MINISTERO DELL'UNIVERSITÀ E DELLA RICERCA
DIREZIONE GENERALE PER IL COORDINAMENTO E LO SVILUPPO DELLA RICERCA
PROGRAMMI DI RICERCA SCIENTIFICA DI RILEVANTE INTERESSE NAZIONALE
RICHIESTA DI COFINANZIAMENTO (DM n. 1175 del 18 settembre 2007)

PROGETTO DI UNITÀ DI RICERCA - MODELLO B
Anno 2007 - prot. 2007NPW748_002

1 - Area Scientifico-disciplinare

01: Scienze matematiche e informatiche 100%

2 - Durata del Progetto di Ricerca

24 Mesi

3 - Coordinatore Scientifico

RANZATO **FRANCESCO**

Professore Associato confermato

Università degli Studi di PADOVA

Facoltà di SCIENZE MATEMATICHE FISICHE e NATURALI

Dipartimento di MATEMATICA PURA E APPLICATA

il progetto partecipa alla riserva del 10% di cui al punto 7 dell'art. 3

4 - Responsabile dell'Unità di Ricerca

BAGNARA	**ROBERTO**	
Professore Associato confermato	**07/12/1963**	**BGNRRT63T07D458Z**
Università degli Studi di PARMA		
Dipartimento di MATEMATICA		
0521/906917 (Prefisso e telefono)	**0521/906950** (Numero fax)	**bagnara@cs.unipr.it**

5 - Curriculum scientifico

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 quindicennio 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 30 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" COFIN'02 "Verifica di sistemi reattivi basata su vincoli" COFIN'04 "Interpretazione astratta: progettazione ed applicazioni". Per questi ultimi due progetti e` stato responsabile dell'unità di Parma. Nell'ambito delle "Azioni Integrate Italia-Spagna", è stato 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 per programmi logici con vincoli (http://www.cs.unipr.it/China/). Guida il gruppo che ha sviluppato la "Parma Polyhedra Library" (PPL): la PPL è una libreria moderna, robusta ed efficiente per la manipolazione di poliedri convessi (http://www.cs.unipr.it/ppl/). Guida anche il progetto interdisciplinare "PURRS" (Parma University's Recurrence Relation Solver, http://www.cs.unipr.it/purrs/). Ha dato inizio al progetto ECLAIR, il cui obiettivo finale è la realizzazione di un analizzatore statico innovativo per linguaggi imperativi.

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. For more than 10 years he has been active in the fields of program analysis, abstract interpretation and, more generally, semantics-based program manipulation. He has coauthored more than 30 papers that have been published in journals and proceedings of international conferences. He has participated in 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"; COFIN'02 MURST project "Constraint Based Verification of Reactive Systems"; COFIN'04 MURST project "AIDA - Abstract Interpretation Design and Applications." For the last two projects, he was responsible for the Parma research unit. He was 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 (http://www.cs.unipr.it/China/). He is leading the Parma Polyhedra Library (PPL) project: the PPL is a modern, robust and efficient library for the manipulation of convex polyhedra (http://www.cs.unipr.it/ppl/). He is also leading the PURRS project (Parma University's Recurrence Relation Solver, http://www.cs.unipr.it/purrs/). He initiated the ECLAIR project, whose final aim is the realization of an innovative static analyzer for imperative languages.

6 - Pubblicazioni scientifiche più significative del Responsabile dell'Unità di Ricerca

1. BAGNARA R., DOBSON K, HILL P. M, MUNDELL M, ZAFFANELLA E. (2007). Grids: A Domain for Analyzing the Distribution of Numerical Values. 16th International Symposium on Logic-based Program Synthesis and Transformation (LOPSTR'06). July 2006. (vol. 4407, pp. 219-235). ISBN/ISSN: 3-540-28584-9. doi:10.1007/978-3-540-71410-1_16 Lecture Notes in Computer Science, Springer-Verlag, Berlin.
2. BAGNARA R., HILL P. M, ZAFFANELLA E. (2007). Applications of Polyhedral Computations to the Analysis and Verification of Hardware and Software Systems. Quaderno 458. 18 gennaio 2007. Dipartimento di Matematica, Università di Parma.
3. BAGNARA R., HILL P. M, ZAFFANELLA E. (2006). The Parma Polyhedra Library: Toward a Complete Set of Numerical Abstractions for the Analysis and Verification of Hardware and Software Systems. Quaderno 457. 18 dicembre 2006. Dipartimento di Matematica, Università di Parma.
4. BAGNARA R., HILL P. M, ZAFFANELLA E. (2006). Widening Operators for Powerset Domains. INTERNATIONAL JOURNAL ON SOFTWARE TOOLS FOR TECHNOLOGY TRANSFER. vol. 8(4/5), pp. 449-466 ISSN: 1433-2779. doi:10.1007/s10009-005-0215-8.
5. BAGNARA R., HILL P. M., MAZZI E., ZAFFANELLA E. (2005). Widening Operators for Weakly-Relational Numeric Abstractions. 12th International Symposium on Static Analysis (London, UK). (vol. 3672, pp. 3-18). ISBN/ISSN: 3-540-28584-9. doi:10.1007/11547662_3 Lecture Notes in Computer Science, Springer-Verlag, Berlin.
6. BAGNARA R., HILL P. M., RICCI E., ZAFFANELLA E. (2005). Precise Widening Operators for Convex Polyhedra. SCIENCE OF COMPUTER PROGRAMMING. vol. 58(1-2), pp. 28-56 ISSN: 0167-6423. doi:10.1016/j.scico.2005.02.003.
7. BAGNARA R., HILL P. M., ZAFFANELLA E. (2005). Not Necessarily Closed Convex Polyhedra and the Double Description Method. FORMAL ASPECTS OF COMPUTING. vol. 17(2), pp. 222-257 ISSN: 0934-5043. doi:10.1007/s00165-005-0061-1.
8. BAGNARA R., RODRIGUEZ-CARBONELL E., ZAFFANELLA E. (2005). Generation of Basic Semi-algebraic Invariants Using Convex Polyhedra. 12th International Symposium on Static Analysis (London, UK). (vol. 3672, pp. 19-34). ISBN/ISSN: 3-540-28584-9. doi:10.1007/11547662_4 Lecture Notes in Computer Science, Springer-Verlag, Berlin.
9. BAGNARA R., ZAFFANELLA E., HILL P. M. (2005). Enhanced Sharing Analysis Techniques: A Comprehensive Evaluation. THEORY AND PRACTICE OF LOGIC PROGRAMMING. vol. 5(1&2), pp. 1-43 ISSN: 1471-0684. doi:10.1017/S1471068404001978.
10. MESNARD F., BAGNARA R. (2005). cTI: a Constraint-Based Termination Inference Tool for ISO-Prolog. THEORY AND PRACTICE OF LOGIC PROGRAMMING. vol. 5(1&2), pp. 243-257 ISSN: 1471-0684. doi:10.1017/S1471068404002017.
11. BAGNARA R., GORI R., HILL P. M., ZAFFANELLA E. (2004). Finite-Tree Analysis for Constraint Logic-Based Languages. INFORMATION AND COMPUTATION. vol. 193(2), pp. 84-116 ISSN: 0890-5401. doi:10.1016/j.ic.2004.04.005.
12. HILL P. M., ZAFFANELLA E., BAGNARA R. (2004). A Correct, Precise and Efficient Integration of Set-Sharing, Freeness and Linearity for the Analysis of Finite and Rational Tree Languages. THEORY AND PRACTICE OF LOGIC PROGRAMMING. vol. 4, pp. 289-323 ISSN: 1471-0684. doi:10.1017/S1471068403001868.
13. BAGNARA R., HILL P. M, ZAFFANELLA E. (2003). Widening Operators for Powerset Domains. 5th Int'l Conf. on Verification, Model Checking and Abstract Interpretation. (vol. 2937, pp. 135-148). ISBN/ISSN: 3-540-20803-8. doi:10.1007/b94790 Lecture Notes in Computer Science, Springer-Verlag, Berlin.
14. BAGNARA R., HILL P. M., RICCI E., ZAFFANELLA E. (2003). Precise Widening Operators for Convex Polyhedra. 10th International Symposium on Static Analysis (San Diego, California, USA). (vol. 2694, pp. 337-354). ISBN/ISSN:

3-540-40325-6. Lecture Notes in Computer Science, Springer-Verlag, Berlin.
15. BAGNARA R., HILL P. M., ZAFFANELLA E. (2002). Set-Sharing Is Redundant for Pair-Sharing. THEORETICAL COMPUTER SCIENCE. vol. 1-2, pp. 3-46 ISSN: 0304-3975. doi:10.1016/S0304-3975(00)00312-1.
16. BAGNARA R., RICCI E., ZAFFANELLA E., HILL P. M. (2002). Possibly Not Closed Convex Polyhedra and the Parma Polyhedra Library. 9th International Symposium on Static Analysis (Madrid, Spagna). (vol. 2477, pp. 213-229). ISBN/ISSN: 3-540-44235-9. Lecture Notes in Computer Science, Springer-Verlag, Berlin.
17. 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. doi:10.1017/S1471068401001338.
18. 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. doi:10.1017/S1471068401001351.
19. BAGNARA R., GORI R., HILL P. M., ZAFFANELLA E. (2001). Finite-Tree Analysis for Constraint Logic-Based Languages. 8th International Symposium on Static Analysis (Parigi, Francia). (vol. 2126, pp. 165-184). ISBN/ISSN: 3-540-42314-1. Lecture Notes in Computer Science, Springer-Verlag, Berlin.
20. BAGNARA R., ZAFFANELLA E., GORI R., HILL P. M. (2001). Boolean Functions for Finite-Tree Dependencies. 8th Internat'l Conf. on Logic for Progr., A.I. and Reasoning (Havana, Cuba). (vol. 2250, pp. 579-594). ISBN/ISSN: 3-540-42957-3. Lecture Notes in Computer Science, Springer-Verlag, Berlin.
21. BAGNARA R., HILL P. M., ZAFFANELLA E. (2000). Efficient Structural Information Analysis for Real CLP Languages. Logic for Programming and Automated Reasoning (Isola della Réunion, Francia). (vol. 1955, pp. 189-206). Lecture Notes in Computer Science, Springer-Verlag, Berlin.
22. BAGNARA R., ZAFFANELLA E., HILL P. M. (2000). Enhanced Sharing Analysis Techniques: A Comprehensive Evaluation. Principles and Practice of Declarative Programming (Montreal, Canada). (pp. 103-114). Association for Computing Machinery.
23. 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.
24. ZAFFANELLA E, BAGNARA R., HILL P. M. (1999). Widening Sharing. Principles and Practice of Declarative Programming (Parigi, Francia). (vol. 1702, pp. 414-432). ISBN/ISSN: 3-540-66540-4. doi:10.1007/10704567 Lecture Notes in Computer Science, Springer-Verlag, Berlin.
25. ZAFFANELLA E., HILL P. M., BAGNARA R. (1999). Decomposing Non-Redundant Sharing by Complementation. 6th International Symposium on Static Analysis (Venezia, Italia). (vol. 1694, pp. 69-84). ISBN/ISSN: 3-540-66459-9. Lecture Notes in Computer Science, Springer-Verlag, Berlin.
26. BAGNARA R. (1998). A Hierarchy of Constraint Systems for Data-Flow Analysis of Constraint Logic-Based Languages. SCIENCE OF COMPUTER PROGRAMMING. vol. 30(1-2), pp. 119-155 ISSN: 0167-6423. doi:10.1016/S0167-6423(97)00009-9.
27. HILL P. M., BAGNARA R., ZAFFANELLA E. (1998). The Correctness of Set-Sharing. 5th International Symposium on Static Analysis (Pisa, Italia). (vol. 1503, pp. 99-114). ISBN/ISSN: 3-540-65014-8. Lecture Notes in Computer Science, Springer-Verlag, Berlin.
28. BAGNARA R., HILL P. M., ZAFFANELLA E. (1997). Set-Sharing is Redundant for Pair-Sharing. 4th International Symposium on Static Analysis (Parigi, Francia). (vol. 1302, pp. 53-67). ISBN/ISSN: 3-540-63468-1. Lecture Notes in Computer Science, Springer-Verlag, Berlin.
29. BAGNARA R. (1996). A Reactive Implementation of Pos using ROBDDs. Programming Languages: Implementations, Logics and Programs (Aachen, Germania). (vol. 1140, pp. 107-121). doi:10.1007/3-540-61756-6_80 Lecture Notes in Computer Science, Springer-Verlag, Berlin.

