MINISTERO DELL'ISTRUZIONE, DELL'UNIVERSITÀ E DELLA RICERCA
DIPARTIMENTO PER LA PROGRAMMAZIONE IL COORDINAMENTO E GLI AFFARI ECONOMICI - SAUS
PROGRAMMI DI RICERCA SCIENTIFICA DI RILEVANTE INTERESSE NAZIO NALE
RICHIESTA DI COFINANZIAMENTO
(*DM n. 20 del 19 febbraio 2002*)
PROGETTO DI UNA UNITÀ DI RICERCA - MODELLO B
Anno 2002 - prot. 2002017471_002

Parte: I

1.1 Programma di Ricerca di tipo: *interuniversitario*

Area Scientifico Disciplinare: ***Scienze Matematiche***

1.2 Durata del Programma di Ricerca: *24 mesi*

1.3 Coordinatore Scientifico del Programma di Ricerca

GABBRIELLI	**MAURIZIO**
(cognome)	(nome)
Università degli Studi di BOLOGNA	**Facoltà di SCIENZE MATEMATICHE FISICHE e NATURALI**
(università)	(facoltà)
INF/01	**Dipartimento di SCIENZE DELL'INFORMAZIONE**
(settore scient.discipl.)	(Dipartimento/Istituto)

gabbri@cs.unibo.it
(E-mail)

1.4 Responsabile Scientifico dell'Unità di Ricerca

BAGNARA	**ROBERTO**	
(cognome)	(nome)	
Professore associato	**07/12/1963**	**BGNRRT63T07D458Z**
(qualifica)	(data di nascita)	(codice di identificazione personale)
Università degli Studi di PARMA	**Facoltà di SCIENZE MATEMATICHE FISICHE e NATURALI**	
(università)	(facoltà)	
INF/01	**Dipartimento di MATEMATICA**	
(settore scient.discipl.)	(Dipartimento/Istituto)	

0521/032317	0521/032350	bagnara@cs.unipr.it
(prefisso e telefono)	(numero fax)	(E-mail)

1.5 Curriculum scientifico del Responsabile Scientifico dell'Unità di Ricerca

Testo italiano

Roberto Bagnara (laurea in Scienze dell'Informazione, Università di Pisa, 1992; Dottorato di Ricerca in Informatica, Università di Pisa, 1997) è stato Ricercatore in Informatica presso l'Università degli Studi di Parma dal 1997 al 2001, ed è ora Professore associato di Informatica presso la stessa Università.
È attivo da un decennio nei settori dell'analisi di programmi, dell'interpretazione astratta e, più in generale, delle tecniche di manipolazione dei programmi basate sulla semantica. È coautore di oltre 20 pubblicazioni su riviste e atti di conferenze internazionali.
Ha partecipato a diversi progetti di ricerca nazionali ed internazionali: ESPRIT Basic Research Action n. 6707 ("ParForce"); COFIN'99 "Certificazione automatica di programmi mediante interpretazione astratta"; COFIN'00 " Interpretazione astratta, sistemi di tipo e analisi Control-Flow"; COFIN'01 "Ragionamento su aggregati e numeri a supporto della programmazione e relative verifiche: dagli algoritmi di decisione alla programmazione con vincoli con multi-insiemi, insiemi e mappe". Nell'ambito delle "Azioni Integrate Italia-Spagna", è il coordinatore italiano del progetto "Ambienti avanzati per lo sviluppo di programmi logici" (IT229, 2001-2002).
È l'autore principale di "China", un analizzatore statico altamente innovativo per programmi logici con vincoli (vedi http://www.cs.unipr.it/China/). Guida il gruppo che ha progettato e implementato la "Parma Polyhedra Library" (PPL): la PPL è una libreria C++ moderna, robusta ed efficiente per la manipolazione di poliedri convessi (vedi http://www.cs.unipr.it/ppl/). Guida anche il progetto interdisciplinare "PURRS" (Parma University's Recurrence Relation Solver) in corso presso il Dipartimento di Matematica dell'Università di Parma. L'obiettivo del progetto è quello di avanzare significativamente lo stato dell'arte nel campo dell'analisi e della verifica automatica di complessità.

Testo inglese

Roberto Bagnara (laurea degree in Computer Science, University of Pisa, 1992; PhD in Computer Science, University of Pisa, 1997) has been an Assistant Professor of Computer Science at the University of Parma from 1997 to 2001 and is now Associate Professor of Computer Science at the same university.
He is active, since 10 years, in the fields of program analysis, abstract interpretation and, more generally, semantics-based program manipulation. He has coauthored more than 20 papers on journals and international conferences.
He participated to several international and national research projects: ESPRIT Basic Research Action Project n. 6707 ("ParForce"); COFIN'99 MURST project "Automatic Program Certification by Abstract

Interpretation"; COFIN'00 MURST project "Abstract Interpretation, Type Systems and Control-Flow Analysis"; COFIN'01 MURST project "Aggregate-
and Number-Reasoning for Computing: from Decision Algorithms to Constraint Programming with Multisets, Sets, and Maps". He is the Italian coordinator for the Italy-Spain bilateral project "Advanced Development Environment for Logic Programs" (IT229, 2001-2002). He is the main author of "China", a state-of-the-art data-flow analyzer for constraint logic programs (see http://www.cs.unipr.it/China/). He is leading the Parma Polyhedra Library (PPL) project: the PPL is a modern, robust and efficient C++ library for the manipulation of convex polyhedra (see http://www.cs.unipr.it/ppl/ for more details). He is also leading the interdisciplinary project PURRS (Parma University's Recurrence Relation Solver) at the Department of Mathematics of the University of Parma. The aim of this project, which is in an early stage of development, is to significantly advance the state of the art in automatic complexity analysis and verification.

