while**.

Ces menus offriront également une visualisation rapide de l'ensemble des constructions disponibles du langage de commande, et constituent ainsi une alternative séduisante à la lecture de la documentation.
Ces mêmes moules syntaxiques seront aussi générés lors de l'édition au clavier, lorsque l'éditeur identifiera le début d'une unité syntaxique.

- **Aide interactive à la programmation**
Une grande difficulté rencontrée par les programmeurs dans des systèmes comme MACSYMA ou SCRATCHPAD II est le très grand nombre de concepts disponibles (liste de modules impressionante, graphe des types compliqué, présence de milliers de fonctions, ...), il faut donc fournir à travers l'interface homme-machine des moyens de recherche rapides et ponctuels dans la documentation.
On peut ainsi envisager différents schémas : aide à la recherche d'un nom de variable ou de fonction (par *"pattern-matching"*), complétion des noms (à la manière d'Emacs), accès rapide à la documentation technique, etc.

- **Aide à la mise au point des programmes**
  L'interface devra faciliter l'utilisation des possibilités de mise au point disponibles au niveau du langage de commande (fonctions *debug* et *trace* du SCAF).

4.5 Rôle de la couleur

La couleur devra améliorer la compréhension des concepts de l'interface. Il est fondamental d'associer une coloration constante pour un objet ou une notion identique, et ceci dans l'ensemble des fenêtres de l'application. De plus, le nombre de couleurs devra être limité afin de permettre une lecture rapide, fondée sur le contraste des couleurs.

Par exemple, la couleur peut servir à faciliter les opérations d'édition courantes en différenciant le curseur, la zone sélectionnée, et la zone en cours de sélection par des couleurs particulières. De même, la coloration des substitutions et abréviations permet de les repérer instantanément, ce qui est très utile lorsque l'utilisateur a choisi une abréviation peu discernable autrement (les valeurs par défaut des abréviations se remarquent assez facilement).

L'usage de la couleur intervient également pour la distinction rapide des fenêtres sur l'écran. Ainsi, toutes les fenêtres de l'application se démarquent des autres par la couleur identique de leur fond, et l'utilisation d'une palette de couleurs homogène.

De plus, l'interface devant fonctionner correctement sur un écran monochrome, la couleur ne servira en aucun cas de véhicule principal de communication. Son rôle sera donc strictement limité à la mise en valeur d'un certain nombre de concepts disponibles par ailleurs.

4.6 Communication avec le système de calcul formel

Les informations concernant le système doivent être complètes, largement disponibles et rapides à appréhender. Les commandes qui lui sont destinées doivent être simples et pratiques pour l'utilisateur non expérimenté. Des menus proposant, de manière exhaustive, l'ensemble des possibilités liées à un concept donné seront disponibles chaque fois que ces possibilités pourront être connues à l'avance.

- **Tableau de bord**
  Un tableau de bord, toujours présent sur l'écran, permet de piloter l'ensemble de l'application.
  Il se présente sous la forme d'une fenêtre rassemblant des informations essentielles sur le système et les boutons des principales commandes.
  Les boutons comporteront, en plus du nom de l'action déclenchée, une indication graphique complémentaire sur la nature de l'effet associé (action immédiate, action à confirmer, déroulement d'un menu ou apparition d'une boîte de saisie).
  Dans le cas d'un menu ou d'une boîte de saisie, la valeur courante associée sera affichée à proximité du bouton sur le tableau de bord, et sera rappelée, ainsi que la valeur par défaut, à l'intérieur du menu ou dans la fenêtre de saisie.

- **Accès aux variables globales du système**
  Une fenêtre doit permettre de visualiser et de modifier l'ensemble des variables globales du système. Cette fenêtre prendra la forme d'un tableau, chaque ligne comportant les informations suivantes : nom de la variable, valeur courante

(éditable), valeur par défaut et indication sur l'ensemble des valeurs autorisées (par exemple : booléen, intervalle numérique, chaîne de caractères).