30. BAGNARA R., GIACOBAZZI R., LEVI G. (1993). An Application of Constraint Propagation to Data-Flow Analysis. Nineth Conference on Artificial Intelligence for Applications (Orlando, Florida). (pp. 270-276). IEEE Computer Society Press, Los Alamitos, CA.

7 - Elenco dei partecipanti all'Unità di Ricerca

7.1 - Componenti

Componenti della sede dell'Unità di Ricerca

nº	Cognome	Nome	Università/Ente	Qualifica	Impegno 1° anno	Impegno 2° anno
1.	BAGNARA	Roberto	Università degli Studi di PARMA	Professore Associato confermato	8	6
2.	ZAFFANELLA	Enea	Università degli Studi di PARMA	Professore Associato non confermato	8	6
	TOTALE				**16**	**12**

Componenti di altre Università / Enti vigilati

nº	Cognome	Nome	Università/Ente	Qualifica	Impegno 1° anno	Impegno 2° anno
1.	AMATO	Gianluca	Università degli Studi "G. d'Annunzio" CHIETI-PESCARA	Ricercatore confermato	8	6
	TOTALE				**8**	**6**

Titolari di assegni di ricerca
Nessuno

Titolari di borse
Nessuno

7.2 - Altro personale

Nessuno

7.3 - Personale a contratto da destinare a questo specifico Progetto

nº	Tipologia di contratto	Costo previsto	Impegno 1° anno	Impegno 2° anno	Note
1.	Altre tipologie	20.000	3	9	Contratti di prestazione per lo svolgimento di attività di ricerca
	TOTALE	**20.000**	**3**	**9**	

7.4 - Dottorati a carico del PRIN da destinare a questo specifico Progetto

Nessuno

8 - Titolo specifico del Progetto svolto dall'Unità di Ricerca

Testo italiano
Tecniche scalabili di interpretazione astratta per l'analisi e la verifica di programmi imperativi

Testo inglese
Scalable Abstract Interpretation Techniques for the Analysis and Verification of Imperative Programs

9 - Abstract del Progetto svolto dall'Unità di Ricerca

Testo italiano
L'utilizzo dell'interpretazione astratta (IA) [CC77,CC79] e le sue applicazioni all'analisi di programmi hanno avuto un sensibile incremento negli ultimi 15 anni. Per esempio, ASTRÉE, un analizzatore statico basato sulla teoria dell'IA, ha potuto dimostrare l'assenza di errori a tempo di esecuzione nel sistema fly-by-wire dell'Airbus A340 (132000 linee di codice C specializzato). Stiamo raggiungendo il punto nel quale analisi statiche basate sull'IA

potranno essere incluse negli ambienti di sviluppo dei linguaggi imperativi più diffusi e vi sono realtà seriamente interessate ad ogni progresso nel campo: per esempio, molte compagnie nell'industria automobilistica (incluse FIAT e Magneti Marelli) hanno progetti interni che ambiscono ad aumentare l'affidabilità del software per mezzo di analisi statiche. Un ulteriore segnale in tal senso è dato dal progetto GlobalGCC (GGCC) all'interno del programma ITEA (Information Technology for European Advancement), il cui scopo è estendere la GNU Compiler Collection (GCC) per mezzo di tecniche di analisi statica a livello di programma. Inserendosi al livello della rappresentazione interna GIMPLE, le analsisi statiche di GGCC saranno disponibili per tutti i linguaggi sorgente e tutti i sistemi target supportati da GCC. Il consorzio GCCC comprende, tra gli altri, Bertin Technologies, CEA-LIST, INRIA-Futurs, Mandriva, Telefonica I+D e UPM. Anche se i tempi sono maturi, affinché questa rivoluzione nello sviluppo del software (almeno quello critico) possa prendere piede occorre individuare le giuste soluzioni a molte delle problematiche ancora sul tappeto. L'unità di Parma possiede le competenze e l'esperienza che possono essere utili per risolvere alcuni di questi nodi. L'unità di Parma si concentrerà sui

WP1: Progettazione di interpretazioni astratte
WP2: Analisi statica mediante interpretazione astratta

per mezzo dei task riassunti di seguito.

Task PR-1.1: Domini astratti per l'analisi dei puntatori e dell'aliasing in programmi C e Java

La maggior parte della ricerca sulle analisi dei puntatori e dell'aliasing è focalizzata sulla compilazione ottimizzata e, in tale contesto, solo le analisi più efficienti sono considerate utili. Alcune sperimentazioni hanno però mostrato che la precisione di tali analisi è insufficiente per la verifica automatica dei programmi. Studieremo la possibilità di regolare tale compromesso tra precisione ed efficienza al fine di consentirne l'applicabilità nel contesto della verifica.

Task PR-1.2: Domini numerici scalabili e precisi

Contrariamente a quanto detto per PR-1.1, per le proprietà numeriche esiste una vasta scelta di domini astratti con vari livelli di precisione ed efficienza ed implementazioni di buona qualità. Come per PR-1.1, i problemi di scalabilità impediscono l'adozione diretta dei domini più precisi, mentre considerazioni sulla precisione suggeriscono di evitare i domini più efficienti, se non quando le altre opzioni sono impraticabili. Studiremo combinazioni scalabili dei domini esistenti, cercando un mix adeguato tra operatori approssimati, widening e tecniche di partizionamento delle variabili

Task PR-2.1: Definizione di uno schema di analisi modulare e generico

I framework di analisi esistenti per i programmi imperativi impongono eccessive restrizioni al linguaggio analizzato, oppure mancano di una vera e propria modularità, impedendo di sfruttare le caratteristiche avanzate dei domini astratti più sofisticati. Basandosi su lavori esistenti, progetteremo un framework immune da tali inconvenienti.

Task PR-2.2: Annotazione di programmi

I linguaggi C, C++ e Java non consentono al programmatore o agli strumenti di supporto alla programmazione di annotare adeguatamente i programmi con informazioni (inferite in modo automatico o fornite dal programmatore) utili alla loro effettiva comprensione. Estensioni del linguaggio che permettano annotazioni dalla semantica chiara sono di sicuro interesse e saranno quindi oggetto di studio.

Task PR-2.3: Analisi di correttezza della manipolazione di stringhe

Gli errori nella manipolazione di stringhe C sono la causa di buona parte dei bachi che affliggono il software odierno e causano molte vulnerabilità di sistema. Studieremo approcci modulari nei quali il problema di dimostrare l'assenza di errori di manipolazione di stringhe viene tradotto in un problema per la cui soluzione sono sufficienti analisi che tracciano il valore di variabili intere, booleane e puntatore.

Task PR-2.4: Analisi di terminazione e complessità

Le analisi di terminazione e complessità di programmi C sono importanti argomenti di ricerca: ad esempio, svolgono un ruolo cruciale nella verifica delle componenti dei sistemi reattivi e dei device driver. Studieremo tecniche, anch'esse basate sull'analisi dei puntatori e delle variabili numeriche, per ottenere automaticamente dimostrazioni di terminazione o limitazioni della complessità di frammenti di programmi.

Testo inglese
Abstract interpretation [CC77,CC79] and its applications to the analysis of programs and other discrete dynamic systems have seen significant advances over the last 15 years. For example, ASTRÉE, a static analyzer firmly based on abstract interpretation theory, was able to prove the absence of any run-time errors in the primary flight control software of the Airbus A340 fly-by-wire system (132,000 lines of rather special C code). We are approaching the point where static analyzers based on abstract interpretation can be included in the development environments of mainstream imperative programs. Even though, at least initially, actors not involved in the production of critical software may not be willing to pay the cost of static-analysis-enhanced development environments, there is significant demand: for example, many companies in the automotive industry (including the italian FIAT and Magneti Marelli) have internal projects aimed at increasing software reliability via static analysis. Another sign of the times is provided by the GlobalGCC (GGCC) project within the ITEA programme (Information Technology for European Advancement), whose goal is to extend the GNU Compiler Collection (GCC) with program-wide static analysis techniques. The static analyses of GGCC will work at the GIMPLE internal representations level, and will thus be available to all the source languages and target systems supported by GCC. The GGCC consortium comprises, among others, Bertin Technologies, CEA-LIST, INRIA-Futurs, Mandriva, Telefonica I+D and UPM. While the time is ripe, there are still significant problems to be solved before this revolution in (critical) software development can happen on a larger scale. The Parma Unit has skills and experiences that may be useful to address some of them. In particular, in the present project, the Parma Unit will focus on

WP1: Abstract Interpretation Design
WP2: Static Analysis by Abstract Interpretation

with the tasks briefly summarized below.

