

CURRICULUM VITAE ET STUDIORUM

Roberto Bagnara

Indice

1. Scheda anagrafica	3
1.1. Recapiti	3
1.2. Attuale posizione professionale	3
1.3. Titoli di studio	3
1.4. Lingue straniere scritte e parlate	3
2. Curriculum vitae et studiorum	3
3. Descrizione dell'attività didattica	5
3.1. Titolarità di insegnamenti ufficiali	5
3.2. Affidamento di insegnamenti ufficiali	6
3.3. Altri corsi	6
3.4. Collaborazioni didattiche	6
3.5. Attività di relatore e correlatore per tesi di laurea	7
3.6. Dottorati di ricerca	8
3.7. Altre attività didattiche	8
4. Descrizione dell'attività di ricerca	8
4.1. Interessi di ricerca	8
4.2. Descrizione dei lavori pubblicati	9
4.3. Software recente	20
4.3.1. Parma Polyhedra Library	20
4.3.2. ECLAIR	20
4.3.3. PURRS	22
4.3.4. CHINA	22
4.3.5. OCRA	22
4.3.6. CLAIR	22
4.4. Periodi di ricerca congiunta all'estero	23
4.5. Seminari	23
4.6. Comitati di programma	25
4.7. Organizzazione di scuole, conferenze, workshop e seminari	26
4.8. Attività editoriali	26
4.9. Attività di referee per riviste internazionali	26
4.10. Attività di referee per conferenze internazionali	27
4.11. Partecipazione a progetti di ricerca	29
4.11.1. Progetti Terminati o in Corso	29

4.11.2. Progetti Presentati	30
4.12. Partecipazione a scuole	31
4.13. Partecipazione a conferenze e workshop nazionali	31
4.14. Partecipazione a conferenze e workshop internazionali	31
5. Elenco delle pubblicazioni	33
5.1. Riviste internazionali	34
5.2. Atti di conferenze internazionali	35
5.3. Atti di workshop internazionali	37
5.4. Curatele	39
5.5. Tesi di dottorato	39
5.6. Rapporti tecnici	39
5.7. Lavori sottomessi a riviste internazionali	42
5.8. Altri scritti	42

1. Scheda anagrafica

Cognome e nome: BAGNARA Roberto
Data di nascita: 7 dicembre 1963
Luogo di nascita: Faenza (RA)
Residenza: Via di Belvedere 1 — 57028 Suvereto (LI)
Professione: docente universitario
Posizione militare: in congedo illimitato per fine ferma
Stato civile: coniugato

1.1. Recapiti

Ufficio: Dipartimento di Matematica
Università di Parma
Viale G. P. Usberti 53/A
43100 Parma
Tel: 0521/906917
Fax: 0521/906950

Abitazione: Via di Belvedere 1
57028 Suvereto (LI)
Tel: 0565/828310

Mobile: 339/8593517
Email: bagnara@cs.unipr.it

1.2. Attuale posizione professionale

Professore associato, inquadrato nel settore scientifico disciplinare INF/01 (Informatica), presso il Dipartimento di Matematica dell'Università di Parma.

1.3. Titoli di studio

- Diploma di Perito Industriale per le Telecomunicazioni, Istituto Tecnico Industriale di Cesena (FO), anno scolastico 1981–82.
- Laurea con lode in “Scienze dell’Informazione”, Università degli Studi di Pisa, 17 luglio 1992.
- Dottorato di ricerca in “Informatica” (VIII ciclo), Università degli Studi di Pisa, 8 settembre 1997.

1.4. Lingue straniere scritte e parlate

Francese (avanzato), inglese (avanzato), spagnolo (intermedio).

2. Curriculum vitae et studiorum

1982 Consegue il diploma di Perito Industriale per le Telecomunicazioni con la votazione di 52/60.

1983–84 Assolve agli obblighi di leva prestando servizio militare dal 18 maggio 1983 al 7 maggio 1984 presso il II Battaglione Granatieri “Cengio”, Roma.