- **Accès à la documentation**
  La documentation du SCAF doit être consultable de façon permanente. La manipulation donnant accès a la documentation doit être évidente et immédiate pour l'utilisateur.
  Pour cela, on dispose d'un bouton "HELP" sur le tableau de bord. L'effet du bouton "HELP" consiste à dérouler un menu permettant soit d'accéder aux diffèrentes sections du manuel de référence du SCAF, soit d'obtentir une aide ponctuelle relative à une fonction ou variable globale désignée à la souris.
  Les composantes de l'interface devant également être auto-documentées, il sera de plus possible par l'intermédiaire du bouton "HELP" de désigner un objet graphique (tel qu'une abréviation ou une barre de défilement) pour obtenir les explications correspondantes.

- **Pilotage et suivi d'algorithmes**
  Le langage de commande du SCAF doit offrir au programmeur la possibilité de communiquer avec l'utilisateur lors de l'exécution du code.
  En effet, certains algorithmes complexes, tels que Knuth-Bendix ou le calcul de bases standard (Gröbner basis), tirent parti d'un échange d'informations entre le système et l'utilisateur : le programme peut ainsi indiquer où il en est et les résultats des calculs intermédiaires, ce qui peut intéresser l'utilisateur et l'amener à se substituer au programme lors de certains choix cruciaux.
  Dans d'autres cas plus simples, le programme peut juste souhaiter informer l'utilisateur du lancement d'une routine particulière, ou bien prévenir que la suite d'un calcul risque d'être longue.
  Au niveau de l'interface, cette technique mettra en jeu une fenêtre dédiée au déroulement de ce dialogue ou à l'affichage des informations relatives à l'exécution de l'algorithme.
  L'interaction entre le programme en cours d'exécution et l'utilisateur peut être conçue selon deux schémas distincts : soit comme un dialogue à l'initiative exclusive du calculateur (le programme informe l'utilisateur et lui fait prendre certaines décisions); soit comme un pilotage de l'algorithme par l'utilisateur (le programme informe régulièrement l'utilisateur par l'affichage de certaines données et l'utilisateur a de manière permanente la possibilité d'imposer ses choix au programme en appuyant sur des boutons à l'aide de la souris).
  De plus, les choix effectués par l'utilisateur durant les calculs seront mémorisés. Un symbole particulier accompagnera dans l'historique les requêtes ayant engendrés de tels choix. Ce symbole servira également de bouton poussoir pour visualiser les choix correspondant à une requête de l'historique.
  Au niveau du code de l'algorithme, le développeur devra intégrer au programme des instructions relativement simples permettant à l'interface de générer les fenêtres de dialogue. Ces instructions indiqueront le type de dialogue souhaité, les informations à afficher, les boutons à créer et les actions correspondantes.

- **Utilisation du traceur de courbes**
  Un bouton du tableau de bord doit faciliter l'utilisation du traceur de courbes intégré au SCAF. L'effet de ce bouton sera l'apparition d'une fenêtre de dialogue

couplée à la fenêtre destinée à afficher le tracé. La fenêtre de dialogue aidera l'utilisateur à désigner une équation et les différents paramètres nécessaires au programme traceur de courbes.

4.7 Communication avec l'extérieur

L'interface doit proposer un moyen simple et rapide pour transférer vers l'extérieur (autres applications, imprimante, etc) des données [8] manipulées par le système.

On pourrait ainsi utiliser deux menus pour choisir, d'une part, un format de sortie (TEX, PostScript, C, Fortran, Lisp, ...) , et d'autre part, un canal (fichier, cut buffer de Xwindow, imprimante, ...). La transmission du contenu du bloc marqué s'effectuerait alors en appuyant sur un bouton du tableau de bord. Une fois sélectionnés, le canal et le format de sortie seront affichés en permanence, à proximité du bouton commandant la transmission.

Au delà de ces possibilités, on peut imaginer un mécanisme permettant à l'utilisateur d'écrire et d'intégrer à l'interface une fonction générant du code dans un format de sortie particulier [9] à partir d'un format de référence, syntaxiquement simple et sémantiquement allégé (le format Lisp par exemple).

4.8 Aide à l'écriture de textes scientifiques

Il s'agit d'un cas particulier de communication entre le SCAF et une application de type éditeur de texte.

