BULLETIN D'INFORMATIQUE APPROFONDIE ET APPLICATIONS

N° 44 JUIN 1996

SCIENCES DE L'EDUCATION ET DE L'INFORMATION

REDACTION

Université de Provence
Equipe Hermès. Case 33
3, place Victor Hugo
F - 13331 Marseille Cedex 3
Téléphone: 04 91 10 62 30
Télécopie : 04 91 50 91 10

DEPOSITAIRE

Université de Provence
Bibliothèque Universitaire
3, place Victor Hugo
F - 13331 Marseille Cedex 3
Téléphone: 04 91 62 44 16
Télécopie : 04 91 95 75 57

Publication gratuite trimestrielle de l'Université

Edition 1997

ISSN 0291 - 5413

BULLETIN D'INFORMATIQUE APPROFONDIE ET APPLICATIONS

N° 44 JUIN 1996

SCIENCES DE L'EDUCATION ET DE L'INFORMATION

REDACTION

Université de Provence
Equipe Hermès. Case 33
3, place Victor Hugo
F - 13331 Marseille Cedex 3
Téléphone: 04 91 10 62 30
Télécopie : 04 91 50 91 10

DEPOSITAIRE

Université de Provence
Bibliothèque Universitaire
3, place Victor Hugo
F - 13331 Marseille Cedex 3
Téléphone: 04 91 62 44 16
Télécopie : 04 91 95 75 57

Publication gratuite trimestrielle de l'Université

Edition 1997

ISSN 0291 - 5413

EDITORIAL,

LE JEU ET L'INFORMATIQUE.

Il devrait être rigoureusement interdit d'utiliser l'informatique pour autre chose que pour des jeux, et éventuellement, pour de l'enseignement, à condition, dans ce cas-là de prendre beaucoup de précautions et de préciser nettement le cadre. L'informatique est une branche de la connaissance bien trop dangereuse pour la mettre entre toutes les mains. Et encore moins entre les mains des "cerveaux" officiels, si j'ose dire. Nos sociétés sont encore bien trop infantiles pour pouvoir jouer librement avec de telles armes. Ceux qui prétendent diriger, et qui devraient disposer d'une largeur de vue exceptionnelle, n'ont malheureusement à leur disposition que des lunettes à œillères qui ne leur permettent pas de voir plus loin que leur prochaine réélection et encore ...

Et puis il y a ceux qui normalement sont connus pour détenir le savoir, les diplômés des vraies grandes écoles et enseignant dans les vraies grandes écoles. Ne pas confondre avec ceux qui exercent leurs prestations dans les petites universités de province réservées à la formation des chômeurs de haut niveau. Ces beaux esprits ont appris à raisonner en toute abstraction, aussi leurs théories sont elles parfaitement irréfutables. Sans aucun rapport avec la réalité, mais irréfutables. On laisse volontiers les seconds et troisièmes couteaux se dépatouiller avec les problèmes de la vie courante qui sont trop souvent à l'image des cages d'escalier des HLM de banlieue.

Priorité à "l'économie". Une certaine économie. Selon "Saint Maastricht" les exemples de grandes réalisations, ayant conduit droit à des désastres, ne sont pas rares. Nobel, promouvant sa dynamite, prétendait qu'elle représentait l'assurance de la fin de toute guerre. Assertion très drôle, considérée rétrospectivement.

Même chose pour l'aviation, formidable progrès de l'homme, capable de modifier profondément sa nature qui le colle inexorablement au sol. Et devenu rapidement, comme tous les autres progrès techniques, un formidable instrument de mort. Au mieux, et je parle de l'aviation civile, un formidable instrument de gaspillage de place, de carburant, de pollution par le bruit et les gaz, une fantastique machine à répandre les virus les plus dangereux à travers la planète, grâce au mélange intempestif des populations. Comme par hasard les vecteurs sont essentiellement les grands V.R.P. de la planète, les répartiteurs de "l'économie".

J'ose à peine citer l'énergie nucléaire avec ses Three Miles Island, ses Tchernobyl, ses sous-marins russes pourrissants à Mourmansk, ses marchands ambulants qui bradent le plutonium ex-soviétique dans toutes les rues de la Terre. Sans parler des centrales arrêtées dont on ne sait plus que faire tant elles restent dangereuses pour des siècles. Et pour des siècles encore resteront dangereux les stocks de déchets atomiques qui ne cessent de s'accroître et qu'on essaye de caser chez les plus démunis, les naïfs qui croient encore à la grandeur de la science et à ce que leur racontent les experts. Les mêmes experts qui disaient qu'à Tchernobyl il n'y avait eu qu'une seule victime, et encore parce qu'elle s'était brûlé le doigt avec un fer à souder pendant qu'elle réparait la centrale.

Eh bien, toute cette expérience demeure vaine. Comme dans la vallée du Rhône. Déjà l'hiver y avait provoqué une catastrophe lors d'une chute de neige particulièrement importante, et qui avait bloqué l'autoroute. Cela n'a guère servi de leçon, puisque pour achever 1996 et commencer 97, le temps a remis ça. Ce coup-ci non seulement l'autoroute, mais aussi la voie ferrée a été bloquée. Les caténaires étaient gelés, on se croirait dans un film de Buster Keaton.

Tout se passe comme si, lorsqu'il s'agit de lancer un grand projet, quelque part dans les sphères décidantes on entendait ce dialogue:
« ... si tout marche comme prévu, combien de morts ? »
«- normalement pas plus de 100 000 après la troisième année, Monsieur le Président .»
«- Alors fonçons ! »
Et on fonce. Un récent reportage a montré comment l'atoll de Bikini a servi de cobaye, population comprise, à des essais nucléaires mal contrôlés. Mais les américains ne sont pas les seuls capables d'agir de la sorte. Même ce pauvre Charpak vient apporter sa caution naïve (naïve ?) à un phénomène économique plus que douteux.

Mais alors, que vient faire l'informatique là au milieu. Simplement l'informatique dans son utilisation anarchique joue le rôle apparent d'un amplificateur de rendement. Lorsqu'un ordinateur s'implante quelque part, le chômage s'enrichit. Nos sociétés, visant à l'efficacité économique exclusive, ont irrémédiablement détérioré la qualité du travail, de sorte que le nombre d'emplois pour lesquels l'ordinateur remplace avantageusement l'homme, ne peut que s'accroître vertigineusement. Phénomène doublé par la délocalisation. Et triplé par la vente de ce qui reste encore à vendre comme industrie un tant soit peu valable. Comme c'est le cas pour Thomson à Daewoo, qui s'empressera de virer tout le personnel pour le remplacer par de parfaits esclaves. Et la qualité ? me direz-vous. Allons, allons vous répondrai-je, parlons de choses sérieuses.