Task PR-1.1: Abstract domains for points-to and aliasing analysis of C and Java programs

Most of the research on points-to and aliasing analysis focuses on optimized compilation and, for that application, only very quick analyses have been considered worthwhile. However, preliminary investigations show that the precision attainable by that analyses is insufficient for the purposes of program verification. We will investigate whether the complexity/precision trade-off can be solved in ways that are adequate for the latter applications.

Task PR-1.2: Scalable and precise numeric abstract domains

Differently from the case of PR-1.1, there is a wide choice of well-studied abstract domains with various degrees of precision and efficiency and high-quality implementations. Similarly to the case of PR-1.1, hard scalability problems (such as an exponential complexity) prevent the blind adoption of the most precise domains, whereas precision considerations suggest to adopt the most efficient domains only in restricted contexts, when all else fails. We will study scalable combinations of the existing domains by integrating approximate operators, widenings and partitionings of the variables' space.

Task PR-2.1: Definition of a modular and generic analysis framework

Existing analysis frameworks for imperative programs either put too strong restrictions on the analyzed languages or are not truly modular, preventing the exploitation of the characteristics of the more sophisticate analysis domains. Building on existing work, we will design a framework that tries to overcome all these difficulties.

Task PR-2.2: Annotation of programs

The languages that are the subject of our research, mainly C, C++ and Java, do not allow programmers and programming tools to annotate programs with definite knowledge

(whether it be automatically inferred or provided by the programmer), intentions and expectations. Language extensions that allow this kind of assertions with a well-defined semantics are interesting for a number of reason: we will work in this direction.

Task PR-2.3: Analysis of string cleanness

Errors in the manipulation of C strings are the source of a significant proportion of the bugs that plague today's software and are responsible of many vulnerabilities. We will study modular ways whereby the problem of ensuring that a program is clean from string manipulation errors is translated into a problem whose solution can be approximated by analyses tracking the values of integer, Boolean and pointer variables.

Task PR-2.4: Analysis of termination and complexity

Termination and complexity analyses of C programs are important topics: termination of the individual components of reactive systems and termination of device drivers are two examples where they play a crucial role. We will study techniques, again based on the analysis of pointers and numerical variables, to automatically prove the termination or bound the complexity of program fragments.

10 - Parole chiave

nº	Parola chiave (in italiano)	Parola chiave (in inglese)
1.	INTERPRETAZIONE ASTRATTA	ABSTRACT INTERPRETATION
2.	ANALISI STATICA	STATIC ANALYSIS
3.	DOMINI ASTRATTI	ABSTRACT DOMAINS
4.	ANALISI DI PROGRAMMI	PROGRAM ANALYSIS
5.	VERIFICA DI PROGRAMMI	PROGRAM VERIFICATION

11 - Stato dell'arte

Testo italiano
INTRODUZIONE

I difetti del software contribuiscono in maniera significativa al suo costo totale di sviluppo [RTI02], in particolare se scoperti dopo la fase di rilascio del prodotto. I difetti dalle conseguenze più costose sono quelli che introducono vulnerabilità di sistema. A tale proposito, l'analisi sistematica di migliaia di segnalazioni ha permesso al CERT, l'autorevole ente che opera nel campo della sicurezza informatica (http://www.cert.org/), di evidenziare come, alla base dei problemi di sicurezza più comuni, vi sia un numero limitato di cause. Il caso più frequente è quello di errori di programmazione quali overflow durante l'accesso ai buffer o nei calcoli sui tipi interi, errori nella formattazione di stringhe ed altri errori nella gestione della memoria. E' quindi necessario studiare, da un lato, metodologie di progettazione del software che aiutino a prevenire tali problemi, dall'altro, strategie di validazione del software che, tenendone conto, siano di supporto ad una loro efficace individuazione e rimozione. Naturalmente, un prodotto software immune dagli errori suddetti potrebbe comunque avere altri difetti che influiscono sulla sua funzionalità ad un più alto livello di astrazione. In ogni caso, l'assenza degli errori "di basso livello" è un prerequisito irrinunciabile, mancando il quale qualunque analisi svolta ai livelli superiori non fornirebbe alcuna reale garanzia. Per favorire le ricadute a livello pratico, l'unità di ricerca focalizzerà il suo lavoro sui linguaggi imperativi più diffusi, come C (con possibili

estensioni a C++) e Java.

WP1: PROGETTAZIONE DI INTERPRETAZIONI ASTRATTE

Task PR-1.1: Domini astratti per l'analisi dei puntatori e dell'aliasing in programmi C e Java

L'analisi dei puntatori e dell'aliasing per i linguaggi imperativi è un passo importante, a volte essenziale, per migliorare la precisione di numerose altre analisi. Nonostante la vasta letteratura sull'argomento, ad oggi è difficile ottenere risultati abbastanza precisi utilizzando una quantità ragionevole di risorse di calcolo [Ber07,Hin01]. Le analisi dei puntatori attualmente in uso per la compilazione ottimizzata, essendo indipendenti dal contesto e dal controllo di flusso, non forniscono un livello di precisione sufficiente ai fini della verifica dei programmi. D'altra parte, le analisi dipendenti da contesto e/o dal flusso proposte in letteratura non catturano tutti i costrutti di linguaggi come il C [Ber07], oppure sembrano essere state abbandonate [Deu94,EGH94].

Task PR-1.2: Domini numerici scalabili e precisi

I risultati preliminari sull'analisi di proprietà numeriche per programmi C reali hanno mostrato l'esistenza di problemi di precisione e scalabilità. Se l'uso indiscriminato del dominio dei poliedri convessi impedisce la scalabilità, d'altra parte il ricorso al dominio degli intervalli o anche degli ottagoni può a volte portare ad un numero inaccettabile di falsi allarmi (ad es., per l'analisi di string cleanness). La soluzione di tali problemi è quindi di primaria importanza.

WP2: ANALISI STATICHE MEDIANTE INTERPRETAZIONE ASTRATTA

Task PR-2.3: Analisi di correttezza della manipolazione di stringhe

La maggior parte delle vulnerabilità presenti nei programmi C ha origine da errori di manipolazione di stringhe; l'assenza di tali errori (string cleanness) è un prerequisito per la scrittura di programmi che abbiano un comportamento prevedibile e robusto agli attacchi. I programmi C sono particolarmente soggetti a tali errori per via dell'assenza di un tipo di dato stringa predefinito. Sebbene vi siano alcune analisi statiche per il rilevamento dei buffer overflow, molto lavoro rimane da fare per renderle efficaci [DRS03,Ell04].

Task PR-2.4: Analisi di terminazione e di complessità

L'analisi di terminazione per programmi C è un argomento di ricerca di primaria importanza. Il linguaggio C è ampiamente usato per lo sviluppo di dispositivi imbarcati, spesso caratterizzati da elementi di criticità (ad es., nell'automazione industriale) e per i quali la terminazione di ogni componente funzionale è una proprietà essenziale. Un dispositivo che mantiene una data interazione con l'ambiente può essere modellato da un ciclo composto da alcune fasi (lettura sensori, calcolo della reazione appropriata, invio dei comandi agli attuatori), per le quali la mancata terminazione sarebbe causa di malfunzionamenti gravi quanto un fallimento di sistema. Un altro esempio è fornito dai driver dei dispositivi hardware, la cui mancata terminazione può causare lo stallo del sistema operativo. E' anche per questa ragione che, al fine di semplificare la verifica automatica dei driver del proprio sistema operativo, la Microsoft ha iniziato un progetto di ricerca sull'analisi di terminazione dei programmi C [CPR06]. Assicurata la terminazione, il passo logico successivo è costituito dall'analisi di complessità, che consente di garantire, ad esempio, il rispetto delle scadenze temporali da parte dei sistemi software.

Testo inglese
INTRODUCTION

It is widely acknowledged that software defects significantly contribute to the high costs of software development [RTI02]. Moreover, defects that are detected only after the product has been released have an higher cost, both for producers and consumers. Software defects whose consequences are particularly expensive are those that introduce system vulnerabilities. After the thorough analysis of thousands of reports about computer systems vulnerabilities, CERT, the authoritative center for computer security (http://www.cert.org/), concluded that most of the actually exploited security problems are due to a limited number

of causes. The most frequent ones originate from programming mistakes: out of bounds accesses to buffers, overflows in operations on native integers, string manipulation errors and other errors related to memory management. It is therefore important to study, on the one hand, software design methodologies that can help preventing these problems and, on the other hand, software validation strategies that, being aware of these problems, do help in their detection and removal. In most cases, of course, a software product unaffected by the aforementioned problems can still have defects that impact the user, as other and more abstract properties must be considered to uncover higher-level flaws. However, the absence of "low level" defects is clearly a prerequisite to the reliability of any higher-level analysis. In order to have an impact on this state of affairs, the unit focuses its research on mainstream imperative languages such as C (with possible extensions to C++) and Java.

WP1: ABSTRACT INTERPRETATION DESIGN

Task PR-1.1: Abstract domains for points-to and aliasing analysis of C and Java programs

Aliasing/pointer analysis in imperative languages is a useful, sometimes essential step to improve the precision of several other analyses. Despite the vast literature on the subject, it is currently very difficult to obtain reasonably precise results while using a reasonable amount of resources for the analysis [Ber07,Hin01]. Flow- and context-insensitive pointer analyses in use for optimized compilation, while being able to scale to millions of lines of code, do not provide enough precision for the purposes of program verification. On the other hand, flow- and/or context sensitive analyses proposed in the literature either are unable to capture all the constructs of languages such as C [Ber07], or seem to have been abandoned for some (unknown) reason [Deu94,EGH94].

Task PR-1.2: Scalable and precise numeric abstract domains

Preliminary results on the analysis of numerical properties of real C programs clearly showed that there are problems in both scalability and precision of the analysis. A blind use of the domain of convex polyhedra implies non scalability, while resorting to intervals or even octagons sometimes does not allow to acceptably reduce the number of false alarms (e.g., for string cleanness analysis). Solving these problems of scalability and precision is thus a priority.