L'objectif est de répondre à trois souhaits exprimés par la plupart des utilisateurs de systèmes de calcul formel. Le premier est la simple intégration à un document en cours d'édition d'une formule obtenue à l'aide du SCAF. Le deuxième est l'insertion d'une figure dans le document (il peut s'agir d'une courbe tracé par le SCAF dans une fenêtre ou d'une banale recopie d'écran). Le troisième est la génération d'un résumé de session de calcul formel (au format TEX ou PostScript). Ce résumé pourra être obtenu en rassemblant une suite pertinente de couples requête/réponse selectionnés librement à la souris en remontant les historiques des fenêtres mathématiques.

Dans tout les cas, le principe sera d'alléger au maximum la tâche de l'utilisateur. Toutefois, il ne s'agira que d'une aide à la création de documents : dans le cas de TEX par exemple, l'utilisateur devra le plus souvent ajouter quelques commandes pour intégrer parfaitement la formule au document (choix d'une taille différente ou césure à modifier).

4.9 Relation avec le gestionnaire de fenêtres

Sous Xwindow, les actions suivantes : *move* (déplacement d'une fenêtre), *resize* (agrandissement ou rétrecissement) et *iconify*, (substitution d'une icône à une fenêtre), sont habituellement effectuées d'une façon uniforme par l'intermédiaire du gestionnaire de fenêtres. Cependant, c'est au programmeur d'écrire, si nécessaire, les méthodes non

[8] Dans la pratique, il s'agira le plus souvent de formules mathématiques. Toutefois, d'autres entités du système ou de l'interface seront concernées : programmes décrivant un algorithme, extraits de session, recopie de fenêtre ou d'écran, ...

[9] Un utilisateur peut souhaiter, par exemple, faire communiquer le SCAF avec un tableur ou une base de données.

standards pour effectuer chacune de ces actions sur les différentes variétés de fenêtre composant l'interface (ce sera notamment le cas pour les fenêtres comportant des éditeurs de formules). Le resultat doit être transparent à l'utilisateur du système, désireux de retrouver les sensations liées à son gestionnaire de fenêtres lors des manipulations des différentes composantes de l'interface.

4.10 Paramétrage de l'interface

Au delà des possibilités offertes par le gestionnaire de fenêtres, l'utilisateur doit avoir la possibilité (par choix d'options ou édition d'un fichier de paramètres) de modifier statiquement un large sous-ensemble des caractéristiques visuelles de l'interface : polices de caractères, couleurs, taille par défaut des fenêtres, etc. De plus, certains choix devraient pouvoir être effectués dynamiquement (pendant l'exécution du programme) : utilisation d'une ou deux fenêtres lors du dialogue avec le calculateur formel, critères de césures des formules trop longues, etc.

5 Conclusion

L'absence d'interfaces homme-machine évoluées à certainement été un obstacle sérieux à la diffusion des systèmes de calcul formel dans la communauté scientifique. Aujourd'hui, les matériels et les outils logiciels disponibles permettent la réalisation d'interfaces graphiques sophistiquées. La principale difficulté à surmonter pour la réalisation d'une telle interface dans le domaine du calcul formel est de satisfaire une exigence de qualité : respecter la typographie et les habitudes de travail tout en fournissant un ensemble d'outils conviviaux et performants.

Ce document avait pour objectif de fournir une première vision de ce que pourrait être l'interface d'un système de calcul formel dans un proche avenir. Il ne constitue nullement la spécification de la meilleure interface imaginable pour un tel système, mais seulement une image des besoins exprimés par les utilisateurs. Dans le cadre du projet Safir de l'Inria, il servira de modèle pour la spécification de l'interface homme-machine du système de calcul formel SISYPHE[10] [12].

Ce document a présenté aussi une méthodologie de développement basée sur l'utilisation d'outils logiciels de haut niveau. De ce point de vue, la prochaine étape de notre travail fera apparaître des problèmes nouveaux. En effet, après les avoir évalués, il s'agira de choisir et de faire communiquer ces outils pour parvenir à un ensemble cohérent répondant aux spécifications initiales.

6 Remerciements

L'auteur tient à remercier Marc Gaëtano, André Galligo, José Grimm et les équipes Koala et Side de l'Inria, pour l'aide précieuse qu'ils lui ont apporté depuis le début de ses travaux, ainsi que les nombreux utilisateurs de systèmes de calcul formel qui ont contribué par leurs remarques à l'élaboration des spécifications présentées dans cet article.