Et pendant ce temps-là, les populations déshéritées, qui ont parfaitement compris qu'elles ne faisaient pas le poids dans un conflit économique, se retournent vers une solution désespérée, comme à Toulon, Orange, Vitrolles. Dans les années trente déjà, les populations acculées à l'extrême, dans cette Allemagne à la complète dérive économique, ont basculé dans un système de destruction total. Cette expérience n'a, elle non plus, pas servi à grand chose, et nous ignorons si la direction que nous prenons et qui lui ressemble, est irréversible. Cela n'a pas l'air de gêner nos grands responsables qui ne semblent voir là que de la simple concurrence pour qui se situera le plus à droite.

Peut-être, après tout, est-il nécessaire que l'entropie reprenne ses droits de temps en temps, et qu'il faille en passer par une bonne démolition de la société pour repartir sur de meilleures bases.

De meilleures bases ?

Edmond Bianco

COMPUTER ASSISTED PROGRAMMING FOR SYSTOLIC SYSTEMS

S.GHANEMI D.J. EVANS

Keywords :
Parallel Programming, Systolic Systems, Transputer, OCCAM Language, Graphic Syntactic Oriented Editor, Process Oriented Simulation, Performance Evaluation.

ABSTRACT :

Systolic architectures have proven to be cost-effective and high performance solutions for a large variety of problems often occurring in, for example, signal and image processing. Usually, the development of a Systolic solution for a given problem takes place into three major steps, the specification definition, design and hard-implementation. The design witch consists of an ad-hoc choice from a family of possible solutions, car, also be devided into three sub-steps, the choice of an adequate architecture, its soft-description using OCCAM language and its validation and evaluation on a transputer system. In this paper, we present a software tool, Computer Assisted Programming for Systolic Systems which is a substantial part of an OCCAM Programming Environment. It combines the advantages of two integrated tools: A Graphical Editor and a Systolic Editor. At the current time, other parts of the Programming Environnement are at the development stage.

1. INTRODUCTION:

Recent developments in circuit technology has led to the Very Large Scale Integration (VLSI) which enormously increased the number of electronic devices in a single chip, the size of a postale stamp. It is forseen that up to 1,000,000 components per chip are likely in the near future [KUNG 1988]. Until the advent of VLSI technology, the development of parallel computers with a large number of processors was limited by the unafordable manufacture costs. Kung H.T was the first to realise that the VLSI industry together with the automata theory could be the key success to building high-performance parallel computers at a very low cost [KUNG 82].

However, the growth in VLSI complexity motivated the development of CAD tools. Such approaches are extremely imparative due to the considerable large number of gates per chip in current technology. While these CAD software take a an algorithmic VLSI solution down to the actual hardware implementation, the designer still needs some assistance at the design phase.

In this paper, a Computer Assisted Programming tool for the Soft-Systolic Systems is presentated. It is part an OCCAM Programming Environment which is currently under development.

2. SIMULATION OF SYSTOLIC SYSTEMS:

We use the fact that OCCAM programs can be divorced from transputer configurations by simply using the language as a simulation tool for testing and evaluating several trade-offs in different solutions. Figure 1 shows the general structure of the programs where branching indicates parallel execution. The construction of programs follows ideas developed by G. M Megson [MEGSON 1984]. Consequently OCCAM programs simulate the formel proofs by replacing I/O description by actual results.

Although, the simulation does not garantee correctness, it is nevertheless a less consuming approach which does not result in unsolvable equations. Furthermore, a working program retains the possibility of actual transputer implementation and so solves two problems in one attempt.

The techniques described in the above cited paper have been used successfully to implement systolic designs in OCCAM, [GHANEMI 1987] and [MEGSON 1987].

3. PRESENTATION OF GEPASS:

Systolic designing which is an ad-hoc process heavily relies on common sense and personel experience. Consequently, in order to derive the most efficient and economical solution it is necessary to study and evaluate several trade-offs in different systolic architectures that are related to the same problem. This rather unavoidable stage which consumes much of the design costs can only be solved by mathematical considerations. However, this latter approach usually leads to unsolvable equations [MEGSON 1984]. Therefore, the solution should be found in the former approach if the involved design costs can be considerably reduced.

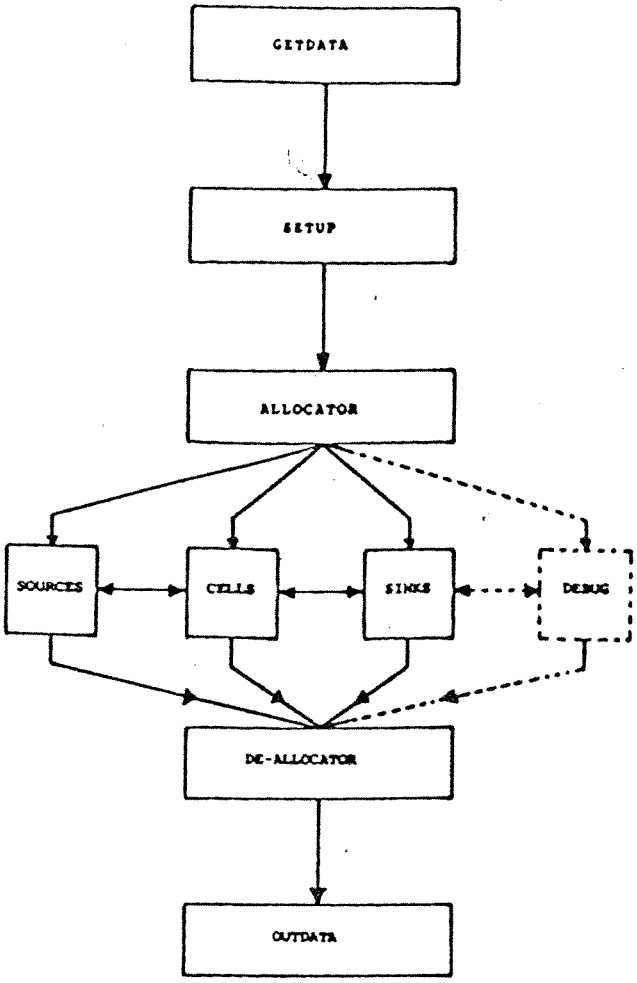


Fig. 1: The general structure of an OCCAM program
to simulate a given systolic array.