WP2: STATIC ANALYSIS BY ABSTRACT INTERPRETATION

Task PR-2.3: Analysis of string cleanness

Most of the vulnerabilities occurring in programs in C stem from string manipulation errors; the absence of such errors, i.e., the string cleanness, is a prerequisite for secure programs that have a predictable behavior and are robust against malicious attacks. Programs in C are particularly prone to such errors due to the lack of a native string data type. Static analysis techniques for the detection of buffer overflows have been proposed, but work remains to be done in order to make them effective [DRS03,Ell04].

Task PR-2.4: Analysis of termination and complexity

Termination analysis for C programs is a main topic in current static analysis research. The C language is widely used to develop embedded devices, often including critical aspects (in industrial control, automotive industry, etc.), where the proper termination behavior of each functional component is essential. Imagine a device whose goal is to maintain a certain type of interaction with the environment. This can be conceived as an infinite loop consisting of several phases (read sensors, compute a proper reaction, send commands to actuators), whose termination failure would cause malfunctions as bad as classical system failures. Another example comes from device drivers, where a termination failure can cause the whole operating system to hang. For this reason, Microsoft started a research project to study automatic termination analysis of C programs: drivers for the Microsoft operating systems are developed by hundreds of hardware producers and their automatic verification is crucial to the US software giant [CPR06]. Once termination is guaranteed, complexity analysis is the logical step forward to ensure, e.g., the hard deadlines are met by software systems.

12 Riferimenti bibliografici

[BDH+07] R. Bagnara, K. Dobson, P. M. Hill, M. Mundell, and E. Zaffanella. Grids: A domain for analyzing the distribution of numerical values. In Proceedings of LOPSTR'06, vol. 4407 of LNCS, pp. 219--235, 2007.

[Ber07] D. Berlin. Points-to analysis and the real world. Retrieved October 16, 2007, from http://www.dberlin.org/blog/2007/03/18/points-to-analysis-and-the-real-world/, 2007.

[BHMZ05] R. Bagnara, P. M. Hill, E. Mazzi, and E. Zaffanella. Widening operators for weakly-relational numeric abstractions. In Proceedings of SAS'05, vol. 3672 of LNCS, pp. 3--18, 2005.

[BHRZ05] R. Bagnara, P. M. Hill, E. Ricci, and E. Zaffanella. Precise widening operators for convex polyhedra. Science of Computer Programming, 58(1--2):28--56, 2005.

[BHZ05] R. Bagnara, P. M. Hill, and E. Zaffanella. Not necessarily closed convex polyhedra and the double description method. Formal Aspects of Computing, 17(2):222--257, 2005.

[BHZ06] R. Bagnara, P. M. Hill, and E. Zaffanella. Widening operators for powerset domains. Software Tools for Technology Transfer, 8(4/5):449--466, 2006.

[BHZ07] R. Bagnara, P. M. Hill, and E. Zaffanella. An improved tight closure algorithm for integer octagonal constraints. Quaderno 467, Dipartimento di Matematica, Università di Parma, 2007. Available at http://www.cs.unipr.it/Publications/.

[BHZ08] R. Bagnara, P. M. Hill, and E. Zaffanella. The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. Science of Computer Programming, 2008. To appear.

[CC76] P. Cousot and R. Cousot. Static determination of dynamic properties of programs. In Proceedings of the Second International Symposium on Programming, pp. 106--130, 1976. Dunod, Paris, France.

[CC77] 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 POPL'77, pp. 238--252, 1977.

[CC92a] P. Cousot and R. Cousot. Comparing the Galois connection and widening/narrowing approaches to abstract interpretation. In Proceedings of PLILP'92, vol. 631 of LNCS, pp. 269--295, 1992.

[CC92b] P. Cousot and R. Cousot. Inductive definitions, semantics and abstract interpretation. In Proceedings of POPL'92, pp. 83--94, 1992.

[CCF^+05] P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. The ASTRÉE analyzer. In Proceedings of ESOP'05, vol. 3444 of LNCS, pp. 21--30, 2005.

[CFW96] A. Cortesi, G. Filé, and W. Winsborough. Optimal groundness analysis using propositional logic. Journal of Logic Programming, 27(2):137--167, 1996.

[CPR06] B. Cook, A. Podelski, and A. Rybalchenko. Termination proofs for systems code. In Proceedings of PLDI'06, pp. 415--426, 2006.

[Deu94] A. Deutsch. Interprocedural may-alias analysis for pointers: Beyond k-limiting. In [PLD94], pp. 230--241.

[DRS01] N. Dor, M. Rodeh, and S. Sagiv. Cleanness checking of string manipulations in C programs via integer analysis. In Proceedings of SAS'01, vol. 2126 of LNCS, pp. 194--212,

2001.

[DRS03] N. Dor, M. Rodeh, and S. Sagiv. CSSV: Towards a realistic tool for statically detecting all buffer overflows in C. In Proceedings of PLDI'03, pp. 155--167, 2003.

[EGH94] M. Emami, R. Ghiya, and L. J. Hendren. Context-sensitive interprocedural points-to analysis in the presence of function pointers. In [PLD94], pp. 242--256.

[Ell04] R. Ellenbogen. Fully automatic verification of absence of errors via interprocedural integer analysis. Master's thesis, School of Computer Science, Tel-Aviv University, December 2004.

[Hin01] M. Hind. Pointer analysis: Haven't we solved this problem yet? In Proceedings of PASTE'01, pp. 54--61, 2001.

[HPB00] M. V. Hermenegildo, G. Puebla, and F. Bueno. An assertion language for constraint logic programs. In Analysis and Visualization Tools for Constraint Programming, vol. 1870 of LNCS, pp. 23--61. Springer, 2000.

[HPR97] 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.

[Int99] International Organization for Standardization. ISOIEC 9899:1999: Programming Languages --- C. ISO, Geneva, 1999.

[PLD94] Proceedings of the PLDI'94, vol. 29 of ACM SIGPLAN Notices, Orlando, Florida, 1994. Association for Computing Machinery.

[RTI02] RTI. The economic impacts of inadequate infrastructure for software testing. Planning Report 02-3, NIST, Gaithersburg, MD, 2002.

[SHP06] F. Spoto, P. M. Hill, and É. Payet. Path-length analysis for object-oriented programs. In EAAI'06, Vienna, Austria, 2006. Available at http://profs.sci.univr.it/~spoto/papers.html.

[Sch98] D. A. Schmidt. Trace-based abstract interpretation of operational semantics. LISP and Symbolic Computation, 10(3):237--271, 1998.

[SS05] S. Secci and F. Spoto. Pair-sharing analysis of object-oriented programs. In Proceedings of SAS'05, vol. 3672 of LNCS, pp. 320--335, 2005.

13 - Descrizione del programma e dei compiti dell'Unità di Ricerca

Testo italiano
INTRODUZIONE

L'obiettivo del progetto è investigare sull'applicazione di tecniche di interpretazione astratta allo scenario più realistico: applicazioni scritte da comuni programmatori in linguaggi di largo uso come C, C++ e Java. Per poter influire positivamente su tale scenario, ogni soluzione proposta deve essere integrabile con il software, gli ambienti e le pratiche di programmazione esistenti nonché con le conoscenze comunemente possedute dai programmatori.

WP1: PROGETTAZIONE DI INTERPRETAZIONI ASTRATTE

Task PR-1.1: Domini astratti per l'analisi dei puntatori e dell'aliasing in programmi C e Java

Il dominio proposto in [EGH94] per l'analisi dei puntatori sullo stack nei programmi C consente di definire un'analisi, dipendente dal contesto e dal controllo di flusso, in grado di catturare sia la possibilità sia la presenza certa di aliasing, incluse alcune proprietà dei puntatori ad array e struct. Un ragionamento preciso sulla correttezza e precisione dell'approccio di [EGH94] è complicato dalla mancanza di una specifica formale. Vogliamo quindi riprogettare ed implementare una variante di questo dominio astratto, mettendo una cura particolare nel chiarire tutte le ipotesi che stanno alla base delle dimostrazioni di correttezza dell'analisi e prendendo come semantica concreta di riferimento quella specificata dallo standard del linguaggio C [Int99], possibilmente arricchita dalla formalizzazione di alcuni comportamenti dipendenti dall'implementazione. Questo consentirà una migliore separazione tra le scelte progettuali che sono imposte dalla natura conservativa dell'analisi, rispetto a quelle che sono adottate per semplificarne l'implementazione o per migliorarne l'efficienza e la scalabilità; inoltre, una specifica più accurata permetterà lo studio formale delle dipendenze e possibili sinergie tra l'analisi dei puntatori ed altre analisi di proprietà di safety.

Per l'analisi di aliasing nei programmi Java, studieremo la definizione di un framework concettuale comune all'interno del quale possano essere inquadrate la maggior parte delle proposte che si trovano in letteratura. In ciò prendiamo spunto dalla situazione che si ha nel campo della programamzione logica, dove quasi tutte le analisi sono ottenute come astrazione della stessa semantica, basata sull'insieme delle parti delle sostituzioni con gli operatori di ridenominazione, proiezione ed unificazione [CFW96]. In particolare, pensiamo di riformulare molte delle note varianti di analisi di aliasing e di shape come astrazioni di una stessa semantica denotazionale, sensibile al flusso di esecuzione e al contesto. Ciò consentirà il confronto formale tra i principali punti di forza e debolezza dei vari approcci, nonché la possibilità di separare il problema della rappresentazione delle informazioni dei domini astratti dalla specifica dei corrispondenti algoritmi per il calcolo dell'analisi. Basando gli studi su di una solida semantica denotazionale, sarà inoltre possibile investigare nuovi domini ed operatori astratti. Per esempio, [SS05] introduce un'analisi di pair-sharing che si ispira ad un noto dominio astratto sviluppato per l'analisi di programmi logici. Pur non essendo particolarmente precisa, tale analisi ha buone doti di scalabilità. La precisione potrebbe migliorare combinando il dominio con altre proprietà, quali la costanza o altri tipi di informazione strutturale. Vorremmo inoltre studiare l'applicabilità al caso in esame di altre proprietà e domini sviluppati nel contesto dei linguaggi logici, quali la linearità e set-sharing.