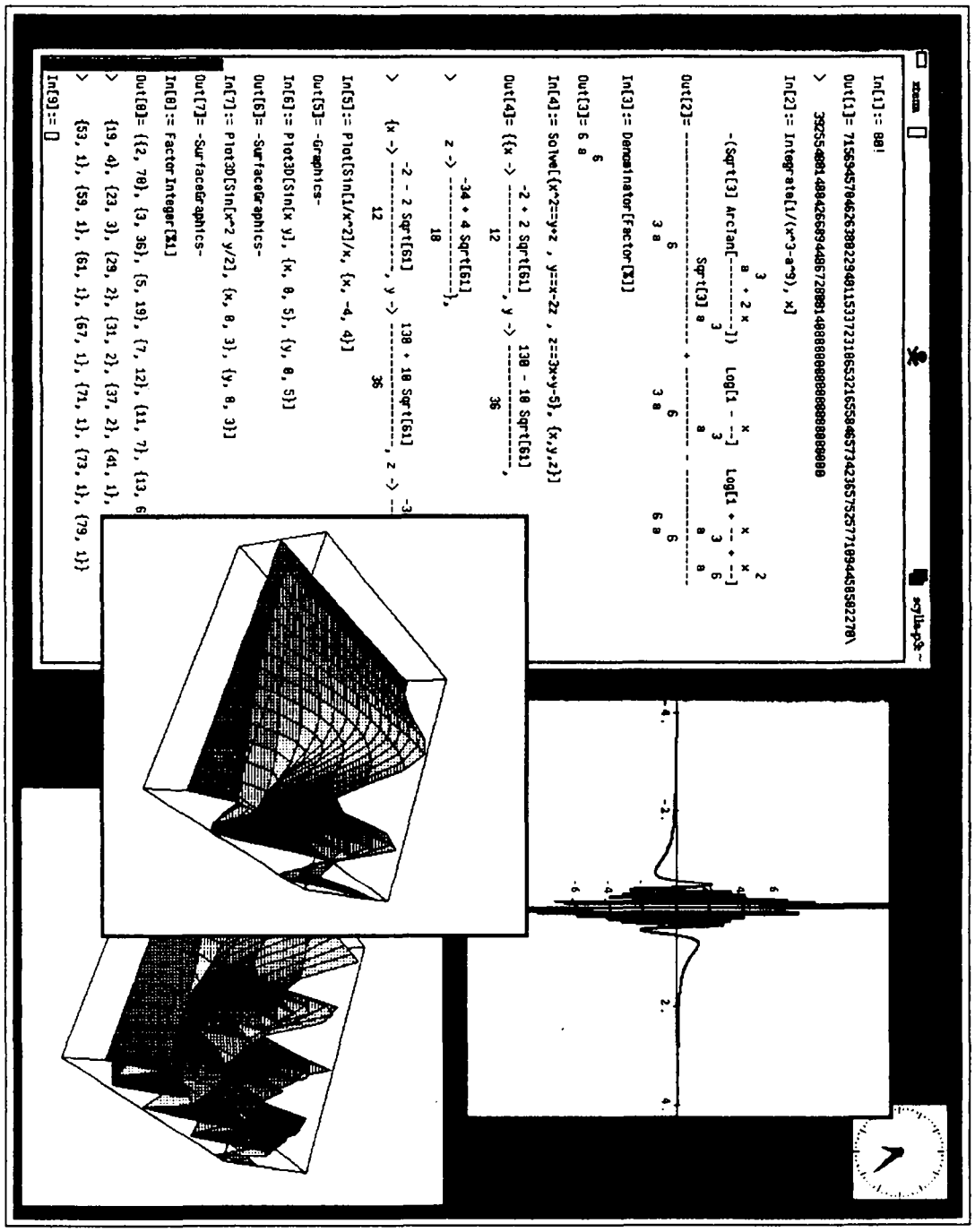
[10] En cours de développement à l'Inria et à l'Université de Nice.