A quick look at the different solutions for a given problem reveals that while retaining the same structure of the architecture (i.e linear, array or hexagonal), it is only the way in which data and control are routed through the architecture that changes (i.e cell interconnection and data communication). This means that at any given time only a fraction of the implemented program is modified. In a traditional programming environment the entire program needs to be rewritten and recompiled several times before getting the final version. Consequently, if we were to follow the same approach we lose more time than we can save. Therefore, in order to gain more time, it is only necessary to check the syntactic and semantic validation of the added instructions within the program. This has the advantage of considerably reducing the lengthy and repeatedly process of editing, compiling and executing. The approach to use is based on the concept of Syntactic Oriented Editing.

A second improvement which might be of great help to the systolic designer is the use of some dedicated software packages as design assisting tools. Especially, the step of manually coding a particular systolic solution into a simulating OCCAM program can be done automatically. Thus, leaving the designer to put more effort into the process of deriving and suggesting alternative solutions.

In this respect, GEPASS (stands for Graphical Editor & Programming Assistance for Systolic Systems), which is a Computer Assisted Programming tool for Soft-Systolic Systems was developed [GHANEMI 1990a]. It has the advantage of combining the merits of both the Graphical and Syntactic Editors into one integrated tool. The former which is a natural way of system interfacing is used in order to build graphically systolic architectures. The latter is needed for eventually expanding internal details of some systolic entities, for example, the definition of the internal operation of a cell.

3.1 GRAPHICAL EDITOR:

The Graphical Editor, as its name indicates, is used to provide an easy and friendly way of drawing the structure of the proposed systolic solution. It is developed around the Object concept (Not to be confused with the Object Oriented concept). An object which can be either simple or complex is a graphical representation of a systolic entity such as a cell, canal or port. In order to efficiently manipulate these objects several functions were defined. Examples of such functions include creation and deletion of objects, movement of objects around the structure...etc. A more comprehensive list of all functions with examples can be found in [GHANEMI 1990b]. Fig 2 which is one of the menu of GEPASS shows some graphical functions.

Because of the simplicity of the entities in systolic systems only a sufficient and limited number of basic objects are defined. These are geometrical figures such as squares, circles and hexagons to represent different types of cells, lines or segments to represent channels and dots to represent ports. From these simple objects it is possible to form any complex object. For example a cell with two ports is represented by a square and two dots each on one side of the square. Similarily, complex objects can be combined to create more complex objects.

One particular characteristic of graphically designing systolic systems is the problem of the size (i.e the number of cells) which might increase considerably so that the screen would seem insufficient to display the entire structure. Therefore, two additionnal functions are essential: the ZOOM IN and ZOOM OUT functions. The first increases gradually the size of every objects so that it becomes possible to see clearly any relevant details in the architecture. The second function which is the inverse of the first one, allows the designer to see the entire structure while omitting some details.

During the editing phase, GEPASS generates interactively the simplified syntactic tree of the equivalent OCCAM program. Generating the tree instead of directly generating the OCCAM program has many advantages. For example, it is easy to update the tree without a complete rewrite of the entire OCCAM program every time changes are made. However, at the request of the designer, the associated OCCAM program can be easily derived from the tree after unparsing it.

Fig. 2a: Creation of the object SINK by pressing the F4 key.

Fig. 2b: Linking together the objects SOURCE, SINK and a set of objects CELLS by pressing the F5 key.