Task PR-1.2: Domini numerici scalabili e precisi

Esiste un'ampia letteratura sul progetto di domini astratti numerici: da semplici proprietà non relazionali (ad esempio intervalli e congruenze non relazionali) a domini che codificano proprietà relazionali ben più complesse (insiemi di poliedri convessi, disuguaglianze polinomiali ecc.). La questione è come ci si possa avvicinare all'efficienza e scalabilità dei domini più semplici ottenendo una precisione confrontabile con quella dei domini più complessi. La maggior parte degli studi in corso, nella ricerca di compromessi vantaggiosi, esplora lo spazio esistente tra gli estremi sopra menzionati. In questo progetto, affronteremo il problema:

1) progettando operatori approssimati che consentano di cedere precisione, in modo controllato e in contesti determinati, in cambio di efficienza (nuove euristiche di widening [CC76,CC77,CC92a], versioni non ottimali degli operatori di transizione di stato, partizionamenti dello spazio delle variabili [CCF+05], ...);

2) progettando nuove combinazioni dei domini esistenti e dei corrispondenti operatori per integrarli efficacemente.

Per questo lavoro, l'esperienza maturata dall'unità di Parma sul progetto di domini numerici e operatori di widening sarà molto utile [BDH+07,BHZ05,BHMZ05,BHRZ05,BHZ06]. L'implementazione e la sperimentazione avverranno nel contesto della Parma Polyhedra Library (PPL) [BHZ08].

E' rilevante il fatto che ricercatori dell'unità di Parma prenderanno attivamente parte al progetto GlobalGCC (GGCC) del programma ITEA (Information Technology for European Advancement), il cui obiettivo è quello di estendere la GNU Compiler Collection (GCC) con tecniche di analisi statica globale. Le analisi statiche di GGCC saranno al livello della rappresentazione interna GIMPLE, e saranno così utilizzabili per tutti i linguaggi sorgente

ed i sistemi target supportati da GCC. Le analisi di proprietà numeriche di GGCC saranno basate sulla PPL. Il consorzio GGCC (che comprende, tra gli altri, Bertin Technologies, CEA-LIST, INRIA-Futurs, Mandriva, Telefonica I+D e UPM) sta lavorando ad una modifica del progetto che prevede l'integrazione dell'Università di Parma. La sinergia tra GGCC e il presente progetto è chiara: sui problemi di scalabilità e precisione l'unità di Parma potrà lavorare nello scenario più realistico possibile.

WP2: ANALISI STATICA MEDIANTE INTERPRETAZIONE ASTRATTA

Task PR-2.1: Definizione di uno schema di analisi modulare e generico

Studieremo i problemi collegati al progetto e all'implementazione di uno schema generico di analizzatori statici per i linguaggi imperativi di largo uso. Lo schema sarà istanziato su analisi dei puntatori e dei valori delle variabili numeriche e booleane (tali variabili possono essere del programma o fittizie per rappresentare, ad esempio, la dimensione dei buffer allocati o la presenza dei terminatori di stringa). Le innovazioni principali che ci interessa introdurre sono la flessibilità e la generalità, sia in termini delle proprietà che in termini del linguaggio analizzato. Da una parte, studieremo tecniche di analisi generale che, seppure indipendenti dalle specifiche proprietà, consentano di sfruttare appieno l'informazione fornita dai domini più sofisticati (ad esempio quelli relazionali). D'altro canto, faremo sì che tutte le caratteristiche "critiche" degli odierni linguaggi imperativi (come le eccezioni, i puntatori e i meccanismi di controllo non strutturati) siano convenientemente approssimate dalla semantica astratta che formalizza il processo di analisi. Crediamo che, a tale scopo, gli approcci basati su regole [CC92b,Sch98] siano particolarmente promettenti.

Task PR-2.2: Annotazione di programmi

C, C++ e Java forniscono solo un supporto rudimentale per la specifica delle conoscenze, intenzioni e attese degli sviluppatori. Le possibilità di specificare queste insieme al codice vero e proprio può essere vantaggiosamente sfruttata per realizzare strumenti di sviluppo migliori. Il nostro interesse a dotare di questa possibilità i linguaggi di programmazione più usati ha anche altre motivazioni: la possibilità di annotare i sorgenti con vari tipi di asserzioni consente di modularizzare il progetto di analizzatori (i quali potrebbero così essere facilmente combinati in cascata) e di acquisire fiducia nella loro implementazione (utilizzando diversi prototipi, l'uno per controllare l'altro). Studieremo la specifica di un linguaggio di asserzioni, nello stile di [HPB00], che dovrebbe consentire al programmatore di codificare nel programma le condizioni logiche che garantiscono che lo stesso sia immune da una certa classe di errori e/o rispetti i limiti imposti dalle politiche di allocazione delle risorse. Il linguaggio di asserzioni sarà estensibile per adattarsi a specifici domini applicativi e non sarà limitato a teorie decidibili (come quelle usate nei sistemi basati sul controllo dei tipi), consentendo così di combinare controlli statici e dinamici delle proprietà. La disponibilità di un linguaggio di asserzioni espressivo sarà sfruttata anche per abilitare la comunicazione dall'analizzatore al programmatore: lo strumento di analisi sarà in grado di annotare automaticamente il programma con una selezione delle proprietà scoperte dall'analisi. L'obiettivo è quello di ottenere un ambiente di sviluppo fortemente integrato, dove il feedback derivante dal confronto delle proprietà attese e quelle calcolate potrà dare vita ad un processo di raffinamento sinergico, con benefici sia per lo sviluppo che per il debugging delle applicazioni. Ad esempio:

1) asserzioni ritenute importanti dai programmatori possono essere marcate in modo che l'analizzatore le usi come "suggerimenti" (abilitando, tra l'altro, quei meccanismi, come il widening-up-to [HPR97], demandati a preservare la precisione dell'analisi in quello specifico contesto);

2) l'analizzatore, ove possibile, può marcare come verificate o refutate quelle asserzioni che ha potuto mostrare vere o false, eventualmente semplificandone altre.

Task PR-2.3: Analisi di correttezza della manipolazione di stringhe

Studieremo un approccio trasformazionale al problema di assicurare la correttezza delle operazioni sulle stringhe nei programmi C (string cleanness). L'approccio consiste nel trasformare un programma, P, che usa stringhe in un'altro, P', dove gli array di caratteri e i puntatori sono sostituiti da strutture ad hoc e dove sono state inserite delle asserzioni. La trasformazione avrà le seguenti proprietà: (1) sarà completamente automatica; (2) se P' non

può violare nessuna delle sue asserzioni, allora ogni esecuzione di *P* sarà esente da errori nella manipolazione delle stringhe. Le stringhe saranno sostituite da strutture che astraggono la loro semantica per mezzo di variabili intere e booleane (per modellare, ad esempio, la dimensione del buffer allocato, la posizione di un puntatore all'interno del buffer, e la presenza dei terminatori). Le asserzioni modellano la correttezza delle operazioni sulle stringhe: ad esempio, prescrivono che la posizione di un puntatore che accede ad un buffer sia non negativa e non superi la dimensione del buffer. Il problema della string cleanness è così trasformato in un problema che richiede ragionamento su interi, booleani e puntatori: le tecniche di interpretazione astratta e analisi statica sviluppate in questo progetto consentiranno, sperabilmente, di avere un numero basso di falsi positivi. L'idea ha origine dai lavori di Dor, Ellenbogen, Rodeh e Sagiv [DRS01, DRS03, Ell04], ma nella loro soluzione ogni stadio del processo di verifica (pre-processing, trasformazione, analisi dei puntatori, analisi degli interi) è fortemente dipendente dagli altri e li vincola in modo severo. Questa è una delle ragioni per le quali iCSSV (interprocedural C String Static Verifier) ha avuto un successo solo parziale [Ell04]. L'approccio che abbiamo in mente è più modulare e ci consentirà di sfruttare gli sviluppi delle tecniche di analisi dei puntatori e degli interi. Inoltre, diversamente dal caso di iCSSV, saremo in grado di usare un'analisi combinata di puntatori e interi, forse riducendo il numero di falsi positivi. Va osservato che, sia nel nostro approccio che in quello di [DRS01, DRS03, Ell04], ogni puntatore e buffer richiede diverse variabili (booleane e intere): certamente ciò porterà ad esacerbare il conflitto tra complessità e precisione, rendendo quindi il lavoro del WP1 ancor più importante.

Task PR-2.4: Analisi di terminazione e complessità

L'analisi di terminazione per codice C è complessa, a causa di diversi fattori, che la differenziano significativamente (ma non completamente) dall'analisi di terminazione per linguaggi logici, per la quale esistono già tecniche piuttosto sofisticate. In particolare:

- C è un linguaggio imperativo nel quale il processo di calcolo evolve per mezzo di assegnamenti distruttivi. E' quindi fondamentale poter prevedere, con una buona approssimazione, l'effetto che un assegnamento distruttivo ad un campo di una struttura dati potrà avere sulle altre strutture dati. A tal fine servono informazioni sui puntatori e dunque questa ricerca ha una naturale relazione con lo sviluppo di un'analisi dei puntatori per il C, altro obiettivo dell'unità di Parma.

- In C i costrutti che possono dare luogo a non terminazione sono i comandi iterativi (while, do-while, for), le chiamate di funzione mutuamente ricorsive, e le istruzioni goto. A quanto sappiamo, nessuno degli approcci esistenti (sia per il linguaggio C che per altri linguaggi) ha mai preso in considerazione tutte queste possibilità contemporaneamente.

Intendiamo strutturare le nostre analisi di terminazione per C nel modo seguente:

1) **Ricostruzione dei cicli nel controllo.** Vengono identificate le sorgenti di possibile non terminazione: cicli espliciti, chiamate di funzione ricorsive e salti all'indietro.

2) **Ricostruzione dei cicli nei dati.** Sfruttando i risultati dell'analisi dei puntatori viene analizzata la ciclicità delle strutture dati. Questa informazione serve nel successivo passo 3.

3) **Generazione di vincoli:** a) sui valori delle variabili numeriche; b) sulla lunghezza massimale di un percorso di puntatori seguito a partire da ciascuna variabile (path-length analysis). Pensiamo di adattare quest'ultima analisi, già formalizzata da Spoto (unità di Verona) per il linguaggio Java [SHP06], al linguaggio C.

4) **Sintesi di funzioni di ranking.** I vincoli così generati vengono elaborati con tecniche automatiche che cercano di identificare una funzione di misura (norma) che, oltre ad essere limitata inferiormente, ad ogni iterazione di un ciclo (comunque realizzato) decresca di una quantità limitata inferiormente da una costante positiva. Tali proprietà garantiscono che la funzione trovata sia una funzione di terminazione per il programma.

Testo inglese
INTRODUCTION

Our aim is to investigate the issues related to the application of abstract interpretation

techniques to the most realistic scenario: applications written by ordinary programmers in widespread languages such as C, C++ and Java. In order to have a positive impact on such a scenario, any proposed solution should be amenable to integration with existing software, development environments and practices as well as developers' knowledge.