1.6 Pubblicazioni scientifiche più significative del Responsabile Scientifico dell'Unità di Ricerca

1. *BAGNARA R. (1998).* ***A Hierarchy of Constraint Systems for Data-Flow Analysis of Constraint Logic-Based Languages****. SCIENCE OF COMPUTER PROGRAMMING. vol. 30, pp. 119-155 ISSN: 0167-6423 .*
2. *BAGNARA R., HILL P. M., ZAFFANELLA E. (2002).* ***Set-Sharing Is Redundant for Pair-Sharing****. THEORETICAL COMPUTER SCIENCE. ISSN: 0304-3975 In corso di pubblicazione.*
3. *BAGNARA R., SCHACHTE P. (1999).* ***Factorizing Equivalent Variable Pairs in ROBDD-Based Implementations of Pos****. Algebraic Methodology and Software Technology (AMAST'98, Amazonia, Brasile). (vol. 1548, pp. 471-485). Lecture Notes in Computer Science, Springer-Verlag, Berlin.*
4. *HILL P. M., BAGNARA R., ZAFFANELLA E. (2002).* ***Soundness, Idempotence and Commutativity of Set-Sharing****. THEORY AND PRACTICE OF LOGIC PROGRAMMING. vol. 2, pp. 155-201 ISSN: 1471-0684 .*
5. *ZAFFANELLA E., HILL P. M., BAGNARA R. (2002).* ***Decomposing Non-Redundant Sharing by Complementation****. THEORY AND PRACTICE OF LOGIC PROGRAMMING. vol. 2, pp. 233--261 ISSN: 1471-0684 .*

1.7 Risorse umane impegnabili nel Programma dell'Unità di Ricerca

1.7.1 Personale universitario dell'Università sede dell'Unità di Ricerca

Nº	Cognome	Nome	Dipart./Istituto	Qualifica	Settore scient.	Mesi uomo	
						2002	2003
Personale docente:							
1	*BAGNARA*	*ROBERTO*	*MATEMATICA*	*Prof. associato*	*INF/01*	*8 (ore: 1100)*	*8 (ore: 1100)*

2	*RICCI*	*GABRIELE*	*MATEMATICA*	*Prof. associato*	*INF/01*	*2 (ore: 275)*	*4 (ore: 550)*
3	*ROSSI*	*GIANFRANCO*	*MATEMATICA*	*Prof. ordinario*	*INF/01*	*2 (ore: 275)*	*5 (ore: 685)*
Altro personale:							

1.7.2 Personale universitario di altre Università

Nº	Cognome	Nome	Università	Dipart./Istituto	Qualifica	Settore scient.	Mesi uomo	
							2002	2003
Personale docente:								
Altro personale:								

1.7.3 Titolari di assegni di ricerca

Nº	Cognome	Nome	Dipart./Istituto	Anno del titolo	Mesi uomo	
					2002	2003
1	*ZAFFANELLA*	*ENEA*	*Dip. MATEMATICA*	*2000*	*7 (ore: 959)*	*8 (ore: 1100)*

1.7.4 Titolari di borse per Dottorati di Ricerca e ex L. 398/89 art.4 (post-dottorato e specializzazione)

Nº	Cognome	Nome	Dipart./Istituto	Anno del titolo	Mesi uomo

1.7.5 Personale a contratto da destinare a questo specifico programma

Nº	Qualifica	Costo previsto	Mesi uomo
1.	*Assegnista di ricerca*	*15000*	*11 (ore: 1507)*
2.	*Diplomato/laureato*	*8000*	*6 (ore: 825)*
3.	*Diplomato/laureato*	*1500*	*2 (ore: 275)*

1.7.6 Personale extrauniversitario dipendente da altri Enti

Nº	Cognome	Nome	Dipart./Istituto	Qualifica	Mesi uomo

2.1 Titolo specifico del programma svolto dall'Unità di Ricerca

Testo italiano

Tecniche e strumenti basati su vincoli ed interpretazione astratta per la verifica di sistemi complessi

Testo inglese

Constraint and Abstract Interpretation Techniques and Tools for the Verification of Complex Systems

2.2 Settori scientifico-disciplinari interessati dal Programma di Ricerca

- *INF/01 - INFORMATICA*

2.3 Parole chiave

Testo italiano
VERIFICA AUTOMATICA ; ANALISI STATICA ; VINCOLI ; INTERPRETAZIONE ASTRATTA ; ANALISI DI COMPLESSITÀ

Testo inglese
AUTOMATED VERIFICATION ; STATIC ANALYSIS ; CONSTRAINTS ; ABSTRACT INTERPRETATION ; COMPLEXITY ANALYSIS

2.4 Base di partenza scientifica nazionale o internazionale

Testo italiano

La crescente dipendenza della società dalle applicazioni informatiche fa sí che l'analisi e la verifica della correttezza dei sistemi complessi rappresenti sempre di più un fattore critico del processo di sviluppo. Il malfunzionamento dei sistemi, siano essi hardware, software o protocolli di comunicazione, può comportare danni rilevanti di ogni genere: dalla perdita finanziaria alla perdita di vite umane. Inoltre, quando i difetti non sono rilevati prima dell'impiego del sistema, l'applicazione di eventuali misure correttive è , quando possibile, ben più difficile e costosa. Esempi dal recente passato includono il millennium bug, gli errori di alcune versioni del processore Pentium, lo scoperto da 32 miliardi di dollari alla N.Y. Bank [6], il fallimento iniziale del vettore Ariane 5, e gli incidenti mortali del Therac-25 [29]. La verifica formale assistita dal calcolatore (computer-aided verification, il settore di ricerca che investiga su strutture dati ed algoritmi specializzati per la validazione automatica di software, hardware e protocolli) è dunque un campo di crescente interesse.
La tecnica attualmente più in uso per la verifica di sistemi a stati

finiti (sistemi con un numero finito di stati possibili) è il model checking [10], dove i modelli del sistema sono specificati come strutture di Kripke e le proprietà del sistema sono rappresentate da formule di un'opportuna logica temporale. Il problema del model checking consiste nel controllare se un modello di Kripke soddisfa una data formula temporale. Con l'impiego di strutture dati particolari, i Binary Decision Diagrams (BDD, una rappresentazione compatta per formule Booleane [8]), i sistemi di model checking simbolico [31] sono in grado di verificare sistemi con numero di stati fino a 1020 e sono stati applicati con successo alla validazione di progetti hardware e di protocolli di comunicazione.

Diversi costituenti critici della moderna tecnologia informatica sono caratterizzati da un numero di stati che cresce esponenzialmente con il numero di componenti del sistema. Altri hanno una natura intrinsecamente illimitata che, agli effetti pratici, può essere considerata infinita. Per questi sistemi le attuali tecniche di verifica basate su model-checking sono impraticabili. Questo progetto si propone di investigare tecniche di analisi e di verifica per tali sistemi basate su vincoli e interpretazione astratta.

Il passaggio dalla logica Booleana (come nel model checking di sistemi a stati finiti) a logiche basate su vincoli fornisce un modo naturale per estendere il paradigma del model checking a sistemi caratterizzati da spazi di stati infiniti o virtualmente tali. I vincoli, infatti, possono essere visti come rappresentazioni simboliche di insiemi, possibilmente infiniti, di stati del sistema. Come viene mostrato nei lavori pionieristici sull'interpretazione astratta, i sistemi di vincoli possono essere usati per la costruzione di domini astratti per l'analisi di sistemi complessi [13, 14]. I vincoli aritmetici lineari possono essere usati, ad esempio, per approssimare l'insieme degli stati dei sistemi real-time e ibridi [22, 23]. Ulteriori ricerche sembrano comunque necessarie per consolidare questi risultati e per avere un impatto significativo su una gamma più estesa di applicazioni.