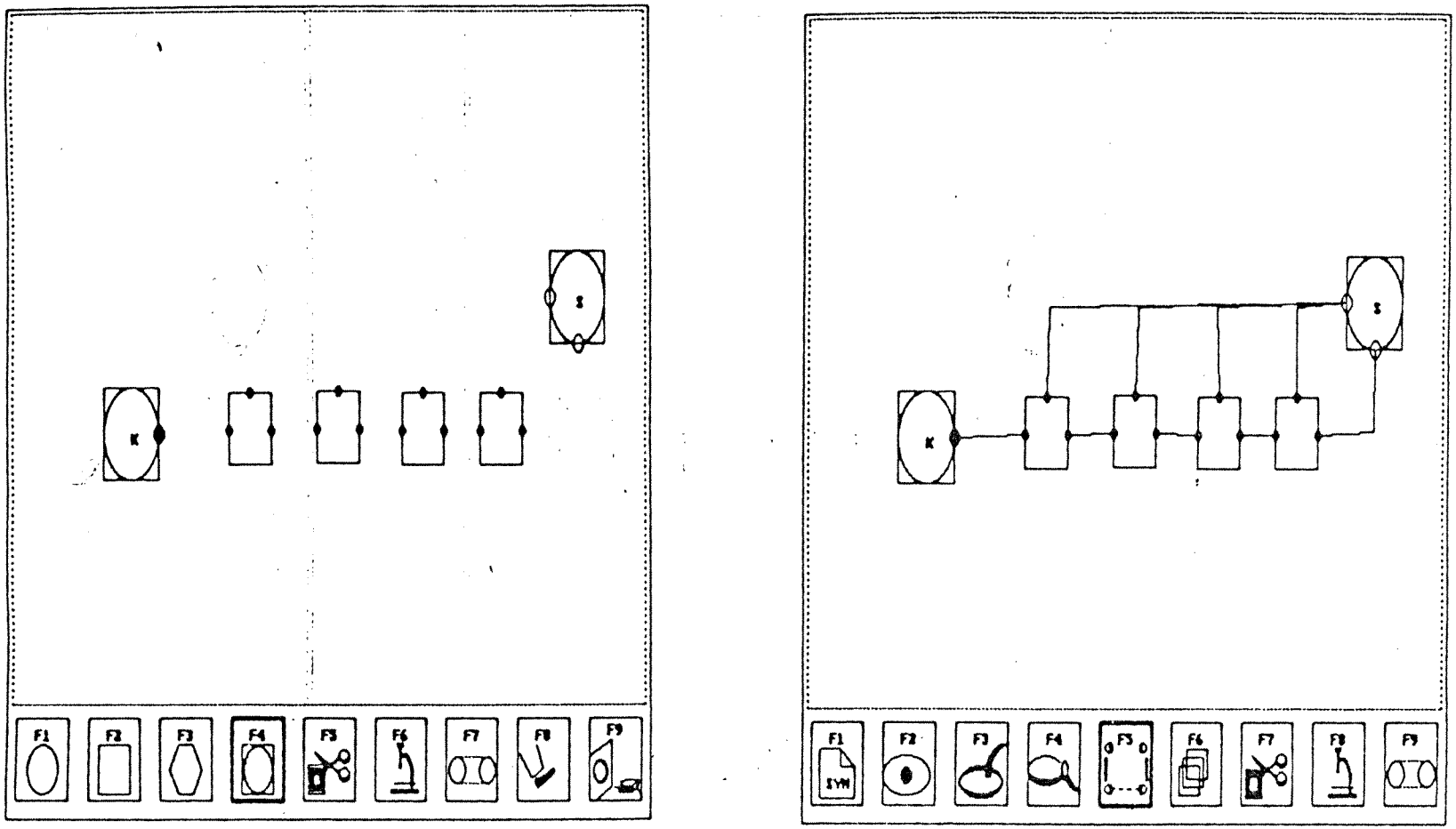


Fig. 2: Two menus from the GEPASS software. From a set of identical cells, a sink and a source the command *link_Channel* is used to form the complete system.

A typical syntactic tree might well consist of a set of nodes, one for every declared procedure, one for the system body and one for the body of the main program. A procedure declaration is represented in the tree by the procedure name and a list of formal parameters which correspond to the cell's ports. The system is a set of parallel calls for the following procedures: sink, source and cells.

3.2 SYNTACTIC EDITOR:

Since it has proved not practical to design graphically some parts of the proposed systolic solution, for instance the internal definition of the cell's computation, we developed a simplified Syntactic Editor for the OCCAM language. It mostly takes in charge only those parts of the OCCAM program that were left unexpanded by the Graphical Editor. Hence both the editors cooperate efficiently in deriving a complete OCCAM program that is syntactically and semantically correct.

Every entity or node in the syntactic tree is a process (i.e declaration or constructor process such as DEF and PAR) and can be manipulated through a set of commands based on the structural movement of the cursor. Such commands can be issued, for example, to insert, delete, select, move, copy or duplicate an entity. Of course, the syntactic integrity is always checked by the editor after every command.

While structural commands are needed to manipulate entities, textual commands are required to expand empty place holders found in entities. An example of a template and place holders is given bellow:

VAR *<variable_list>*:

PAR
  <process>

WHILE *<condition>*
  <process>

Where VAR, PAR and WHILE are key words of the language and the between brackets are place holders that need to be expanded.

4. VALIDATION AND EVALUATION:

After generating an OCCAM program, it is necessary to validate its logical behaviour and evaluate its performance. Since OCCAM compilers on PCs are not yet available on the market, we developed our own OCCAM interpreter, called **MICAM,** which stands for Micro-based Interpreter for programs in OCCAM [GHANEMI 1990c]. It accepts an OCCAM program as input, generates its associated model based on the Process-Oriented concept and then interpretes it, giving as output all the necessary results such as performance figures.

Because of the implemented language used for the interpreter (Turbo PASCAL) MICAM features an important characteristic of portability. Therefore, it could be easily used in any computer system, provided it has a Turbo PASCAL compiler.

In addition, it could run independently from GEPASS since it incorporates its own developed modules for systactic and semantic analysis.

A second feature in MICAM, worth mentionning here, is the fact that it was designed not specifically for systolic systems. Hence, it could be used to simulate and interprete any other synchronous or asynchronous parallel architectures such as SIMD, ARRAY and MIMD architectures. Furthermore, it could well be used by graduate and post-graduate students to design and evaluate their own developed synchronisation and communication mechanisms through the use of the OCCAM language.

5. CONCLUSION:

The use of CAD tools is well developed in most industrial, manufacturing and engineering fields. However, little work has been done to assist the systolic designer, i.e from the programming point of view. In this paper, we showed the need for sophisticated programming environment specially designed to program systolic systems and presented two software tools, GEPASS and MICAM, that can be integrated in this type of programming environment.

GEPASS which is a graphical editor for programming soft-systolic systems is a friendly and easy way to use software package that automatically generates OCCAM programs from their graphical representation. It features several commands that update and modify both the architectural structure and the OCCAM program. On the other hand, MICAM which is an OCCAM interpreter, that runs on PCs, allows the designer to validate and evaluate his programs.

Because of the interesting characteristics of the OCCAM language, MICAM can be well used to simulate and study other types of synchronous and asynchonous architectures. Furthermore, it could also be used for tutorial purposes for both undergraduate and postgraduate levels.

I must, however, acknowledge the following graduate students, Mr. Hocine M.H, Miss Diabi N., Miss Zerrari F.Z and Miss Zerrari S. who worked really hard to realise the proposed projects under my supervision. I am particulary pleased with the outstanding quality of the work they produced. To all of these I wish a very successful professional life.

REFERENCES:

Fagerstrom J. and Patel R.K.M [1986]:
"*High -level Simulation of Systolic Systems*", Research report, CADLAB, Jun 1986, Linkoping University, Swenden.

Ghanemi S. [1987]:
"*Non Numerical Parallel Algorithms for Asynchronous Parallel Computer Systems*", PhD. Thesis, The Computer Studies Dept., Loughborough University of Technology", Sept. 1987.

Ghanemi S., Houcine M.H and Diabi N. [1990a]:
"Programmation Assistee par Ordinateur a interface Graphique pour les Systemes Systoliques", memoire d'ingenieur, Institut d'Informatique, Universite d'Annaba Septembre 1990.

Ghanemi S., Houcine M.H and Diabi N. [1990b] :
"GEPASS: Guide de l'utilisateur", Annaba, Sept., 1990

Ghanemi S., Zerrari F.T. and Zerrari S. [1990c] :
"Simulation des Systemes Systoliques: Generation de modeles ", memoire d'ingenieur, Institut d'Informatique, Universite d'Annaba, Septembre 1990.

Kung H.T [1982] :
"Why Systolic Architectures",IEEE Computer, Jan. 1982

Kung S.Y [1988] :
"VLSI Array Processors", Prentice Hall, Englewood Cliffs New Jersey, USA.