WP1: ABSTRACT INTERPRETATION DESIGN

Task PR-1.1: Abstract domains for points-to and aliasing analysis of C and Java programs

The approach put forward in [EGH94] for stack pointer analysis of C programs allows for the definition of a flow- and context-sensitive analysis that is able to capture both possible and definite aliasing, including some information on pointers to elements of aggregate data types (arrays and structs). The specification in [EGH94] is rather informal and, hence, a precise reasoning on the correctness and precision of the approach is difficult. We plan to re-design, properly formalize and implement a variant of this domain, putting particular care in disambiguating all the hypotheses underlying the correctness arguments, while taking as the reference concrete semantics the one specified by the C language standard [Int99], possibly enriched by implementation-specific behaviors. This will allow for a better separation of the design choices that are mandated by the conservative nature of the analysis, from those that are adopted, e.g., to simplify the implementation of the analysis, or to improve its efficiency and scalability; moreover, such a proper formalization will allow for a correct investigation of the coupling and synergies between the pointer analysis and other safety properties.

For the analysis of aliasing in Java programs, we will try to identify a common conceptual framework where most of the proposals in the literature may be cast. This is inspired by similar experiences put forward in the field of logic programming. Here, although some details may differ, most of the analyses are based essentially on the same concrete semantics, namely, the domain of power set of substitutions endowed with operators of renaming, projection and unification [CFW96]. Therefore, we plan to reformulate the most common variants of aliasing and shape analysis as abstractions of a common denotational, flow and context sensitive semantics. We expect, as a result, the ability to formally evaluate the relative strength and weakness of the various approaches, as well as the ability to decouple the abstract domains representing the dynamic data structures from the algorithms computing the abstract behavior. We also think that a solid denotational semantics at the foundation may help in investigating new abstract domains and operators. For example [SS05] introduces a pair-sharing analysis, based on ideas borrowed from analyses developed for logic programs. Although pair-sharing analysis is not very precise, it has good scalability properties. Its precision could be improved by combining it with other kinds of properties, such as constancy or some structural information. Moreover, we could investigate the relevance to aliasing analysis of other well known domains for logic programming, such as linearity or set-sharing.

Task PR-1.2: Scalable and precise numeric abstract domains

There is a vast body of literature concerning the design of numeric abstract domains, ranging from simple non-relational properties (e.g., intervals or non-relational congruences) to sophisticated relational properties (e.g., power-sets of convex polyhedra or polynomial inequalities). Clearly, issues arise when the efficiency and scalability of the simpler proposals has to be paired with the precision of the more sophisticated ones, so that most of the current research, looking for appropriate tradeoffs, lies in between of the two extremes mentioned above. We will attack this problem by:

1) designing approximated operators on the existing domains, so as to be able to trade precision for efficiency in a controlled way and in selected contexts (e.g., new widening heuristics [CC76,CC77,CC92a], non-optimal versions of state transition operators, partitioning of the variables' space [CCF+05]);

2) designing new combinations of existing domains and of the corresponding operators for their effective integration.

For this work, the experience of the Parma Unit on the design of numerical domains and widening operators will be very useful [BDH+07,BHZ05,BHMZ05,BHRZ05,BHZ06]. Implementation and experimentation will take advantage of the Parma Polyhedra Library

[BHZ08].

It is worth mentioning that researchers of the Parma Unit will actively take part into the GlobalGCC (GGCC) project, within the ITEA programme (Information Technology for European Advancement), whose objective is to extend the GNU Compiler Collection (GCC) with program-wide static analysis techniques. The static analysis of GGCC will work at the GIMPLE internal representations level, and will thus be usable for all the source languages and target systems supported by GCC. The numeric analysis engine of GGCC will be the Parma Polyhedra Library. The GGCC consortium (which comprises, among others, Bertin Technologies, CEA-LIST, INRIA-Futurs, Mandriva, Telefonica I+D and UPM) is working on a project modification that foresee the integration of the University of Parma. The synergy between GGCC and the present project is clear: concerning the aforementioned scalability and precision problems the Parma Unit will be able to work in the most realistic scenario.

WP2: STATIC ANALYSIS BY ABSTRACT INTERPRETATION

Task PR-2.1: Definition of a modular and generic analysis framework

We will study the problems related to the design and implementation of a generic framework for static analysis of imperative languages in widespread use, which will be later instantiated to obtain analyses to approximate pointer aliasing and the values of numeric and Boolean variables (such variables can be explicit program variables or fictitious ones expressing, e.g., the size of allocated buffers or the presence of string terminators). The main innovations of the framework we wish to build are flexibility and generality, both in terms of the analyzed properties and of the analyzed language. On the one hand, we will study general analysis techniques that, while independent from the specific analyzed properties, still allow to fully take advantage of the information provided by more sophisticate domains (e.g., those encoding relational information to various extents). On the other hand, we will make sure all critical features of today's imperative programming languages (such as exceptions, pointers and non-structured control-flow mechanisms) can be conveniently captured by the abstract semantics that formalizes the analysis process. We believe that rule-based approaches [CC92b,Sch98] are good candidates for the achievement of these goals.

Task PR-2.2: Annotation of programs

The real programming languages that we target only provide rudimentary support for the specification of developers' knowledge, intentions and expectations. The ability to specify these alongside the real code can be fruitfully exploited to realize better development models (that, we believe, would be fully justified at least for the development of mission-critical software). However, our interest in developing this possibility for mainstream languages has also other motivations: the ability of annotating the program sources with various kinds of assertions allows, among other things, to modularize the design of analyzers (which could then be easily cascaded) and to gain confidence in their implementation (by running different prototypes one against the other). We will investigate the specification of a suitable assertion language, in the style of [HPB00], that should enable the programmer to encode in the program itself the logic conditions that guarantee that the program (fragment) is unaffected by some given class of errors and/or it respects the bounds imposed by the resource allocation policies. The assertion language will be extensible to adapt to the specific application domain; also, it will not be constrained to decidable theories (such as those used in type-checking based systems), allowing for the mixing of static and dynamic property checking. The availability of an expressive assertion language will also be exploited to enable communication the other way around, from the analyzer to the programmer: the analysis tool will be able to automatically annotate the program with a selection of the properties it could discover during the analysis. The final aim is a highly integrated development environment, where feedback coming from the comparison of expected and computed properties can start a synergistic refinement process, benefiting both the application development and debugging phases. For instance:

1) assertions deemed important by the programmers (those written by themselves or some of the ones automatically obtained from the analysis) can be marked so that the analyzer may use them as "hints" (enabling those mechanisms aimed at preserving the precision of the analysis in that specific context, like the widening-up-to [HPR97]);

2) the analyzer, where possible, marks as verified or refuted those assertions it can prove to be true or false; it may also simplify assertions that it could not automatically verify.

Task PR-2.3: Analysis of string cleanness

We will study a transformational approach to the problem of ensuring cleanness of string operations in C programs. The approach consists in transforming C program *P* using strings into another program, *P*', where arrays of characters and pointers are substituted by ad hoc structures and where assertions are included. The transformation will have the following properties: (1) it will be fully automatic; (2) if *P*' cannot violate any of its assertions, then all the executions of *P* do not result in any string manipulation errors. Strings will be replaced by structures that abstract their semantics by means of integer variables (modeling the size of an allocated buffer and the position of a pointer within the buffer) and Boolean variables (to model, e.g., the presence of terminators). The assertions model string cleanness: for instance they prescribe that the position of a pointer accessing a buffer is nonnegative and does not exceed the size of the buffer. The string cleanness problem is thus transformed into a problem the requires reasoning on integers, Booleans and pointers and the abstract interpretation and static analysis techniques developed in this project will hopefully allow to have only a small number of false alarms. This idea originates from the works of Dor, Ellenbogen, Rodeh and Sagiv [DRS01, DRS03, Ell04]. However, in their solution every stage of the verification process (pre-processing, transformation, pointer analysis, integer analysis) is heavily dependent on the others and severely constrains them. This is one of the reasons why iCSSV (interprocedural C String Static Verifier) was only partly successful [Ell04]. The approach we aim at is more modular and will enable us to easily put at work the advances in pointer and integer analyses. Moreover, differently from what happens in iCSSV, we will be able to use a combined pointer-and-integer analysis, possibly reducing the number of false alarm. It must be observed that, both in our approach and in the one of [DRS01, DRS03, Ell04], each pointer and buffer requires several (integer and Boolean) variables: surely this will exacerbate the complexity/precision tradeoff and thus makes the work in WP1 on scalability even more important. Another reason of the moderate success of iCSSV was in fact the scalability of integer analysis (as well as sub-optimal widening strategies).

Task PR-2.4: Analysis of termination and complexity

Termination analysis for C code is complex due to several factors, which make it different (although not completely) from the termination analysis for logic programs, already addressed by rather refined techniques. In particular:

- C is an imperative languages in which computation evolves through destructive assignments. It is therefore important to determine, with good approximation, the effect that a destructive assignment to a field of a data structure can have on the other data structures. To this aim, points-to information is crucial and hence this part of the research has a natural relation with the development of a points-to analyzer for C, whose study and development are also addressed by Parma Unit.

- In C, the constructs that can cause non-termination are iterative commands (while, do-while, for), mutually recursive function calls and goto statements. As far as we know, nobody ever addressed all these problems simultaneously (in C or other languages).

We wish to structure our termination analyses for C as follows:

1) **Identify control loops.** All the sources of possible non-termination in a C program are identified: explicit loops, recursive function calls and backwards jumps.

2) **Identify loops in data.** By exploiting the information provided by the pointer analysis, possible loops in data structures are examined. This information is used in step 3 below.

3) **Generation of constraints**: a) on the values of numeric variables; b) on the maximal length of a chain of pointers followed starting from each variable (path-length analysis). We plan to adapt the latter analysis, also formalized by Spoto (Verona Unit) for the case of Java [SHP06], to the C language.

4) **Synthesis of ranking functions.** The generated constraints are elaborated through automatic techniques so as to try and find a measure function (norm) which, besides being

bounded from below, during each iteration of a looping construct (of any of the kind mentioned in point 1 above) decreases by a quantity limited from below by a positive constant. These properties guarantee that the synthesized function is a termination function for the program.