Un sistema di vincoli utilizzato in molte applicazioni ha per elementi sistemi finiti di equazioni e disequazioni lineari. Questi vincoli definiscono il dominio dei poliedri convessi che viene utilizzato come dominio astratto in molti strumenti per l'analisi e la verifica di sistemi hardware e software. Le librerie esistenti per la manipolazione di poliedri convessi soffrono di alcune limitazioni che, in determinati contesti, rendono il loro utilizzo problematico [20, 21, 26, 30, 37]. Alcune librerie, non essendo in grado di gestire numeri a precisione arbitraria, possono incorrere in errori causati dall'overflow. Analogamente, le librerie che utilizzano l'aritmetica a virgola mobile, possono incorrere in problemi di underflow e, più in generale, ad errori di arrotondamento che possono invalidare il risultato finale. Molte librerie non forniscono alcun supporto per la gestione dei poliedri topologicamente non chiusi, ovvero dei poliedri ottenibili mediante l'aggiunta di vincoli di disuguaglianza stretta. Le poche librerie nelle quali questa estensione è possibile in realtà non forniscono un supporto completo, incorrendo in inefficienze evitabili e lasciando all'applicazione utente il compito non banale di interpretare i risultati ottenuti. Tutto questo in aggiunta a problemi di efficienza che limitano la loro applicabilità [24].

Un altro filone di ricerca riguarda proprietà dei sistemi complessi

che riguardano l'impiego di risorse, quali il tempo di calcolo impiegato, lo spazio di memoria occupato e il volume di messaggi scambiati sulla rete. Ad esempio, verificare che le scadenze dei sistemi hard real-time siano rispettare richiede la capacità di determinare (o approssimare per eccesso) il massimo tempo di esecuzione di ogni task. Il campo dell'analisi automatica di complessità non è stato molto esplorato, con alcune rilevanti eccezioni quali [17]. Nonostante questo, il tema è di grande interesse per una serie di tecniche e applicazioni emergenti quali gli agenti mobili, il data mining e il commercio elettronico.
Uno degli aspetti importanti per l'analisi di complessità è rappresentato dalla possibilità di risolvere, in modo completamente automatico, sistemi di relazioni di ricorrenza. Esistono in letteratura tecniche e programmi per la soluzione di ricorrenze [11,25] che però trattano una casistica piuttosto limitata e presuppongono l'interazione con un operatore umano. (Pacchetti commerciali come Mathematica e Maple, opportunamente guidati da un operatore, sono in grado di manipolare e semplificare alcune espressioni che rientrano nel dominio delle ricorrenze.)
È noto che è possibile dare una soluzione in forma chiusa (cioè, essenzialmente, una formula con un numero finito di segni di somma) per tutte le relazioni di ricorrenza di ordine finito a coefficienti costanti, in cui l'eventuale parte non omogenea è una combinazione lineare di prodotti di polinomi, esponenziali e funzioni trigonometriche elementari. Lo stesso risultato vale per i sistemi di ricorrenze a coefficienti costanti di ordine uno, in cui l'eventuale parte non omogenea è dello stesso tipo (si veda, ad esempio, [27]). La ricerca degli ultimi 50 anni circa ha portato all'estensione di questi risultati al caso in cui la parte non omogenea è più generale (per esempio è una funzione razionale) oppure i coefficienti della ricorrenza non sono costanti [35, 38]. In questi casi si cercano soluzioni esprimibili come combinazione lineare di funzioni ipergeometriche (o di altre funzioni trascendenti note), o come somme parziali delle serie che definiscono le stesse. Risulta dunque necessario utilizzare strumenti di somma approssimata per le serie risultanti, o per le loro somme parziali.

Testo inglese

Our society is becoming increasingly dependent on computer applications. As a consequence, the ability to analyze and verify the functional correctness of complex systems as they are being developed is of increasing importance. Malfunctioning of a system, whether it be hardware, software or a communication protocol, can cause serious damage of all kind, from financial loss to casualties. Furthermore, corrective measures, when at all possible, are far more difficult and expensive than they would be if the defects had been caught earlier. Widely known examples from the near past are the millennium bug, the errors in some versions of the Pentium processor, the N.Y. Bank $32 billion overdraft [6], the initial failure of the Ariane 5 rocket, and the Therac-25 accidents [29]. Computer-aided verification is the research field devoted to the investigation of specialized algorithms and data structures for the automatic validation of hardware and protocols.
Model checking [10] is one of the most popular techniques used for the

verification of finite state systems (i.e., systems with a finite number of possible states). In model checking, the models of the system are given as Kripke structures and the system's properties are represented by formulae of a suitable temporal logic. The model checking problem amounts to determining whether or not a Kripke model satisfies a specific temporal formula. Data structures such as Binary Decision Diagrams (BDD, a compact representation for Boolean formulae [8]) allow a technique called symbolic model checking [31] to deal with systems where the number of states has order 1020. This technique has been successfully applied to the validation of hardware designs and communication protocols.

Several critical parts of modern computer-based technology are characterized by a number of states which grows exponentially with the number of the components of the system. Other systems have an intrinsic unbounded nature that, for all practical purposes, can be considered infinite. For these systems, the usual verification techniques based on model checking are not strong enough. In this project we want to study more powerful analysis and verification techniques that are based on constraints [33] and abstract interpretation [13].

Passing from Boolean logic (as in finite-state model checking) to constraint-based logics gives us a natural way to extend the paradigm of model checking to systems with infinite state space. Constraints, in fact, can be seen as symbolic representations of infinite sets of system states. Moreover, as shown in seminal works on abstract interpretation [13, 14], constraint systems can be used to build abstract domains for the conservative analysis of complex systems. For instance, linear arithmetic constraints can be used to approximate the set of states of real-time and hybrid systems [22, 23]. Further research seems to be necessary to consolidate these results so as to achieve a significant impact on a broader range of applications.

One constraint system that has been used in several applications has linear equations and inequalities as its elements. These constraints define the domain of convex polyhedra that is an abstract domain employed in several systems for the analysis and verification of hardware and software components. The existing libraries for the manipulation of convex polyhedra suffer from limitations that, in particular contexts, make their usage problematic [20, 21, 26, 30, 37]. Some libraries cannot handle arbitrary precision numbers and can thus incur overflow problems. Similarly, libraries that use floating point numbers can cause underflow problems and, more generally, suffer from rounding errors that can make the final result totally unreliable.