Megson G.M [1984]:
"Soft-Systolic simulation of un-bounded problems", The Computer Studies Dept. Report, LUT, No 330, 1984.

Megson G.M [1986] :
"Nouvel Algorithms for the Soft-Systolic Paradigm ", PhD Thesis, The Computer Studies Department, Loughborough University of Technology, Feb. 1987.

Extending the independence notion to the first-order formulas for the Satisfiability Enumeration

H. DRIAS

Abstract

A considerable amount of research has been done on the satisfiability problem in the case of propositional calculus and on the NP-complete problems in general. We find in the literature of recent date, algorithms to solve the problem, approaches to enumerate the solutions of the problem and analysis of time and space complexities. Moreover, some variants of the satisfiability problem like 3-SAT[1], r,s-SAT [10], HORN-SAT[3] have been investigated. Other problems related to the satisfiability problem have been defined then explored. This is the case of MAX-SAT problem for instance that consists on maximizing the number of satisfied clauses.

However, little work has been developed on this problem in the case of the first-order logic. This expresses in a way the difficulty of the subject rather than the outweight of the importance of the propositional calculus case on the first-order calculus case. The latter having notably applications in the theory of databases and in artificial intelligence.

Our main contribution here is a theoretical study of the notion of independence of formulas in both propositional and first-order logic. The independence notion has been used to enumerate the solutions of the satisfiability problem in the case of propositional formulas[9].

The results achieved here emerge mainly from the intuitive idea of independence of two formulas. Roughly speaking, two formulas are independent if the sets of values that do not satisfy respectively these formulas have no shared value. First, we define formally the independence of two formulas for both cases and then establish results for the characterization of the independent formulas. In the propositional calculus setting, the notion yields very interesting results. On the other hand, we show the difficulty of characterizing independent first-order formulas in the general case and specify restricted cases where the notion is suited for the characterization.

Introduction

The satisfiability problem(SAT for short) is well known to be NP-complete. Many of its variants like 3-SAT, HORN-SAT and MAX-SAT have been explored enough to take into account some interesting results. For instance 3-SAT, which the satisfiability problem restricted to 3 literals per clause is NP-complete[1] while 2-SAT is polynomial[5]. The satisfiability problem in the case of Horn clauses denoted HORN-SAT, is solved in linear time [3].

These results concern the satisfiability problem only in the setting of propositional calculus. Part of our interest in the first-order logic case is the great variety of applications of its results notably in the theory of relational database query languages and knowledge bases in artificial intelligence.

We plan to undertake the enumeration of the solutions of the satisfiability problem in the setting of first-order calculus using the independence notion of clauses.This notion has been used in enumerating the solutions of a SAT instance in the propositional logic case. An algorithm has been developed to calculate the number of solutions of a SAT instance owing to this notion[10]. The same approach is also used to answer to the satisfiability of a given instance. Of course, the time complexity of the algorithm is exponential. A prunning method has been introduced to reduce the search tree.

Two clauses of a SAT instance are said to be independent if the set of boolean instanciations that do not satisfy respectively the clauses are disjoint. This leads us to transform a SAT instance into an equivalent set of independent clauses. Since the non-solutions of each clause of the latter are not redundant that is they do not appear as non-solutions in the other clauses, the total number of solutions can be achieved for the set of independent clauses and hence for the SAT instance.
In this paper, we present a formal study to shed light on the independence notion of formulas in both the propositional and first-order logic cases.

Definition1

let $\mathcal{P}$ and $\mathcal{B}$ be two well-formed formulas with respectively k_1 and k_2 the numbers of free variables of $\mathcal{P}$ and $\mathcal{B}$. Let $k = \text{maximum}(k_1,k_2)$. If $k \neq 0$, $\mathcal{P}$ and $\mathcal{B}$ are said to be independent for an interpretation M of domain $\mathcal{D}$ if the set of values of $\mathcal{D}^k$ that are in relation with $(\sim\mathcal{P})$ and the set of values of $\mathcal{D}^k$ that are in relation with $(\sim\mathcal{B})$ are disjoint or in other words, if the set of values of $\mathcal{D}^k$ that do not satisfy $\mathcal{P}$ and the set of values of $\mathcal{D}^k$ that do not satisfy $\mathcal{B}$ are disjoint.
If both formulas are closed that is $k = 0$, $\mathcal{P}$ and $\mathcal{B}$ are independent for an interpretation M of domain $\mathcal{D}$ if one of the two formulas is true whereas the other is false for the interpretation.($\sim$ denoting the negation connective)

Definition2

Two well formed formulas $\mathcal{P}$ and $\mathcal{B}$ are independent if for every interpretation M of domain $\mathcal{D}$, the set of values of $\mathcal{D}^k$ that do not satisfy $\mathcal{P}$ and the set of values of $\mathcal{D}^k$ that do not satisfy $\mathcal{B}$ are disjoint, k having the meaning given in Definition1.

Definition3

A literal formula is an atomic formula with or without the negation connective. In propositional calculus, a literal is defined to be a proposition with or without the negative connective.

Definition4

Two literal formulas are similar if they verify the following conditions:

a) They are unifiable

b) The unifier is homogenous that is it contains only variables, the names of functions do not appear.

Definition5

A first-order clause is a disjunction of literals formulas. In propositional calculus, a clause is a disjunction of literals.

Examples:

$$A_1^1(x) \vee A_1^2(x,y)$$

$$\sim A_1^2(w,z) \vee A_2^1(w)$$

In these two clauses, we have used the disjunctive operator v and the predicates names A_1^1, A_1^2 and A_2^1. In the following A_i^j designates a name of predicate with j terms. The subscript i is used to make the distinction between the different predicates having the same number of terms. The negation operator is denoted $\sim$

Proposition1

In the first-order calculus, two clauses $\mathcal{P} = \mathcal{P}_1 \vee \mathcal{P}_2 \vee \ldots\ldots \vee \mathcal{P}_n$ *and* $\mathcal{B} = \mathcal{B}_1 \vee \mathcal{B}_2 \vee \ldots\ldots \vee \mathcal{B}_m$ *are independent for an interpretation M if there exists in* $\mathcal{P}$ *a literal formula* $\mathcal{P}_i$ *and in* $\mathcal{B}$ *a literal formula* $\mathcal{B}_j$ *such that* $\mathcal{P}_i$ *and* $\sim\mathcal{B}_j$ *are similar* $1 \le i \le n$, $1 \le j \le m$.