7 Annexe

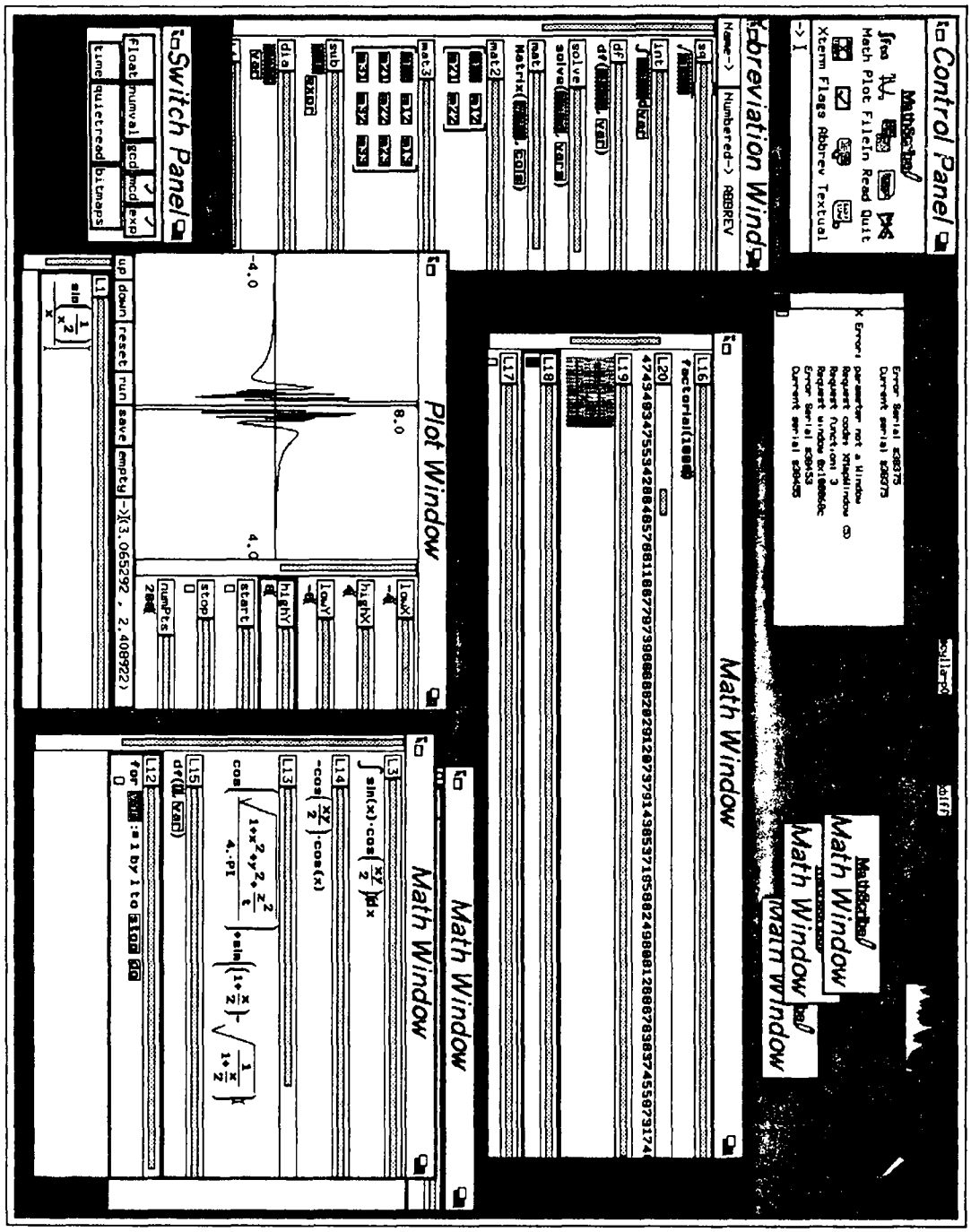
7.1 Exemple de session avec Macsyma

```
(c1) integrate(1/(sqrt(a*x^2+b*x)),x);
Is  a  positive or negative?
neg;
Is  b  zero or nonzero?
nonzero;
Time= 250 msec.

                                        2 a x + b
                                   asin(---------)
                                         abs(b)
(d1)                             - ---------------
                                      sqrt(- a)
Displaytime= 16 msec.
(c2) diff(%,x);
Time= 16 msec.

                                        2 a
(d2)                 - ------------------------------------
                                                          2
                                                 (2 a x + b)
                       sqrt(- a) abs(b) sqrt(1 - -----------)
                                                       2
                                                      b
Displaytime= 16 msec.
(c3) radcan(%);
Totaltime= 1533 msec.  GCtime= 566 msec.

                                        1
(d3)                          --------------------
                              sqrt(x) sqrt(a x + b)
Displaytime= 0 msec.
(c4)
```