Several libraries do not support the handling of polyhedra that are topologically not closed, i.e., polyhedra that can be obtained by combining equations, non-strict and strict inequalities. The few libraries that are able to deal with such polyhedra do not provide full support for them, resulting in avoidable inefficiencies and leaving the client with the non trivial task of the correct interpretation of the obtained results. To this must be added other efficiency problems that limit their applicability [24].

Another research direction concerns those properties of complex systems that deal with resource usage: computation time, required memory space, network bandwidth used. For example, verifying that the deadlines of hard real-time systems are met, requires a means of

determining (or bounding from above) the maximum execution time of each task. The field of automatic complexity analysis is largely unexplored with some relevant exceptions such as [17]. Nevertheless, this theme is of great interest for a range of emerging techniques and applications such as mobile agents, data mining and electronic commerce.

One of the most important aspects of complexity analysis is the possibility of solving, in a completely automated way, systems of recurrence relations. In the literature, there are both techniques and software for solving recurrences [11, 25], but they only deal with a rather resticted range of cases, and assume the interaction with a human operator. (Commercial packages like Mathematica and Maple can, properly guided by an operator, handle and simplify expressions that belong to the domain of recurrence relations).

It is known that it is possible to give a closed form solution (that is, essentially, a formula with a finite number of summation signs) for all finite order recurrences with constant coefficients, whose non-homogeneous part, if any, is a linear combination of products of polynomials, exponentials and elementary trigonometrical functions. The same result holds for systems of order one recurrences with constant coefficients, whose non-homogeneous part is of the same kind (see, e.g., [27]). In the last 50 years or so, research led to the extension of this result to the case where the non-homogeneous part is more general (for instance, it is a rational function) or the coefficients are not constant [35, 38]. In these cases, one looks for solutions expressed in terms of linear combinations of hypergeometric functions (or other known transcendental functions), or as partial sums of their defining series. It follows that tools are needed for computing approximate sums of such series.

2.4.a Riferimenti bibliografici

[1] A. Aiken. Introduction to set constraint-based program analysis. Science of Computer Programming, 35(2):79--111, 1999.

[2] C. Ancourt. Gé né ration Automatique de Codes de Transfert pour Multiprocesseurs à Mé moires Locales. PhD thesis, Université de Paris VI, March 1991.

[3] R. Bagnara. Data-Flow Analysis for Constraint Logic-Based Languages. PhD thesis, Dipartimento di Informatica, Università di Pisa, Pisa, Italy, 1997. Printed as Report TD-1/97.

[4] R. Bagnara. A hierarchy of constraint systems for data-flow analysis of constraint logic-based languages. Science of Computer Programming, 30(1--2):119--155, 1998.

[5] R. Bagnara, P. M. Hill, E. Ricci, and E. Zaffanella. The Parma Polyhedra Library User's Manual. Department of Mathematics, University of Parma, Parma, Italy, release 0.3 edition, February 2002. Available at http://www.cs.unipr.it/ppl/.

[6] J. M. Berry. Computer snarled N.Y. bank: $32 billion overdraft resulted from snafu. The Washington Post, (13 December 1985):D7, 1985.

[7] N. Bjø rner, A. Browne, M. Coló n, B. Finkbeiner, Z. Manna, M. Pichora, H. B. Sipma, and T. E. Uribe. STeP: The Stanford Temporal Prover (Educational Release) User's Manual. Computer Science

Department, Stanford University, Stanford, California, version 1.4-a edition, July 1998. Available at http://www-step.stanford.edu/.
[8] R. E. Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. ACM Computing Surveys, 24(3):293--318, 1992.
[9] P. Celis. Remark: Corrections and errors in John Ivie's some MACSYMA programs for solving recurrence relations. ACM Transactions on
Mathematical Software, 10(4):477--478, 1984. See [25].
[10] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Transactions on Programming Languages and Systems,
8(2):244--263, 1986.
[11] J. Cohen and J. Katcoff. Symbolic solution of finite-difference equations. ACM Transactions on Mathematical Software, 3(3):261--271, 1977.
[12] P. Cousot and R. Cousot. Static determination of dynamic properties of programs. In B. Robinet, editor, Proceedings of the Second International Symposium on Programming, pages 106--130. Dunod,
Paris, France, 1976.
[13] P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In Proceedings of the Fourth Annual ACM Symposium on Principles of Programming Languages, pages 238--252, 1977.
[14] P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In Conference Record of the Fifth Annual ACM Symposium on Principles of Programming Languages, pages 84--96, Tucson, Arizona, 1978. ACM Press.
[15] E. Davis. Constraint propagation with interval labels. Artificial Intelligence, 32:281--331, 1987.
[16] T. Dean. Temporal imagery: An approach to reasoning about time for planning and problem solving. Technical Report 433, Yale University, New Haven, CT, 1985.
[17] S. Debray and N.-W. Lin. Cost analysis of logic programs. ACM Transactions on Programming Languages and Systems, 15(5):826--875, 1993.
[18] D. L. Dill. Timing assumptions and verification of finite-state concurrent systems. In J. Sifakis, editor, Proceedings of the International Workshop on Automatic Verification Methods for Finite State Systems, volume 407 of Lecture Notes in Computer Science, pages 197--212, Grenoble, France, 1989. Springer-Verlag, Berlin.
[19] A. Dovier, C. Piazza, E. Pontelli, and G. Rossi. Sets and constraint logic programming. ACM Transactions on Programming Languages and Systems, 22(5):861--931, 2000.
[20] K. Fukuda. Polyhedral computation FAQ. Swiss Federal Institute of Technology, Lausanne and Zurich, Switzerland, available at http://www.ifor.math.ethz.ch/~fukuda/fukuda.html, 1998.
[21] E. Gawrilow and M. Joswig. polymake: a framework for analyzing convex polytopes. In G. Kalai and G. M. Ziegler, editors, Polytopes - Combinatorics and Computation, pages 43--74. Birkhä user, 2000.
[22] N. Halbwachs, Y.-E. Proy, and P. Roumanoff. Verification of real-time systems using linear relation analysis. Formal Methods in System Design, 11(2):157--185, 1997.