1984–87 Prende parte ad un concorso per un posto di Assistente Tecnico presso l’Università di Bologna, vincendolo. Assume servizio il 9 maggio 1984 presso il Dipartimento di Fisica di detta università. Lavora nel gruppo di Fisica Biomedica, divenendo responsabile dello studio e sviluppo dei sistemi di acquisizione dati in tempo reale per gli esperimenti effettuati dal suddetto gruppo. Si occupa inoltre di analisi dei segnali e delle immagini partecipando, in qualità di esperto software, a ricerche riguardanti:

- acquisizione dati ed analisi in tempo reale di segnali cardiaci (in collaborazione con l’ospedale S. Orsola di Bologna);
- acquisizione dati ed analisi in tempo reale di segnali sismici (in collaborazione con il Dipartimento di Geofisica dell’Università di Bologna);
- riconoscimento automatizzato (mediante analisi delle immagini) di cellule cancerose;
- classificazione automatizzata (mediante analisi delle immagini) di particelle, per lo studio della deposizione negli aerosol.

Come parte integrante delle sue mansioni sviluppa un interprete ed un compilatore FORTH usati (sia in ambiente uniprocessor che in ambiente multi-processor con memoria condivisa) per l’acquisizione dati e l’analisi in tempo reale. Partecipa alla costruzione di una rete per la trasmissione di dati tra le apparecchiature di laboratorio e il centro di calcolo del Dipartimento di Fisica. Nell’ambito di questo lavoro sviluppa un programma Kermit per macchine basate sulla famiglia di processori MC680x0. Tale programma è stato usato su processori *embedded* nei sistemi di acquisizione dati, in sistemi CP/M68K, OS9, VERSADOS e altri [81]. L’implementazione, chiamata “Kermit68K”, è stata distribuita dal “Columbia University Center for Computing Activities”. Collabora all’esperimento “DELPHI” del CERN (il Laboratorio Europeo per la Fisica delle Particelle), prendendo parte alle attività del gruppo “DAS” (Data Acquisition Systems) e occupandosi dello sviluppo di software per la diagnostica automatica dei malfunzionamenti hardware dei sistemi di acquisizione dati. Rassegna le proprie dimissioni volontarie il 31 dicembre 1987.

1988 Viene scelto dal CERN come *Technical Student Fellow*. L’1 gennaio 1988 inizia il lavoro al CERN (Data Handling Division, Online Computing Group, Readout Architecture Section), prendendo parte allo sviluppo di sistemi *real-time*, in particolar modo per ciò che concerne il software di base per microprocessori (sistemi di *Event Handling* e di *Asynchronous Service Trap* [17, 82]) e il software di supporto per microprocessori (cross-assemblatori, cross-compiler e pre-processor). Lavora inoltre, insieme a Tim Berners-Lee (il padre del World-Wide Web) ed altri, allo sviluppo di sistemi RPC (*Remote Procedure Call*) [18, 83] per l’implementazione di sistemi distribuiti. Conclude il suo periodo al CERN il 30 settembre 1988. Continua però il rapporto di collaborazione come consulente esterno ed è di nuovo al CERN dall’1 luglio al 31 agosto 1989.