Example:

$$\mathcal{P}(x,y) = A_1^1(x) \vee A_1^2(x,y)$$

and

$$\mathcal{B}(u,v) = \sim A_1^2(u,v) \vee A_2^1(v)$$

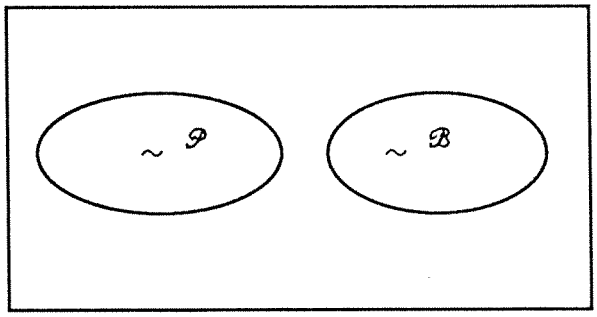
Proof. Case: $k \neq 0$, k having the meaning given in Definition1.
$\mathcal{P}_i$ being a literal formula of $\mathcal{P}$. Since $\mathcal{P}$ is a disjunction of literals, the values of $\mathcal{D}^k$ that do not satisfy $\mathcal{P}$, do not satisfy neither $\mathcal{P}_i$. (i)

Similarly, the values of $\mathcal{D}^k$ that do not satisfy $\mathcal{B}$, do not satisfy neither $\mathcal{B}_j$. Thus the values of $\mathcal{D}^k$ that do not satisfy $\mathcal{B}$, satisfy $\sim\mathcal{B}$.
$\mathcal{P}_i$ and $\sim\mathcal{B}_j$ being similar, the values of $\mathcal{D}^k$ that satisfy $\mathcal{P}_i$, satisfy also $\sim\mathcal{B}_j$ and the values of $\mathcal{D}^k$ that do not satisfy $\mathcal{P}_i$, do not satisfy neither $\sim\mathcal{B}_j$ but do satisfy $\mathcal{B}_j$ and afterwards $\mathcal{B}$. (ii)
Combining (i) and (ii), we obtain that the values of $\mathcal{D}^k$ that do not satisfy $\mathcal{P}$, satisfy $\mathcal{B}$. In other words, the set of values of $\mathcal{D}^k$ that do not satisfy $\mathcal{P}$ is included in the set of values of $\mathcal{D}^k$ that satisfy $\mathcal{B}$. But since the set of values of $\mathcal{D}^k$ that satisfy $\mathcal{B}$ and the set of values of $\mathcal{D}^k$ that do not satisfy $\mathcal{B}$ are disjoint, we conclude that the set of values of $\mathcal{D}^k$ that do not

satisfy $\mathcal{P}$ and the set of values of $\mathcal{D}^k$ that do not satisfy $\mathcal{B}$ are disjoint. And hence $\mathcal{P}$ and $\mathcal{B}$ are independent.

Case: k = 0 The result is obvious.(end of proof)

If two clauses $\mathcal{P}$ and $\mathcal{B}$ are independent for an interpretation M, the respective sets of values that do not satisfy these two clauses are disjoint. This can be schematized as follows:



The circle $\sim\mathcal{P}$ represents the set of values of $\mathcal{D}^k$ that do not satisfy $\mathcal{P}$. And the circle $\sim\mathcal{B}$ represents the set of values of $\mathcal{D}^k$ that do not satisfy $\mathcal{B}$.

If $\mathcal{P} = \mathcal{P}_1 \vee \mathcal{P}_2 \vee \ldots\ldots \vee \mathcal{P}_n$
and $\mathcal{B} = \mathcal{B}_1 \vee \mathcal{B}_2 \vee \ldots\ldots \vee \mathcal{B}_m$

where the $\mathcal{P}_i$ and the $\mathcal{B}_j$ are literal formulas, then

$$\sim\mathcal{P} = \sim\mathcal{P}_1 \wedge \sim\mathcal{P}_2 \wedge \ldots\ldots \wedge \sim\mathcal{P}_n$$

$$\sim\mathcal{B} = \sim\mathcal{B}_1 \wedge \sim\mathcal{B}_2 \wedge \ldots\ldots \wedge \sim\mathcal{B}_m$$

$\wedge$ being the conjunction operator.

If $\mathcal{P}$ and $\mathcal{B}$ are independent then (the circle $\sim\mathcal{B}$) $\cap$ (the circle $\sim\mathcal{P}$) $= \emptyset$
This amounts to say that

$$\sim\mathcal{P}_1 \wedge \sim\mathcal{P}_2 \wedge \ldots\ldots \wedge \sim\mathcal{P}_n \wedge \sim\mathcal{B}_1 \wedge \sim\mathcal{B}_2 \wedge \ldots\ldots \wedge \sim\mathcal{B}_m \qquad (1)$$

is false for the interpretation.

Lemma

$\mathcal{P}$ and $\mathcal{B}$ are independent for an interpretation M if anf only if $\sim\mathcal{P} \wedge \sim\mathcal{B}$ is false for this interpretation.

Proposition2

In the propositional calculus, two clauses $\mathcal{P}$ and $\mathcal{B}$ are independent if and only if there exist in $\mathcal{P}$ a literal formula $\mathcal{P}_i$ and in $\mathcal{B}$ a literal formula $\mathcal{B}_j$ such that $\mathcal{P}_i = \sim\mathcal{B}_j$

Proof. a)Let demonstrate that if there exist in $\mathcal{P}$ a literal $\mathcal{P}_i$ and in $\mathcal{B}$ a literal $\mathcal{B}_j$ such that $\mathcal{P}_i = \sim\mathcal{B}_j$ then $\mathcal{P}$ and $\mathcal{B}$ are independent.
In the propositional calculus, the $\mathcal{P}_i$ and the $\mathcal{B}_j$ are simple propositions without variable and which have the value true or false.The set of values that do not satisfy $\mathcal{P}$, do not satisfy $\mathcal{P}_i$ neither since

$\sim\mathcal{P} = \sim\mathcal{P}_1 \wedge \sim\mathcal{P}_2 \wedge \ldots\ldots \wedge \sim\mathcal{P}_n$

They do not satisfy neither $\sim\mathcal{B}_j$ since $\mathcal{P}_i = \sim\mathcal{B}_j$.
The set of values that do not satisfy $\mathcal{B}$, do not satisfy $\mathcal{B}_j$ neither since