[23] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech: A model checker for hybrid systems. Software Tools for Technology Transfer, 1(1+2):110--122, 1997.
[24] T. A. Henzinger, J. Preussig, and H. Wong-Toi. Some lessons from the hytech experience. In Proceedings of the 40th Annual Conference on Decision and Control, pages 2887--2892. IEEE Computer Society Press, 2001.
[25] J. Ivie. Some MACSYMA programs for solving recurrence relations. ACM Transactions on Mathematical Software, 4(1):24--33, 1978. See also [9].
[26] B. Jeannet. Convex Polyhedra Library, release 1.1.3c edition, March 2002. Documentation of the ''New Polka'' library available at http://www.irisa.fr/prive/Bertrand.Jeannet/newpolka.html.
[27] W. G. Kelley and A. C. Peterson. Difference Equations. An Introduction with Applications. Academic Press, Inc. Harcourt Brace Jovanovich, 1991.
[28] K. Larsen, F. Larsson, P. Pettersson, and W. Yi. Efficient verification of real-time systems: Compact data structure and state-space reduction. In Proceedings of the 18th IEEE Real-Time Systems Symposium (RTSS'97), pages 14--24, San Francisco, CA, 1997. IEEE Computer Society Press.
[29] N. G. Leveson and C. S. Turner. An investigation of the Therac-25 accidents. IEEE Computer, 26(7):18--41, 1993.
[30] V. Loechner and D. Wilde. Parameterized polyhedra and their vertices. International Journal of Parallel Programming, 25(6):525--549, 1997.
[31] K. L. McMillan. Symbolic Model Checking. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1993.
[32] A. Miné . A new numerical abstract domain based on difference-bound matrices. In Proceedings of the 2nd Symposium on Programs as Data Objects (PADO 2001), volume 2053 of Lecture Notes in Computer Science, pages 155--172, Aarhus, Denmark, 2001. Springer-Verlag, Berlin.
[33] U. Montanari. Networks of constraints: Fundamental properties and applications to picture processing. Information Sciences, 7:95--132, 1974.
[34] T. S. Motzkin, H. Raiffa, G. L. Thompson, and R. M. Thrall. The double description method. In H. W. Kuhn and A. W. Tucker, editors, Contributions to the Theory of Games -- Volume II, number 28 in Annals of Mathematics Studies, pages 51--73. Princeton University Press, Princeton, New Jersey, 1953.
[35] M. Petkovsek, H. S. Wilf, and D. Zeilberger. A=B. A. K. Peters, 1996.
[36] S. Vere. Planning in time: Windows and durations for activities and goals. IEEE Trans. Patt. Anal. Mach. Intell., 5(3):246--267, 1983.
[37] D. K. Wilde. A library for doing polyhedral operations. Master's thesis, Oregon State University, Corvallis, Oregon, December 1993. Also published as IRISA Publication interne 785, Rennes, France, 1993.
[38] H. S. Wilf. Generatingfunctionology. Academic Press, 1994.

2.5 Descrizione del programma e dei compiti dell'Unità di Ricerca

Testo italiano

L'unità di ricerca, che ha maturato significative esperienze nel campo dei vincoli (in particolare insiemistici [1, 19] e aritmetici [3, 5]), dell'interpretazione astratta e dell'analisi dei programmi, si propone di investigare l'uso dei vincoli nei problemi di analisi e verifica di sistemi complessi.

Il primo obiettivo è quello di studiare linguaggi di specifica basati su vincoli che si prestino a rappresentare il comportamento dei sistemi di interesse e a realizzare le procedure per la loro verifica ed analisi. Il problema da affrontare ha dunque una natura duplice: da una parte si cercano linguaggi di vincoli espressivi, che possano modellare le proprietà di interesse con sufficiente precisione; dall'altra si cercano tecniche efficienti per la manipolazione di questi vincoli, che possano costituire strumenti effettivi di verifica. Rivestono una notevole importanza, per il presente progetto, quei sistemi il cui comportamento è parametrico rispetto ad una o più quantità ignote come, ad esempio, il numero di clienti in un protocollo per sistemi con architettura client-server. Per questo motivo, si presterà particolare attenzione a quei sistemi di vincoli (e ai relativi risolutori) in grado di esprimere proprietà di sistemi aperti e/o parametrici.

Riguardo ai vincoli aritmetici, il dominio dei poliedri convessi [14] è molto usato sia in analisi statica che nel campo della verifica [22, 23] per la rappresentazione di proprietà esprimibili mediante sistemi di equazioni e disequazioni lineari. Anche quando le proprietà in questione sono più complesse e caratterizzabili solamente mediante l'utilizzo di vincoli non lineari, in molti casi l'approssimazione mediante poliedri convessi fornisce risultati accettabili in termini di precisione. Il dominio dei poliedri convessi è tipicamente implementato applicando il metodo della doppia descrizione [34] che, nel contesto dell'analisi statica e verifica di proprietà, risulta essere più efficiente del metodo del simplesso. Comunque, nessuna delle implementazioni disponibili può , a buon diritto, considerarsi soddisfacente.

Un problema che intendiamo affrontare in questo progetto è quello della gestione appropriata dei poliedri non necessariamente chiusi (NNC), ovvero di quei poliedri definibili a partire da sistemi di vincoli che possono contenere disuguaglianze lineari strette. Il poter trattare le disuguaglianze strette è importante, ad esempio, per esprimere regioni temporali che non si intersecano, come viene spesso richiesto per modellare applicazioni in cui sono presenti concorrenza, protocolli di sincronizzazione, interazione asincrona e vincoli di temporizzazione. Attualmente, nessuna implementazione del metodo della doppia descrizione offre un supporto completo per i poliedri NNC. Quelle che li supportano parzialmente, tipicamente rappresentano ogni disuguaglianza stretta utilizzando una disuguaglianza non stretta che coinvolge una variabile fittizia aggiuntiva, denotata e. Vengono però lasciati all'utente sia l'onere (non gravoso) di codificare i poliedri NNC nella nuova rappresentazione, nonché quello (più problematico) di interpretare correttamente i risultati così ottenuti. Un'altra limitazione di questo approccio è dovuta al fatto che i sistemi di vincoli calcolati, pur essendo minimizzati sullo spazio di codifica (quello esteso con la variabile e), non lo sono nello spazio codificato (quello con le disuguaglianze strette). Questo comporta sia delle inefficienze evitabili, sia la necessità di complicare ulteriormente il codice utente in modo da riconoscere i