7.2 Exemple de session avec Mathematica



7.3 Exemple de session avec MathScribe



Références

[1] J. Palay Andrew. The Andrew Toolkit; An Overview. In *proceedings of the 1988 Winter USENIX Technical Conference*, pages 9–21, Dallas, Texas, February 1988.

[2] Apple Computer, Inc. *Inside Macintosh.* Addison Wesley, New York, 1985.

[3] Daniel G. Bobrow and al. *The Common Lisp Object System Specification (X3J13-88-002).* American National Standarts Institute, 1988.

[4] M. Bologna, M. Chesi, S. Mannucci, I. Montanelli, P. Torrigiani, and N. Zuffi. The Kernel System of Graspin. In *CASE 87*, Cambridge, MA, 1987.

[5] Vincent Bouthors and Vania Joloboff. *Editeur d'interface EGERIE.* Rapport interne du centre de recherche BULL, Septembre 1989.

[6] Carnegie Mellon University Software Engineering Institute. *SERPENT Overview.* Technical Report CMU/SEI-89-UG-2, Carnegie Mellon University, August 1989.

[7] Jerome Chailloux, Mathieu Devin, and Jean-Marie Hullot. LeLisp, a Portable and Efficient Lisp System. In *Proceedings ACM Symposium on Lisp and Functional programming*, pages 113–122, 1984.

[8] Bruce W. Char, Keith O. Geddes, Gaston H. Gonnet, and Stephen M. Watt. *Maple User's Guide.* Watcom Publications edition, 1985.

[9] Joelle Coutaz. A Layout Abstraction for User System Design. *ACM SIGCHI*, 18–24, January 1985.

[10] Joelle Coutaz. Abstractions for User Interface Design. *IEEE Computer*, 18(9):21–34, September 1985.

[11] Paul Franchi-Zanettacci, Bruno Chabrier, and Vincent Lextrait. GIGAS : a Graphical Interface Generator for Attribute Specifications. In *Actes du colloque "Le Génie logiciel et ses Application"*, Toulouse, Décembre 1988.

[12] André Galligo, José. Grimm, and Loic Pottier. The design of SISYPHE : a system for doing symbolic and algebraic computations. In *Proceedings of DISCO'90*, April 1989.

[13] Danny Goodman. *The Complete HYPERCARD Handbook.* The MacIntosh performance library, 1987.

[14] Mark Green. The University of Alberta User Interface Management System. In ACM, editor, *Computer Graphics*, pages 205–213, SIGGRAPH'85, San Fransisco, July 1985.

[15] The Mathlab Group. *Macsyma Reference Manual, version 10.* M.I.T. edition, 1983.

[16] Antony C. Hearn. *Reduce User's Manual.* Rand publication CP78 edition, April 1985.

[17] Jean Marie Hullot. SOS Interface: Un générateur d'interfaces Homme-Machine. In *Actes des journées AFCET sur les Langages Orientés Objets*, pages 69–78, Bulletin BIGRE+GLOBULE, Janvier 1986.

[18] Sté ILOG. *AIDA Environnement de développement d'applications.* Paris, 1988.

[19] Sté ILOG. *MASAI L'outil de développement interactif d'interfaces graphiques.* Paris, 1989.

[20] Richard D. Jenks. A primer: 11 keys to new scratchpad. In *Proceedings of EUROSAM'84 in Cambridge, England,* pages 123–147, LNCS vol. 174, Springer-Verlag, july 1984.

[21] Richard D. Jenks, Robert S. Sutor, and Stephen M. Watt. *Scratchpad II: An Abstract Datatype System for Mathematical Computation.* Research Report RC 12327, IBM, November 1986.