- 1988–1992** Frequenta il corso di laurea in Scienze dell’Informazione presso l’Università degli Studi di Pisa, seguendo un piano di studi (indirizzo generale) orientato all’approfondimento delle basi logico-matematiche dell’informatica e dei linguaggi di programmazione.
- 1992** Il 17 luglio 1992 consegue il diploma di Laurea in Scienze dell’Informazione con la votazione di 110/110 e lode, discutendo la dissertazione dal titolo “Interpretazione Astratta di Linguaggi Logici con Vincoli su Domini Finiti” [84]. Relatore: Prof. Giorgio Levi, controrelatore: Prof. Ugo Montanari.
- 1992** Partecipa come collaboratore a contratto ad attività di ricerca coordinate dal Prof. Giorgio Levi presso il Dipartimento di Informatica dell’Università degli Studi di Pisa nell’ambito di progetti ESPRIT.
- 1992-1997** Alla fine del 1992 risulta vincitore di due posti dell’ottavo ciclo del Dottorato di Ricerca, rispettivamente in Informatica presso il Dipartimento di Informatica di Pisa e in Matematica Computazionale e Informatica Matematica presso il Dipartimento di Matematica dell’Università di Padova. Esercita opzione per il Dottorato di Ricerca in Informatica. Discute la tesi [53] dinanzi alla commissione nazionale l’8 settembre 1997 con esito positivo.
- 1997** Dall’1 gennaio al 31 ottobre svolge l’attività di *research fellow* presso la *School of Computer Studies* della *University of Leeds* (Inghilterra).
- 1997-2001** Dall’1 novembre 1997 al 15 dicembre 2001 è ricercatore presso il Dipartimento di Matematica dell’Università degli Studi di Parma. In questo periodo inizia a stabilire i contatti che porteranno alla creazione di un piccolo gruppo di ricerca sul tema dell’analisi dei programmi.
- 2001-presente** Dal 16 dicembre 2001 è professore associato presso il Dipartimento di Matematica dell’Università degli Studi di Parma. Contribuisce, tra l’altro, all’istituzione ed organizzazione del corso di laurea in “Informatica” (nuovo ordinamento).

3. Descrizione dell’attività didattica

3.1. Titolarità di insegnamenti ufficiali

Tutte le attività didattiche qui elencate si riferiscono a corsi di laurea dell’Università degli Studi di Parma.

1. *Programmazione I* (1 modulo), corsi di laurea in “Matematica”, “Matematica e Informatica” e “Matematica per la tecnologia e la finanza”, a.a. 2001-2002.
2. *Fondamenti dell’Informatica*, corso di laurea in “Informatica”, aa.aa. 2002-2003, 2003-2004, 2005-2006, 2006-2007 e 2007-2008.
3. *Fondamenti dell’Informatica*, corsi di laurea in “Matematica” e “Matematica e Informatica”, aa.aa. 2002-2003, 2003-2004, 2004-2005, 2006-2007 e 2007-2008.

3.2. Affidamento di insegnamenti ufficiali

Tutte le attività didattiche qui elencate si riferiscono a corsi di laurea dell'Università degli Studi di Parma.

1. *Informatica* (80 ore), corso di laurea in “Biotecnologie” (v.o.), aa.aa. 1999-2000 e 2000-2001.
2. *Laboratorio di informatica* (1 modulo semestrale), corso di laurea in “Matematica” (v.o.), a.a. 2000-2001.
3. *Programmazione (metodi avanzati)* (2 moduli semestrali), corso di laurea in “Matematica” (v.o.), aa.aa. 2000-2001 e 2002-2003.
4. *Programmazione 3* corsi di laurea in “Matematica” e “Matematica e Informatica”, a.a. 2002-2003.
5. *Scrittura matematica e informatica*, corsi di laurea in “Matematica” e “Matematica e Informatica”, a.a. 2002-2003.
6. *Scrittura matematica e informatica*, corso di laurea in “Informatica”, aa.aa. 2003-2004, 2004-2005 e 2005-2006.
7. *Linguaggi di programmazione* corso di laurea in “Informatica”, aa.aa. 2006-2007 e 2007-2008.
8. *Analisi e verifica del software*, corso di laurea in “Informatica”, aa.aa. 2003-2004, 2004-2005, 2005-2006, 2006-2007 e 2007-2008.

3.3. Altri corsi

1. *Computer Aided Verification of Complex Systems: An Introduction*, dottorato in “Matematica, Statistica, Scienze Computazionali e Informatica” e dottorato in “Informatica”, Università degli Studi di Milano, aa.aa. 2006-2007 e 2007-2008.
2. *Algorithmique*, Licence de mathématiques, Université de la Réunion, a.a. 2006-2007.
3. *Laboratorio di informatica*, Corso di Laurea di “Scienze Ambientali”, curriculum in “Ecologia e Gestione della Fascia Costiera”, Università di Siena, sede di Follonica (GR), a.a. 2007-2008.