vincoli ridondanti. Ci proponiamo, partendo da un'opportuna estensione della nozione di doppia descrizione, di giungere a tecniche più soddisfacenti per la gestione delle disuguaglianze strette.
La pur elevata espressività del dominio dei poliedri convessi non è sufficiente per la verifica di tutti i sistemi ibridi [24]. Una possibilità che intendiamo investigare è quella di aumentare l'espressività dei domini di verifica utilizzando gli operatori di composizione di domini di vincoli introdotti in [4]. Tali costruzioni consentono, in modo automatico, di ''migliorare'' un dominio dato incorporando informazioni di tipo implicativo e disgiuntivo, dando luogo ad un'algebra di domini di vincoli dalle proprietà interessanti.
Alcune operazioni sul dominio dei poliedri convessi sono caratterizzate da una complessità computazionale che, nel caso peggiore, è esponenziale. D'altra parte, il dominio degli intervalli [12], pur essendo computazionalmente semplice, offre approssimazioni talvolta troppo grossolane. Una soluzione intermedia è costituita da domini di differenze vincolate, inizialmente studiati nel campo dell'intelligenza artificiale [15, 16, 36] e, successivamente, in quelli della verifica automatica [18, 28] e dell'analisi statica [3, 32]. In [3, Chapter 5] sono state studiate diverse tecniche di inferenza approssimata (sempre mutuate dal campo dell'intelligenza artificiale) ed è stato mostrato come queste possano essere interpretate in termini di domini di vincoli per l'analisi statica. Resta da vedere, e ci proponiamo di farlo in questo progetto, se ed in che modo questi domini di intermedia espressività e complessità possano essere utilizzati per la verifica dei sistemi a stati infiniti.
Per il raggiungimento degli obiettivi del progetto, è importante che le diverse componenti e proprietà dei sistemi sotto osservazione possano essere descritte utilizzando una pluralità di domini di vincoli, in grado di fornire informazioni alternative e complementari. Per questo motivo, in aggiunta allo studio di particolari sistemi di vincoli, studieremo metodologie generali, come quella descritta in [4], per la combinazione di domini di vincoli differenti.
Oltre allo studio di nuove tecniche e metodologie, l'unità si propone di realizzare strumenti innovativi per l'analisi e la verifica. Il progetto della Parma Polyhedra Library [5], una libreria per i poliedri convessi basata sul metodo della doppia descrizione, il cui prototipo è stato sviluppato dal nostro gruppo, ha come obiettivo quello di fornire un'implementazione completa e professionale di tutti quei miglioramenti che la ricerca di cui sopra dovesse individuare. Essendo basata sull'aritmetica intera a precisione arbitraria, la libreria è immune da eventuali problemi di arrotondamento e di overflow. Aspetto caratterizzante del progetto sarà l'applicazione sistematica di tecniche di computazione incrementale e lazy, al fine di limitare per quanto possibile i problemi di efficienza. Tra gli obiettivi perseguiti, ci si propone di fornire un supporto completo alla definizione e all'utilizzo delle disuguaglianze strette (ovvero dei poliedri NNC), non solo da un punto di vista implementativo, ma soprattutto per quanto riguarda la completezza e l'intuitività dell'interfaccia funzionale. Verranno inoltre studiati algoritmi ottimizzati per particolari classi di poliedri, comunemente utilizzate in numerosi campi applicativi, come la classe dei poliedri non negativi (ovvero, sottoinsiemi dello spazio

vettoriale i cui elementi hanno tutte le coordinate non negative) e la classe dei poliedri che approssimano uno spazio di soluzioni discreto (come gli Z-polyhedra [2]).
Particolare attenzione verrà prestata all'interoperabilità dei vari domini di vincoli e alle questioni di scalabilità. È chiaro che ogni dominio di vincoli di complessità non banale dà potenzialmente luogo a problemi di terminazione e di eccessiva occupazione di memoria. Una possibilità (altamente insoddisfacente) è quella di sperare per il meglio ed ''uccidere'' i processi troppo esigenti [7, 24]. Quello che ci proponiamo è invece la realizzazione di strumenti in cui il trade-off tra espressività ed efficienza sia regolabile in modo completamente automatico. La procedura di verifica o di analisi dovrebbe dunque essere in grado di ''scalare marcia'' automaticamente. Per default vengono usate descrizioni espressive (ed onerose): i poliedri convessi, ad esempio. Quando l'esecuzione di un'operazione su dette descrizioni impiega troppo tempo, oppure richiede troppa memoria, il sistema opera un cambio di rappresentazione passando, diciamo, ad un sistema di differenze vincolate; esegue l'operazione richiesta in questa rappresentazione meno precisa, ed eventualmente converte il risultato ottenuto in un poliedro convesso. Se anche l'operazione sulle differenze vincolate risultasse di costo eccessivo, il gioco si può ripetere passando ad una descrizione basata su intervalli, in cui molte operazioni interessanti hanno complessità lineare. Tutto questo è possibile solo se le descrizioni coinvolte sono implementate in modo da poter gestire, rimanendo consistenti e senza perdita di risorse, eventi quali lo scadere di un timeout o il lancio di un'eccezione di memoria: nulla del genere è disponibile nelle implementazioni esistenti.
L'unità di ricerca si propone di operare una svolta nel campo dell'analisi automatica e della verifica di complessità. Uno degli obiettivi è dunque quello della risoluzione automatica di relazioni di ricorrenza per quanto possibile generali. In particolare, ci proponiamo di ottenere:
** Soluzione esatta delle ricorrenze a coefficienti costanti di ordine finito, con parte non omogenea della forma prodotto di polinomio ed esponenziale.*
** Soluzione esatta di sistemi di ricorrenze di ordine uno con parte non omogenea della forma prodotto di polinomio ed esponenziale.*
** Soluzione esatta delle ricorrenze a coefficienti costanti di ordine finito (o di sistemi di ricorrenze) con parte non omogenea generica, in termini di somme parziali di funzioni ipergeometriche o di altre funzioni note, quando questo è possibile, o di loro combinazioni lineari a coefficienti polinomiali o esponenziali.*
** Soluzione esatta delle ricorrenze a coefficienti variabili di ordine uno, con parte non omogenea generica, in termini delle funzioni ipergeometriche o di altre funzioni note, come nel punto precedente.*
** Soluzione approssimata di tutte le ricorrenze descritte sopra, quando la soluzione esatta non è possibile, oppure quando è troppo complicata da gestire.*
** Soluzione esatta o approssimata di ricorrenze generalizzate (del tipo x(n) = 2*x(n/2)+n, che compaiono, per esempio, nello studio di tutti gli algoritmi divide et impera).*
È importante notare che, per le applicazioni di verifica automatica, la soluzione approssimata che ci interessa è sempre nella forma di una minorazione e di una maggiorazione esplicite per la soluzione, e non

nella forma di un termine principale con un termine d'errore dal comportamento asintotico noto, ma del quale non sono date altre informazioni (ad esempio, concernenti il suo segno). In altre parole, ci interessa garantire che la soluzione si trova fra due funzioni ''semplici'' date esplicitamente.
Per quanto detto sopra riguardo all'analisi e verifica di sistemi aperti e/o parametrici, ci poniamo l'obiettivo di operare anche con ricorrenze simboliche, ovvero in cui alcuni coefficienti non sono dati esplicitamente.