[22] Gilles Kahn and al. CENTAUR: the system. In E. Brinksma, G. Scollo, C. Vissers, editor, *Proc. of 9th IFIP WG6.1,* Intern. Symp. on Protocol Specification, Testing and Verification, 1989.

[23] Kerry Kimbrough and Oren LaMott. *Common Lisp User Interface Environment.* Texas instrument, Inc, February 1988.

[24] B. L. Leong. Iris: Design of a User Interface Program for Symbolic Algebra. In *Proceedings of the 1986 ACM-SIGSAM Symposium on Symbolic and algebraic Computation,* pages 1–6, July 1986.

[25] Mark. A. Linton, Paul. R. Calder, and John. M. Vlissides. *InterViews: A C++ graphical interface toolkit.* Technical Report CSL-TR-88-358, Stanford University.

[26] Joel McCormack, Paul Asente, and Ralph R. Swick. *X Toolkit Intrinsics-C language Interface.* Digital Equipment Corporation, MIT edition, 1988.

[27] Brad A. Myers. *Tools for Creating User Interfaces: An Introduction and Survey.* Technical Report CMU-CS-88-107, Carnegie Mellon University, January 1988.

[28] Adrian Nye. *Programming reference manual for the X Window System.* O'Reilly and Associates, Inc, 1987.

[29] OSF/Motif. *OSF/Motif, Programmer's Guide, revision 1.0.* Eleven Cambridge Center, Cambridge, MA 02142, Open Software Foundation edition, 1989.

[30] OSF/Motif. *OSF/Motif, Programmer's Guide, Style Guide, revision 1.0.* Eleven Cambridge Center, Cambridge, MA 02142, Open Software Foundation edition, 1989.

[31] OSF/Motif. *OSF/Motif, Programmer's Reference Manual, revision 1.0.* Eleven Cambridge Center, Cambridge, MA 02142, Open Software Foundation edition, 1989.

[32] Gunther R. Pfaff. User Interface Management Systems. In *Proceedings of the workshop on user interface,* Springer, Seeheim, FRG, November 1983.

[33] Vincent Quint. Two dimensional editing. In IEEE Computer Society, editor, *Actes de IEEE First International Conference on Office Automation,* New Orleans, Louisiana, 1984.

[34] Vincent Quint. Un système interactif pour la production de documents mathématiques. *Technique et Science Informatiques*, 2(3):179–190, 1983.

[35] Robert W. Scheifler and al. *CLX Interface Specification.* September 1987.

[36] Robert W. Scheifler and Jim Gettys. The X Window System. *ACM Transactions on Graphics*, 5(2):79–109, April 1986.

[37] Carolyn J. Smith and Soiffer Neil M. MathScribe: A User Interface for Computer Algebra Systems. In ACM, editor, *Conference Proceedings of Symsac 86*, pages 7–12, July 1986.

[38] Richard Stallman, Daniel Weinreb, and David Moon. *Lisp Machine Window System Manual.* MIT, August 1983.

[39] Guy Steele. *Common LISP: The Language.* Digital Press, Burlington, MA., 1984.

[40] Sun Microsystems, Inc. *SunWindows Programmers' Guide.* 2550 Garcia Avenue, Mountain View, CA 94043, January 1984.

[41] Ralph R. Swick and Terry Weissman. *X Toolkit Athena Widgets.* MIT edition, 1988.

[42] Tim Teitelbaum and Thomas Reps. *The Syntesizer Generator Reference Manual.* Cornell University, NY 14853, August 1985.

[43] Tim Teitelbaum and Thomas Reps. The Cornell Program Synthesizer: A Syntax Directed Programming Environment. *Communications of the ACM*, 24(9):563–573, September 1981.

[44] Tektronix, Inc. *MathScribe User's Manual Version 1.0.* edition, 1988.

[45] Stephen Wolfram. *Mathematica, A System for Doing Mathematics by Computer.* Addison-Wesley, 1988.

[46] Douglas A. Young and Paul S. Wang. GI/S: A Graphical User Interface For Symbolic Computation Systems. *J. Symbolic Computation*, (4):365–380, 1987.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique

ISSN 0249-6399