$\mathcal{B} = \sim\mathcal{B}_1 \wedge \sim\mathcal{B}_2 \wedge \ldots\ldots \wedge \sim\mathcal{B}_m$

We conclude that the two sets are disjoint and thus $\mathcal{P}$ and $\mathcal{B}$ are independent.

b) Let's demonstrate the reciprocity. The set of values that do not satisfy $\mathcal{P}$, do not not satisfy $\mathcal{P}_i$ neither and the set of values that do not satisfy $\mathcal{B}$, do not satisfy $\mathcal{B}_j$ neither. Suppose that no literal $\mathcal{P}_i$ exists in $\mathcal{P}$ and no literal $\mathcal{B}_j$ exists in $\mathcal{B}$ such that $\mathcal{P}_i = \sim\mathcal{B}_j$. In this case, the expression (1) satisfy both $\sim\mathcal{P}$ and $\sim\mathcal{B}$. This contradicts the fact that $\sim\mathcal{P}$ and $\sim\mathcal{B}$ are satisfied by disjoint sets of values.(end of proof)

In the predicate calculus, the atomic formulas being relations, we may have the following situation:
$A \wedge B$ false for a given interpretation, with $A \neq \sim B$. A and B being literal formulas.

Example:
Let M be the following interpretation
$\mathcal{D} = \{ -20, -2, -1, 0, 3, 5\}$
$A \equiv x > 0$
$B \equiv x \leq -2$
The symbol $\equiv$ denotes the biconditional connective.

$A \wedge B$ is false for M though $A \neq \sim B$ for M.
From this fact, the previous proposition cannot be extended to the first-order calculus.

Proposition3

In the predicate calculus, two clauses $\mathcal{P}$ and $\mathcal{B}$ are independent if there exists in $\mathcal{P}$ a literal formula $\mathcal{P}_i$ and in $\mathcal{B}$ a literal formula $\mathcal{B}_j$ such that $\mathcal{P}_i$ and $\sim\mathcal{B}_j$ are similar.

Proof. Since no particular interpretation has been considered in the proof of proposition1, the result of the latter can be extended to any interpretation, Thus proposition3 holds.(end of proof)

To consider the independence of two first-order formulas, it is necessary first to convert each formula into a set of clauses. A conversion procedure may have the following steps:

1 - Elimination of all the logical operators except the $\sim$, $\vee$ and $\wedge$ operators.
example: the following logical formula:

$$(\forall x)\{P(x) \Rightarrow \{ (\forall y)\{P(y) \Rightarrow P(f(x,y))\} \wedge \sim(\forall y)\{Q(x,y) \Rightarrow P(y)\}\}\}$$

where $\Rightarrow$ is the logical implication connective and $\forall$ is the universal quantifier, becomes after the execution of the first step:

$$(\forall x)\{ \sim P(x) \vee \{ (\forall y)\{ \sim P(y) \vee P(f(x,y))\} \wedge \sim(\forall y)\{ \sim Q(x,y) \vee P(y)\}\}\}$$

2- Reduction of the scope of the negation connective. This step consists in eliminating the negation connective for the non atomic formulas and thus obtaining negation connectives only for atomic formulas. The previous formula after executing this operation becomes:

$$(\forall x)\{ \sim P(x) \vee \{ (\forall y)\{ \sim P(y) \vee P(f(x,y))\} \wedge (\exists y)\{Q(x,y) \vee \sim P(y)\}\}\}$$

$\exists$ being the existential quantifier.

3- Standardization of variables:
After the normalization of the variables, the formula of the example becomes:

$$(\forall x)\{ \sim P(x) \vee \{ (\forall y)\{ \sim P(y) \vee P(f(x,y))\} \wedge (\exists z)\{Q(x,z) \vee \sim P(z)\}\}\}$$

4- Elimination of the existential quantifier. After the execution of this operation on the example, we obtain:

$$(\forall x)\{ \sim P(x) \vee \{ (\forall y)\{ \sim P(y) \vee P(f(x,y))\} \wedge \{Q(x,g(x)) \vee \sim P(g(x))\}\}\}$$

5- Conversion into prenex form:

$$(\forall x)(\forall y)\{ \sim P(x) \vee \{ \{ \sim P(y) \vee P(f(x,y))\} \wedge \{Q(x,g(x)) \vee \sim P(g(x))\}\}\}$$

6- Conversion into conjunctive normal form:

$$(\forall x)(\forall y)\{\{ \sim P(x) \vee \sim P(y) \vee P(f(x,y))\} \wedge \{ \sim P(x) \vee Q(x,g(x))\} \wedge \{ \sim P(x) \vee \sim P(g(x))\}\}$$

7- Elimination of the universal quantifier:

$$\{ \sim P(x) \vee \sim P(y) \vee P(f(x,y))\} \wedge \{ \sim P(x) \vee Q(x,g(x))\} \wedge \{ \sim P(x) \vee \sim P(g(x))\}$$

8-Elimination of the conjunctive connectors. Finally, we obtain the system made up with the following clauses:

$\sim P(x) \vee \sim P(y) \vee P(f(x,y))$
$\sim P(x) \vee Q(x,g(x))$
$\sim P(x) \vee \sim P(g(x))$

Definition6

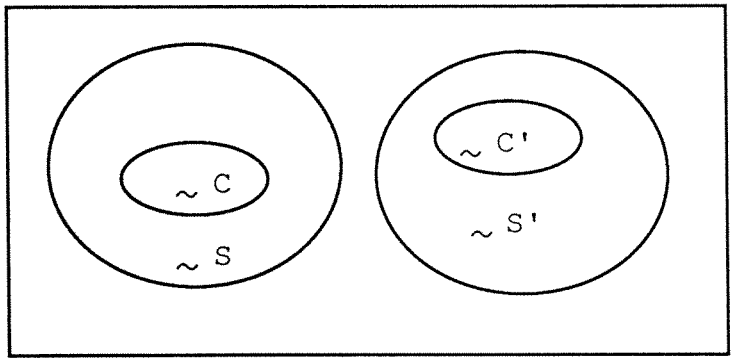
Two systems of clauses S and S' are said to be independent for an interpretation M if the set of values of $\mathcal{D}^k$ that do not satisfy S and the set of values of $\mathcal{D}^k$ that do not satisfy S' are disjoint. $\mathcal{D}$ is the domain of M and k is the maximum of the numbers of the free variables respectively in each of the two systems.