Testo inglese

The research unit has a significant experience in the fields of constraints (set constraints and arithmetic constraints in particular), abstract intepretation and program analysis. The unit plans to investigate the use of constraints in the analysis and verification of complex systems.
Our first objective is the identification of constraint-based specification languages that are suitable for modelling the behavior of the systems of interest and for implementing the corresponding analysis and verification procedures. The problem has thus a twofold nature: on the one hand we look for an expressive constraint language that can model the properties of interest with sufficient precision; on the other hand we look for efficient techniques for the manipulation of such constraints that can form the basis of effective verification tools. For the present project, especially important are those systems whose behavior is parametric with respect to one or more unknown quantities, such as, for example, the number of clients in a protocol for systems characterized by a client-server architecture. For this reason we will pay particular attention to those constraint systems (and the corresponding solvers) that can express properties of open and/or parametric systems.
When considering arithmetic constraints in the context of static analysis and verification [22, 23], one of the preferred choices is the domain of convex polyhedra [14]. An element of this domain can encode any property which can be described by a finite system of linear equalities and inequalities. Even when considering more complex properties, whose precise description would require the handling of non-linear constraints, the approximation provided by convex polyhedra often yields results of acceptable precision. In the context of static analysis and verification, the domain of convex polyhedra is typically implemented by using the double description method [34], which for these purposes turns out to be more efficient than the simplex method. However, none of the available implementations can be considered fully satisfactory.
One of the issues that will be targeted in this project is an adequate handling of the polyhedra that are not necessarily closed (NNC polyhedra, for short), i.e., all the polyhedra that can be described by systems of constraints where strict inequalities are allowed to occur. Strict inequalities are important, for instance, in order to directly represent non-intersecting temporal regions, as it is often the case when modeling applications where concurrency, synchronization protocols, asynchronous interactions and temporal constraints come into play. Nowadays, no implementation of the double description method offers full support for NNC polyhedra. Partial support is

provided in a few software libraries: typically, each strict inequality constraint is represented by a corresponding non-strict inequality involving an additional variable, usually denoted by e. However, it is up to the user to take care that NNC polyhedra are correctly encoded (an easy task, this one) and to properly reinterpret the yielded results (a much more problematic and error-prone task, requiring some expertise from the user). Another limitation of this approach is given by the fact that the computed constraint systems, while being in minimal form on the representation domain (i.e., the vector space extended with variable e), are not in minimal form as far as the original vector space is concerned, so that they may still encode strict inequalities that are redundant. This situation, besides causing avoidable inefficiencies, adds even more responsibilities for the user, who may need to further complicate the code so as to identify and filter out redundant constraints. Our aim is to provide a much more user friendly approach to the handling of strict inequalities, starting from a suitable generalization of the double description method.

The verification of some properties of hybrid systems [24] may require the availability of computation domains whose precision is beyond that attainable by the standard domain of convex polyhedra. A solution we are going to investigate is the possibility of increasing the expressivity of the available domains by applying the constraint domain operators introduced in [4]. These domains constructions can automatically improve a given domain by incorporating disjunctive and/or implicative information, yielding a constraint domain algebra satisfying interesting properties.

A few of the operations defined on the domain of convex polyhedra are characterized by an exponential worst-case complexity. On the other hand, the interval domain [12], while being much simpler from a computational point of view, often yields approximations that are too coarse. An intermediate solution is given by domains encoding constrained differences, initially proposed in the AI field [36, 16, 15] and later also adopted for automatic verification [18, 28] and static analysis [3, 32]. Several other approximate inference techniques (even in this case, originally developed for the AI field) have also been considered in [3, Chapter 5], where it was shown how these can all be interpreted in terms of constraint domains for static analysis. Another goal of the project will be an investigation of domains of intermediate expressivity and complexity, with the aim of studying how they can be applied to the verification of infinite state systems.

For this project to be successful it is important that the different components and properties of the systems under investigation can be described by constraints. This will require a variety of constraint domains, providing alternative and complementary information. For this reason, the research on specific constraint systems will be coupled by the study of general methodologies, such as the one described in [4], for the combination of different constraint domains.

Besides the study of new techniques and methodologies, one of the aims of our research unit is the development of useful tools for static analysis and verification. The goal of the Parma Polyhedra Library project [5] (a software library for the domain of convex polyhedra based on the double description method, whose prototype has been developed by our group) is to provide a complete and highly

professional implementation of any improvement arising from the above mentioned theoretical research. Being built upon a library providing arbitrary precision integer arithmetic, our library is immune from both rounding and overflow problems. Another characterizing aspect of our project will be the systematic application of incremental and lazy computation techniques, so as to avoid, as much as possible, the efficiency problems. As a main target, the library will provide full support for the definition of NNC polyhedra, both from the point of view of the implementation and, more importantly, with respect to the usability and the functional completeness of the user interface. Moreover, we plan to investigate specialized algorithms working on restricted classes of polyhedra that are commonly used in several application fields, such as the class of non-negative polyhedra (for approximating those subsets of the vector space whose elements have all coordinates greater than or equal to zero) and the class of polyhedra approximating a discrete subset of the space (such as the Z-polyhedra [2]).

Particular care will be taken with the possibility of interfacing the library with other constraint domains and with scalability issues. Clearly, any constraint domain of non-trivial complexity may incur non-termination problems and/or excessive memory usage. One, far from satisfactory, possible solution is to simply hope for the best and eventually ''kill'' those processes requiring more than the available resources [7, 24]. In contrast, what we propose is the development of more professional tools where the trade-off between expressivity and efficiency can be automatically tuned. The static analysis or verification procedure, by default, should consider the more descriptive (and costly) constraint domains, such as the convex polyhedra. When the computation of an operation on these descriptions turns out to require too much time, or memory space, the system should perform a change of representation, for instance by approximating the polyhedra using a system of constrained differences; after the execution of the requested operation on this less precise constraint system, the result may eventually be converted back into a convex polyhedron. If the cost of computing with constrained differences still happens to be unaffordable, then the system should perform a further change of representation, e.g., scaling down to the domain of interval constraints, where most of the interesting operations have linear complexity. Such a scenario is feasible only if the considered constraint systems are implemented in such a way that they can consistently react, without any loss of system resources, to events such as a timeout or an exception thrown by the memory allocation routines. No such feature is provided by any of the available implementations.