14 - Descrizione delle attrezzature già disponibili ed utilizzabili per la ricerca proposta

Testo italiano

Nessuna

Testo inglese

Nessuna

15 - Descrizione delle Grandi attrezzature da acquisire (GA)

Testo italiano

Nessuna

Testo inglese

Nessuna

16 - Mesi persona complessivi dedicati al Progetto

		Numero	Impegno 1° anno	Impegno 2° anno	Totale mesi persona
Componenti della sede dell'Unità di Ricerca		2	16	12	28
Componenti di altre Università/Enti vigilati		1	8	6	14
Titolari di assegni di ricerca		0			
Titolari di borse	Dottorato	0			
	Post-dottorato	0			
	Scuola di Specializzazione	0			
Personale a	Assegnisti	0			
	Borsisti	0			
	Altre tipologie	1	3	9	12

contratto					
Dottorati a carico del PRIN da destinare a questo specifico progetto		0	0	0	0
Altro personale		0			
TOTALE		**4**	**27**	**27**	**54**

17 - Costo complessivo del Progetto dell'Unità articolato per voci

Voce di spesa	Spesa in Euro	Descrizione dettagliata (in italiano)	Descrizione dettagliata (in inglese)
Materiale inventariabile	4.500	Computer e accessori, software, libri e periodici per la ricerca.	Computers and accessories, software, research books and journals.
Grandi Attrezzature	0		
Materiale di consumo e funzionamento	4.500	Quota forfetaria del 10% per le spese generali.	Management costs (10% of the overall cost).
Spese per calcolo ed elaborazione dati			
Personale a contratto	20.000	Contratti per attività di ricerca.	Contracts for research activities.
Dottorati a carico del PRIN da destinare a questo specifico progetto	0		
Servizi esterni			
Missioni	13.500	Missioni in Italia ed all'estero in altre sedi del progetto, in istituti di ricerca, per la partecipazione a workshop, convegni e scuole di ricerca. Costi per l'invito di ospiti presso la sede.	Travel costs for visits and attending workshop and conferences. Costs for supporting guest visitors.
Pubblicazioni			
Partecipazione / Organizzazione convegni	2.500	Costi per iscrizioni a workshop e convegni. Costi per l'organizzazione di workshop e convegni.	Conference and school registration fees. Costs for organizing workshops and conferences.
Altro			
TOTALE	**45.000**		

18 - Prospetto finanziario dell'Unità di Ricerca

Voce di spesa	Importo in Euro
a.1) finanziamenti diretti, disponibili da parte di Università/Enti vigilati di appartenenza dei ricercatori dell'unità operativa	
a.2) finanziamenti diretti acquisibili con certezza da parte di Università/Enti vigilati di appartenenza dei ricercatori dell'unità operativa	13.500
b.1) finanziamenti diretti disponibili messi a disposizione da parte di soggetti esterni	
b.2) finanziamenti diretti acquisibili con certezza, messi a disposizione da parte di soggetti esterni	
c) cofinanziamento richiesto al MUR	31.500
Totale	45.000

19 - Certifico la dichiarata disponibilità e l'utilizzabilità dei finanziamenti a.1) a.2) b.1) b.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 e la loro elaborazione necessaria alle valutazioni; D. Lgs, 196 del 30.6.2003 sulla "Tutela dei dati personali")

Firma _____ Data 23/10/2007 ore 22:17

ALLEGATO

Curricula scientifici dei componenti il gruppo di ricerca

Testo italiano

1. **AMATO Gianluca**

Curriculum:

Il dott. Gianluca Amato si laurea in Scienze dell'Informazione presso l'Università di Pisa nell'anno 1996 e consegue il dottorato di ricerca in Informatica presso la stessa università nel 2001. Dal 2002 è ricercatore nel settore scientifico-disciplinare INF/01 Informatica presso l'Università di Chieti-Pescara.

Si occupa principalmente di semantica dei linguaggi di programmazione, in particolare di interpretazione astratta e modelli categoriali. Nell'ambito della teoria dell'interpretazione dell'astratta, dimostra che la completezza dei domini astratti rispetto ad operatori concreti additivi è una proprietà che si preserva rispetto alle operazioni di lub e glb del reticolo delle astrazioni. Dal punto di vista più pratico, si occupa del raffinamento dei framework goal-dependent già esistenti per l'analisi statica di programmi logici, con l'introduzione di operatori più precisi sia per l'unificazione forward che per quella backward. Sempre nell'ambito dell'analisi di programmi logici, studia anche in dettaglio l'interazione tra le informazioni di aliasing e di linearità, e determina per la prima volta gli operatori astratti ottimali per Sharing*Lin e per altri domini similari.

Sempre in quest'ambito si occupa di estendere le analisi statiche studiate sulla programmazione logica pura a linguaggi logici più complessi, basati su formule di Harrop ereditarie piuttosto che su clausole Horn. Il problema viene affrontato direttamente dal punto di vista proof-theoretic, astraendo le prove nel calcolo dei sequenti che corrispondo alle derivazioni SLD.

Si occupa nel contempo di estensioni della programmazione logica e dello sviluppo di un framework, basato sulle categorie indicizzate, che consente di estendere la programmazione logica pura con constraint, tipi di dati astratti, eccezioni, etc.. preservando la corrispondenza tra semantiche dichiarativa, operazionale e di punto fisso.

Pubblicazioni:

- AMATO G., COPPOLA M, GNESI S, SCOZZARI F, SEMINI L. (2006). Modeling Web Applications by the Multiple Levels of Integrity Policy. ELECTRONIC NOTES IN THEORETICAL COMPUTER SCIENCE. vol. 157, pp. 167-185 ISSN: 1571-0661. doi:10.1016/j.entcs.2005.12.053.
- AMATO G., SCOZZARI F. (2003). A general framework for variable aliasing: Towards optimal operators for sharing properties. LECTURE NOTES IN COMPUTER SCIENCE. vol. 2664, pp. 52-70 ISSN: 0302-9743.
- AMATO G. (2001). Correct Answers for First Order Logic. ELECTRONIC NOTES IN THEORETICAL COMPUTER SCIENCE. vol. 48, pp. 45-64 ISSN: 1571-0661. doi:10.1016/S1571-0661(04)00149-5.
- AMATO G., SCOZZARI F. (2005). On abstract unification for variable aliasing. Technical Report TR-05-08, Dipartimento di Informatica, Università di Pisa.
- AMATO G., SCOZZARI F. (2005). Optimality in goal-dependent Analysis of

Sharing. Technical Report TR-05-06, Dipartimento di Informatica, Università di Pisa.

2. **ZAFFANELLA Enea**

Curriculum:

Enea Zaffanella (Laurea in Scienze dell'Informazione, Università di Pisa, 1993; Dottorato di Ricerca in Informatica, Università di Leeds, 2002) dal 2006 è Professore Associato presso l'Università degli Studi di Parma, inquadrato nel SSD INF/01 Informatica; dal 2002 al 2006 ha ricoperto il ruolo di Ricercatore Universitario presso la stessa sede. Da oltre un decennio svolge attività di ricerca nel campo dei metodi formali per i linguaggi di programmazione, con particolare riferimento ai settori dell'analisi statica di programmi e dell'interpretazione astratta. È coautore di più di 30 pubblicazioni su riviste e atti di conferenze internazionali, oltre a varie comunicazioni a workshop internazionali e rapporti tecnici. Ha partecipato a diversi progetti di ricerca nazionali ed internazionali: ESPRIT Basic Research Action n. 6707 (`ParForce'); COFIN99 `Certificazione automatica di programmi mediante interpretazione astratta'; COFIN00 `Interpretazione astratta, sistemi di tipo e analisi Control-Flow'; COFIN01 `Ragionamento su aggregati e numeri a supporto della programmazione e relative verifiche'; COFIN02 `Verifica di sistemi reattivi basata su vincoli'; COFIN04 `Interpretazione astratta: progettazione ed applicazioni'; Azioni Integrate Italia-Spagna IT229 `Ambienti avanzati per lo sviluppo di programmi logici'. Collabora alla progettazione e sviluppo di software utilizzabile per scopi di analisi statica di programmi: China (http://www.cs.unipr.it/China/), un analizzatore statico altamente innovativo per programmi logici con vincoli; la Parma Polyhedra Library (PPL, http://www.cs.unipr.it/ppl/), una libreria C++ moderna, robusta ed efficiente per la manipolazione di vari domini numerici di interesse per l'analisi di programmi; PURRS (http://www.cs.unipr.it/purrs/), un progetto interdisciplinare per la soluzione o approssimazione automatica di relazioni di ricorrenza; CLAIR (http://www.cs.unipr.it/clair/), un prototipo di analizzatore per linguaggi imperativi.

Pubblicazioni:

- BAGNARA R, HILL P. M, ZAFFANELLA E. (2005). Not Necessarily Closed Convex Polyhedra and the Double Description Method. FORMAL ASPECTS OF COMPUTING. vol. 17(2), pp. 222-257 ISSN: 0934-5043. doi:10.1007/s00165-005-0061-1.
- BAGNARA R, HILL P. M, RICCI E, ZAFFANELLA E. (2005). Precise Widening Operators for Convex Polyhedra. SCIENCE OF COMPUTER PROGRAMMING. vol. 58(1-2), pp. 28-56 ISSN: 0167-6423. doi:10.1016/j.scico.2005.02.003.
- BAGNARA R, HILL P. M, ZAFFANELLA E. (2006). Widening Operators for Powerset Domains. INTERNATIONAL JOURNAL ON SOFTWARE TOOLS FOR TECHNOLOGY TRANSFER. vol. 8 (no. 4/5), pp. 449-466 ISSN: 1433-2779. doi:10.1007/s10009-005-0215-8.
- BAGNARA R, HILL P. M, MAZZI E, ZAFFANELLA E. (2005). Widening Operators for Weakly-Relational Numeric Abstractions. 12th International Symposium on Static Analysis. September 7-9, 2005. (vol. 3672, pp. 3-18). ISBN/ISSN: 978-3-540-28584-9. doi:10.1007/11547662_3 Lecture Notes in Computer Science. BERLIN: Springer-Verlag (GERMANY).
- BAGNARA R, HILL P. M, ZAFFANELLA E. (2007). An Improved Tight Closure Algorithm for Integer Octagonal Constraints. Quaderno 467. Dipartimento di Matematica, Università di Parma.
- BAGNARA R, DOBSON K, HILL P. M, MUNDELL M, ZAFFANELLA E. (2007). Grids: A Domain for Analyzing the Distribution of Numerical Values. Logic-based Program Synthesis and Transformation, 16th International Symposium. July 12-14, 2006. (vol. 4407, pp. 219-235). ISBN/ISSN: 978-3-540-71409-5. doi:10.1007/978-3-540-71410-1_16 Lecture Notes in Computer Science. BERLIN: Springer-Verlag (GERMANY).