The independence notion of two systems is important since the number of solutions of the union of the two systems can be achieved by adding the numbers of solutions of the two systems.

Proposition4

In the first-order calculus, two systems of clauses S and S' are independent if and only if each clause of S is independent of each clause of S'.

Proof. a) Let prove that if two systems S and S' are independent, then each clause of S is independent of each clause of S'.
Let C be a clause of S. The set of values of $\mathcal{D}^k$ that do not satisfy C is included in the set of values of $\mathcal{D}^k$ that do not satisfy S. Similarly, if C' is a clause of S', the set of values of $\mathcal{D}^k$ that do not satisfy C' is included in the set of values of $\mathcal{D}^k$ that do not satisfy S'.
The set of values of $\mathcal{D}^k$ that do not satisfy S and the set of values of $\mathcal{D}^k$ that do not satisfy S' are disjoint since S and S' are independent by hypothesis. We deduct that the set of values of $\mathcal{D}^k$ that do not satisfy C and the set of values of $\mathcal{D}^k$ that do not satisfy C' are disjoint. Therefore C and C' are independent.



b) Let prove that if each clause of S is independent of each clause of S' then S and S' are independent.

Let suppose that S and S' are not independent. Then there are values of $\mathcal{D}^k$ that do not satisfy simultaneously S and S'. Let x be one of these values. There exists in S at least one clause C that is not satisfied by x and also there is in S' at least a clause C' that is not satisfied by x. This leads us to conclude that the set of values of $\mathcal{D}^k$ that do not satisfy C and the set of values of $\mathcal{D}^k$ that do satisfy C' are not disjoint and accordingly C and C' are not independent. This contradicts the initial hyposethis.(end of proof)

Proposition5

Two formulas are independent if their respective equivalent systems of clauses are independent.

Proof. The set of values that do not satisfy a formula is equal to the set of values that do not satisfy the system of clauses that is equivalent to this formula and reciprocally.
If two systems of clauses respectively equivalent to two formulas are independent, then the sets of values that do not satisfy these two systems are disjoint. Therefore, these sets of values do not satisfy respectively the two formulas. This amount to conclude that the two logical formulas are independent.(end of proof)

Conclusion

The independence notion has been introduced to enumerate the solutions of the satisfiability problem in the case of propositional formulas. In this paper, we have presented a formal synthesis of the independence notion in both the propositional and first-order calculus. We have defined the independence notion in both the propositional and the first-order theories for clauses, formulas and systems of clauses. Then we have tried to characterize each of these independent entities.

The findings of this theoretical study is summarized in the following points:
For the propositional calculus, the independence of clauses is vigorously characterized by the appearence of one literal in one clause and its opposite in the other one. Moreover when each clause of a SAT system is independent of each clause of another system of clauses then the two systems are independent. The independence of propositional formulas follows from the independence of systems of clauses since a formula can be mapped into a SAT system.

For the first-order theory, we fail to find a complete characterization of independent formulas in the general case. However, we have proved that if formulas appear in the form described in this note, they are independent, the reciprocity being false. In addition to that, we have shown that two systems of clauses are independent if each clause of the first system is independent of each clause of the second one. And finally, two formulas are independent if and only if their corresponding systems of clauses are independent.

References

[1] COOK S. 'The Complexity of Theorem-Proving Procedures' Proceedings of the 3th ACM Symposium on Theory of Computing, pp. 151-158(1971).

[2] DAVIS M. and PUTNAM H. 'A Comuting Procedure for Quantification Theory', J.Assoc. Comput. Mach. 7:201-215(1960).

[3] DOWLING W. and GALLIER J. 'Linear-Time Algorithms for Testing the Satisfiability of Propositional Horn Formulas' J.Logic Programming 3: 267-284 (1984).

[4] DRIAS H. ' On the number of solutions of a satisfiability instance with balanced variables' to appear in proceedings of the 13th British Computer Society Research Colloquium on Information Retrieval. Lancaster U.K. April(1991).

[5] EVEN S. , ITAI A. and SHAMIR A. ' On the Complexity of Timetable and Multicommodity flow problems'. SIAM J.Comput.5(1976) 691-703.

[6] GAREY M.R and JOHNSON D.S ' Computers and Intractability'. Freeman C° (1979).

[7] JAUMARD B. and SIMEONE B. 'On the Complexity of the Maximum Satisfiability Problem for Horn Formulas' Information Processing letters 26; 1-4(1987).

[8] MENDELSON E. ' Introduction to Mathematical Logic'. D. Van Nostrand Company INC (1966).

[9] SIEKMAN J.H. 'Unification Theory' Proceedings of the 7th ECAI, vol.2, pp vi-xxxv(1986).

[10] SIMON J.C and DUBOIS O. 'Finding the Number of Solutions of a Satisfiability Problem'. IEEE proceedings, N° 87-45381, IEEE LFA workshop, Vienna, aug (1987) 65-68.

[11] TOVEY C.A. 'A Simplified NP-complete Satisfiability Problem'. Discrete Applied Mathematics 8 (1984) 85-89. North-Holland.

VOUZZAVEDIBISAR,

UNIVERSITÉ DE LA MÉDITERRANÉE
AIX-MARSEILLE II

Marseille, 16 Février 1996

Le président

Monsieur KNIPPEL

CAB N°5572
Objet : déclaration de périodique
vos courrier des 17.1 et 7.2.96

Monsieur,

J'ai pris connaissance de votre courrier du 7 février relatif au renouvellement de la déclaration du "Bulletin d'Informatique Approfondie et Applications".

Je ne méconnais pas l'intérêt de cette initiative, toutefois l'université ne saurait s'engager scientifiquement et financièrement dans cette opération.

Je vous prie d'agréer, Monsieur, l'expression de mes sentiments les meilleurs.

Didier RAOULT

UNIVERSITÉ D'AIX · MARSEILLE

Jardin du Pharo . 58, Boulevard Charles Livon . 13284 Marseille cedex 07
Tél. : (33) 91 39 65 00 . Télécopie : (33) 91 31 31 36 . Minitel 3615 UNIVMED

Université de Provence
Atelier de Reprographie
Centre Saint Charles
3, place Victor Hugo
F - 13331 Marseille Cedex 3