We wish to significantly advance the field of automatic complexity analysis and verification. One of our aims is, therefore, the automatic solution of recurrence relations in as much generality as possible. In particular, we intend to obtain

** An exact solution of finite order recurrence relations with constant coefficients, whose non-homogeneous part is the product of a polynomial and an exponential.*

** An exact solution of sets of recurrence relations of order one with constant coefficients, whose non-homogeneous part is the product of a polynomial and an exponential.*

** An exact solution of finite order recurrence relations (or sets*

of such relations) with constant coefficients, with a generic non-homogeneous part, in terms of partial sums of hypergeometric series or other known functions (whenever it is possible), or linear combinations of these functions with polynomial or exponential coefficients.
** An exact solution of order one recurrences with variable coefficients, with a generic non-homogeneous part, in terms of hypergeometric functions or other known functions, as above.*
** An approximate solution of all recurrences described above, when the exact solution is not possible, or it is too difficult or complex to handle.*
** An exact or approximate solution of generalized recurrences (such as x(n) = 2*x(n/2)+n), which appear, for instance, in the study of all divide et impera algorithms.*
It is important to remark that, for the applications for automatic verification that we intend to pursue, the approximate solution that we seek is always in the form of an explicit lower and upper bound of the solution, and not in the form of a main term with an error term with known asymptotic behavior, but without further information (for instance, concerning its sign). In other words, we are interested in ensuring that the solution lies between two explicit ''simple'' functions.
What we stated above concerning analysis and verification of open and/or parametric systems, implies that we also intend to handle symbolic recurrences, that is, recurrences where some coefficients are not given explicitly.

2.6 Descrizione delle attrezzature già disponibili ed utilizzabili per la ricerca proposta

Nº	Anno di acquisizione	Descrizione	
		Testo italiano	Testo inglese

2.7 Descrizione della richiesta di Grandi attrezzature (GA)

Attrezzatura I
Descrizione

valore presunto (Euro) **percentuale di utilizzo per il programma**

Attrezzatura II
Descrizione

valore presunto (Euro) **percentuale di utilizzo per il programma**

2.8 Mesi uomo complessivi dedicati al programma

	numero	mesi uomo

Personale universitario dell'Università sede dell'Unità di Ricerca (docenti)	*3*	*29* *(ore: 3973)*
Personale universitario dell'Università sede dell'Unità di Ricerca (altri)	*0*	*0*
Personale universitario di altre Università (docenti)	*0*	*0*
Personale universitario di altre Università (altri)	*0*	*0*
Titolari di assegni di ricerca	*1*	*15* *(ore: 2055)*
Titolari di borse dottorato e post-dottorato	*0*	*0*
Personale a contratto	*3*	*19* *(ore: 2603)*
Personale extrauniversitario	*0*	*0*
Totale	***7***	***63*** ***(ore: 8631)***

Parte: III

3.1 Costo complessivo del Programma dell'Unità di Ricerca

Voce di spesa	Spesa, Euro	Descrizione	
		Testo italiano	Testo inglese
Materiale inventariabile	*8.000*	*2 personal computer "PC compatibili", uno dei quali in configurazione "server"; 1 masterizzatore; 1 stampante laser; libri*	*2 PC compatible personal computers, one of those in server configuration; 1 CD writer; 1 laser printer; books*
Grandi Attrezzature			
Materiale di consumo e funzionamento	*1.000*	*spese telefoniche; fotocopie; dischetti e CD; spese di manutenzione*	*telephone expenses; xerox-copies; diskettes; CDs; maintenance expenses*
Spese per calcolo ed elaborazione dati			
Personale a contratto	*24.500*	*1 assegno di ricerca annuale (11*	*1 one-year research grant (11*

		mesi/uomo); 1 programmatore esperto (6 mesi/uomo); 1 addetto ai servizi di segreteria con mansioni amministrative e organizzative/esecutive che, per carenza di personale del Dipartimento, dovrebbero essere altrimenti svolte da personale meglio impiegato nell'attività di ricerca (2 mesi/uomo complessivi, ma frazionati su tutta la durata del progetto secondo le necessità).	*man-months); 1 senior programmer (6 man-months); 1 secretary charged of administrative and organization/executive duties that, for lack of personnel at the Department, should otherwise be carried out by personnel that would be best employed in the research activity (2 man-months in total, but distributed on the entire duration of the project depending on necessity).*
Servizi esterni			
Missioni	*12.500*	*incontri di lavoro con altri componenti del progetto e spese di viaggio e soggiorno per la partecipazione a conferenze e workshop di interesse per le ricerche relative al progetto*	*working meetings with other members of the project and travel and living expenses for the participation to conferences and workshops of interest for the research topics of the project*
Pubblicazioni			
Partecipazione / Organizzazione convegni	*3.000*	*tasse di iscrizione a conferenze e workshop di interesse per le ricerche relative al progetto*	*registration fees for conferences and workshops of interest for the research topics of the project*
Altro	*1.000*	*seminari*	*seminars*

Il progetto e' gia' stato cofinanziato da altre amministrazioni pubbliche o private (art. 4 bando 2002)? *NO*

Amministrazioni cofinanziatrici:

	Euro
Costo complessivo del Programma dell'Unità di Ricerca	***50.000***

Costo minimo per garantire la possibilità di verifica dei risultati	***36.000***
Fondi disponibili (RD)	***3.000***
Fondi acquisibili (RA)	***12.000***
Cofinanziamento di altre amministrazioni pubbliche o private (art. 4 bando 2002)	***0***
Cofinanziamento richiesto al MIUR	***35.000***

Parte: IV

4.1 Risorse finanziarie già disponibili all'atto della domanda e utilizzabili a sostegno del Programma

QUADRO RD

Provenienza	Anno	Importo disponibile, Euro	Note
Università			
Dipartimento	*2001*	*3.000*	*Fondi FIL*
CNR			
Unione Europea			
Altro			
TOTAL		***3.000***	

4.1.1 Altro

4.2 Risorse finanziarie acquisibili in data successiva a quella della domanda e utilizzabili a sostegno del programma nell'ambito della durata prevista

QUADRO RA

Provenienza	Anno della domanda o stipula del contratto	Stato di approvazione	Quota disponibile per il programma, Euro	Note
Università	*2002*	*disponibile in caso di accettazione della domanda*	*12.000*	*Fondo locale cofinanziamento*
Dipartimento				
CNR				
Unione Europea				
Altro				
TOTAL			***12.000***	

4.2.1 Altro

4.3 Certifico la dichiarata disponibilità e l'utilizzabilità dei fondi di cui ai punti 4.1 e 4.2: *SI*

Firma _____

(per la copia da depositare presso l'Ateneo e per l'assenso alla diffusione via Internet delle informazioni riguardanti i programmi finanziati; legge del 31.12.96 n° 675 sulla "Tutela dei dati personali")

Firma **18/04/2002**
_____ **13:28:38**