- BAGNARA R, RODRIGUEZ-CARBONELL E, ZAFFANELLA E. (2005). Generation of Basic Semi-algebraic Invariants Using Convex Polyhedra. 12th International Symposium on Static Analysis (London, UK). September 7-9, 2005. (vol. 3672, pp. 19-34). ISBN/ISSN: 978-3-540-28584-9. doi:10.1007/11547662_4 Lecture Notes in Computer Science. BERLIN: Springer-Verlag (GERMANY).
- BAGNARA R, HILL P. M, ZAFFANELLA E. (2007). Applications of Polyhedral Computations to the Analysis and Verification of Hardware and Software Systems. Quaderno 458. Dipartimento di Matematica, Università di Parma.
- BAGNARA R, HILL P. M, ZAFFANELLA E. (2006). The Parma Polyhedra Library: Toward a Complete Set of Numerical Abstractions for the Analysis and Verification of Hardware and Software Systems. Quaderno 457. Dipartimento di Matematica, Università di Parma.
- BAGNARA R, ZAFFANELLA E., HILL P. M. (2005). Enhanced Sharing Analysis Techniques: A Comprehensive Evaluation. THEORY AND PRACTICE OF LOGIC PROGRAMMING. vol. 5 (1 & 2), pp. 1-43 ISSN: 1471-0684. doi:10.1017/S1471068404001978.
- BAGNARA R, GORI R, HILL P. M, ZAFFANELLA E. (2004). Finite-Tree Analysis for Constraint Logic-Based Languages. INFORMATION AND COMPUTATION. vol. 193, pp. 84-116 ISSN: 0890-5401.
- HILL P. M, ZAFFANELLA E., BAGNARA R. (2004). A Correct, Precise and Efficient Integration of Set-Sharing, Freeness and Linearity for the Analysis of Finite and Rational Tree Languages. THEORY AND PRACTICE OF LOGIC PROGRAMMING. vol. 4, pp. 289-323 ISSN: 1471-0684.

Testo inglese

1. **AMATO Gianluca**

Curriculum:

Dr. Gianluca Amato graduates in Computer Science at the University of Pisa, in 1996. From the same university he get the PhD in computer science, in 2001. From 2002 it is an assistant professor at the University of Chieti-Pescara.

He is mainly interested in semantics of programming languages, in particular in abstract interpretation and categorical models. In the theory of abstract interpretation, he proves that meet and join operators in the lattice of abstrat domains preserve completeness w.r.t. additive concrete operators. From a practical point of view, he works on the refinement of existing frameworks for goal-dependent analysis of logic programs. He introduces new operators for forward and backward unification which are more precise then standard ones. Moreover, he also studies the interaction between aliasing and linearity properties. He gives for the first time optimal abstract operators for unification and matching in the domain Sharing*Lin and other similar domains.

In the field of abstract interpretation, he also extends static analysis used for pure logic programs to more general logic programs, which are based on Hereditary Harrop Formulas instead of Horn clauses. The approach is completely proof-theoretical, since abstractions are built directly on the proofs in the sequent calculi.

In the same time, he is working to develop a framework, based on indexed categories, which is able to cope with several extensions of pure logic programs (constraints, exceptions, abstract data types,...) while retaining the correspondence between declarative, operational and fixpoint semantics.

Publications:

- AMATO G., COPPOLA M, GNESI S, SCOZZARI F, SEMINI L. (2006). Modeling Web Applications by the Multiple Levels of Integrity Policy. ELECTRONIC NOTES IN THEORETICAL COMPUTER SCIENCE. vol. 157, pp. 167-185 ISSN: 1571-0661. doi:10.1016/j.entcs.2005.12.053.
- AMATO G., SCOZZARI F. (2003). A general framework for variable aliasing: Towards optimal operators for sharing properties. LECTURE NOTES IN COMPUTER SCIENCE. vol. 2664, pp. 52-70 ISSN: 0302-9743.
- AMATO G. (2001). Correct Answers for First Order Logic. ELECTRONIC NOTES IN THEORETICAL COMPUTER SCIENCE. vol. 48, pp. 45-64 ISSN: 1571-0661. doi:10.1016/S1571-0661(04)00149-5.
- AMATO G., SCOZZARI F. (2005). On abstract unification for variable aliasing. Technical Report TR-05-08, Dipartimento di Informatica, Università di Pisa.
- AMATO G., SCOZZARI F. (2005). Optimality in goal-dependent Analysis of Sharing. Technical Report TR-05-06, Dipartimento di Informatica, Università di Pisa.

2. **ZAFFANELLA Enea**

Curriculum:

Enea Zaffanella (Master Degree in Computer Science, University of Pisa, 1993; PhD in Computer Science, University of Leeds, 2002) since 2006 is an Associate Professor in Computer Science at the University of Parma; from 2002 to 2006 he has been an Assistant Professor in the same university. Since more than a decade he is doing research in the field of formal methods for programming languages, in particular for program analysis and abstract interpretation. He has coauthored more than 30 publications on international journals and proceedings of international conferences, as well as many communications to international workshops and technical reports. He took part in several national and international research projects: ESPRIT Basic Research Action n. 6707 (`ParForce'); COFIN99 `Automatic Program Certification by Abstract Interpretation'; COFIN00 `Abstract Interpretation, type systems and control-flow analysis'; COFIN01 `Aggregate- and Number-Reasoning for Computing: from Decision Algorithms to Constraint Programming with Multisets, Sets, and Maps'; COFIN02 `Constraint-Based Verification of Reactive Systems'; COFIN04 `Abstract Interpretation: Design and Applications'; Integrated Action Italy-Spain IT229 `Advanced Development Environments for Logic Programs'. He collaborates to the design and development of software for the static analysis of programs: China (http://www.cs.unipr.it/China/), an innovative static analyser for constraint logic languages; the Parma Polyhedra Library (PPL, http://www.cs.unipr.it/ppl/), a modern, robust and efficient C++ library for the representation and manipulation of several numeric abstractions that are of interest in program analysis; PURRS (http://www.cs.unipr.it/purrs/), an interdisciplinary project for the automatic solution or approximation of recurrence relations; CLAIR (http://www.cs.unipr.it/clair/), a prototype analyzer for imperative languages.

Publications:

- BAGNARA R, HILL P. M, ZAFFANELLA E. (2005). Not Necessarily Closed Convex Polyhedra and the Double Description Method. FORMAL ASPECTS OF COMPUTING. vol. 17(2), pp. 222-257 ISSN: 0934-5043. doi:10.1007/s00165-005-0061-1.
- BAGNARA R, HILL P. M, RICCI E, ZAFFANELLA E. (2005). Precise Widening Operators for Convex Polyhedra. SCIENCE OF COMPUTER PROGRAMMING. vol. 58(1-2), pp. 28-56 ISSN: 0167-6423. doi:10.1016/j.scico.2005.02.003.
- BAGNARA R, HILL P. M, ZAFFANELLA E. (2006). Widening Operators for Powerset Domains. INTERNATIONAL JOURNAL ON SOFTWARE TOOLS FOR TECHNOLOGY TRANSFER. vol. 8 (no. 4/5), pp. 449-466 ISSN: 1433-2779. doi:10.1007/s10009-005-0215-8.
- BAGNARA R, HILL P. M, MAZZI E, ZAFFANELLA E. (2005). Widening

Operators for Weakly-Relational Numeric Abstractions. 12th International Symposium on Static Analysis. September 7-9, 2005. (vol. 3672, pp. 3-18). ISBN/ISSN: 978-3-540-28584-9. doi:10.1007/11547662_3 Lecture Notes in Computer Science. BERLIN: Springer-Verlag (GERMANY).

- BAGNARA R, HILL P. M, ZAFFANELLA E. (2007). An Improved Tight Closure Algorithm for Integer Octagonal Constraints. Quaderno 467. Dipartimento di Matematica, Università di Parma.
- BAGNARA R, DOBSON K, HILL P. M, MUNDELL M, ZAFFANELLA E. (2007). Grids: A Domain for Analyzing the Distribution of Numerical Values. Logic-based Program Synthesis and Transformation, 16th International Symposium. July 12-14, 2006. (vol. 4407, pp. 219-235). ISBN/ISSN: 978-3-540-71409-5. doi:10.1007/978-3-540-71410-1_16 Lecture Notes in Computer Science. BERLIN: Springer-Verlag (GERMANY).
- BAGNARA R, RODRIGUEZ-CARBONELL E, ZAFFANELLA E. (2005). Generation of Basic Semi-algebraic Invariants Using Convex Polyhedra. 12th International Symposium on Static Analysis (London, UK). September 7-9, 2005. (vol. 3672, pp. 19-34). ISBN/ISSN: 978-3-540-28584-9. doi:10.1007/11547662_4 Lecture Notes in Computer Science. BERLIN: Springer-Verlag (GERMANY).
- BAGNARA R, HILL P. M, ZAFFANELLA E. (2007). Applications of Polyhedral Computations to the Analysis and Verification of Hardware and Software Systems. Quaderno 458. Dipartimento di Matematica, Università di Parma.
- BAGNARA R, HILL P. M, ZAFFANELLA E. (2006). The Parma Polyhedra Library: Toward a Complete Set of Numerical Abstractions for the Analysis and Verification of Hardware and Software Systems. Quaderno 457. Dipartimento di Matematica, Università di Parma.
- BAGNARA R, ZAFFANELLA E., HILL P. M. (2005). Enhanced Sharing Analysis Techniques: A Comprehensive Evaluation. THEORY AND PRACTICE OF LOGIC PROGRAMMING. vol. 5 (1 & 2), pp. 1-43 ISSN: 1471-0684. doi:10.1017/S1471068404001978.
- BAGNARA R, GORI R, HILL P. M, ZAFFANELLA E. (2004). Finite-Tree Analysis for Constraint Logic-Based Languages. INFORMATION AND COMPUTATION. vol. 193, pp. 84-116 ISSN: 0890-5401.
- HILL P. M, ZAFFANELLA E., BAGNARA R. (2004). A Correct, Precise and Efficient Integration of Set-Sharing, Freeness and Linearity for the Analysis of Finite and Rational Tree Languages. THEORY AND PRACTICE OF LOGIC PROGRAMMING. vol. 4, pp. 289-323 ISSN: 1471-0684.

.