3.4. Collaborazioni didattiche

1. Gruppo di lezioni (8 ore) su “La compilazione di Prolog e la Warren Abstract Machine”, nell’ambito del corso di *Linguaggi Speciali di Programmazione* del corso di laurea in “Scienze dell’Informazione” dell’Università di Pisa, aa.aa. 1990-1991, 1991-1992, 1992-1993 e 1993-1994; docente: Prof. Giorgio Levi.
2. Esercitazioni per il corso di “Linguaggi Formali e Compilatori” del corso di laurea in “Scienze dell’Informazione” dell’Università di Pisa, a.a. 1995-1996; docente: Prof. Pierpaolo Degano.

3. Lezioni ed esercitazioni teoriche e pratiche per il corso di *Teoria ed applicazioni delle macchine calcolatrici* (TAMC) del corso di laurea in “Scienze Ambientali” dell’Università di Parma, aa.aa. 1997-1998, 1998-1999, 1999-2000 e 2000-2001; docente: Prof. Gianfranco Rossi.
4. Esercitazioni teoriche e pratiche per il corso di *Fondamenti dell’informatica* del corso di laurea in “Matematica” dell’Università di Parma, aa.aa. 1998-1999 e 1999-2000; docente: Prof. Grazia Lotti.
5. Lezioni e supporto ai progetti per il corso di *Metodologie di Programmazione*, del corso di laurea in “Informatica” dell’Università di Parma, aa.aa. 2003-2004 e 2004-2005; docente: Dott. Enea Zaffanella.

3.5. Attività di relatore e correlatore per tesi di laurea

1. MOTTA, M. Un’implementazione efficiente ed innovativa del dominio *Pos* per l’analisi statica di programmi logici. Tesi di laurea in “Scienze dell’Informazione”, Università degli Studi di Pisa. Relatore: G. Levi; correlatore: R. Bagnara; a.a. 1994-1995.
2. SCUDELLARI, M. C. Analisi bottom-up di programmi logici con vincoli basata su trasformazioni *Magic-Set*. Tesi di laurea in “Scienze dell’Informazione”, Università degli Studi di Pisa. Relatore: G. Levi; correlatore: R. Bagnara; a.a. 1994-1995.
3. CASTELLACCI, B. Implementazione di un’analisi di tipi per programmi logici basata su grammatiche regolari. Tesi di laurea in “Scienze dell’Informazione”, Università degli Studi di Pisa. Relatore: G. Levi; correlatore R. Bagnara; a.a. 1995-1996.
4. STAZZONE, A. Annotazione e trasformazione di programmi logici con vincoli. Tesi di laurea in “Matematica”, Università degli Studi di Parma. Relatore: R. Bagnara; correlatore: E. Zaffanella; a.a. 1999-2000.
5. RICCI, E. Rappresentazione e manipolazione di poliedri convessi per l’analisi e la verifica di programmi. Tesi di laurea in “Matematica”, Università degli Studi di Parma. Relatore: R. Bagnara; correlatori: C. Medori ed E. Zaffanella; a.a. 2001-2002.
6. ZOLO, T. Risoluzione automatica di relazioni di ricorrenza. Tesi di laurea in “Matematica”, Università degli Studi di Parma. Relatore: R. Bagnara; correlatori: A. Zaccagnini ed E. Zaffanella; a.a. 2001-2002.
7. PESCECETTI, A. L’algoritmo di Zeilberger per la risoluzione automatica di ricorrenze. Tesi di laurea in “Matematica”, Università degli Studi di Parma. Relatore: R. Bagnara; correlatore: A. Zaccagnini; a.a. 2004-2005.
8. CIMINO, A. Un’implementazione incrementale e su aritmetica esatta del simpleso primale. Tesi di laurea in “Informatica”, Università degli Studi di Parma. Relatore: E. Zaffanella; correlatore: R. Bagnara; a.a. 2004-2005.

9. VINCENZI, A. Interpretazione astratta di operatori per la manipolazione di bit in linguaggi imperativi. Tesi di laurea in “Informatica”, Università degli Studi di Parma. Relatore: R. Bagnara; correlatore: E. Zaffanella; a.a. 2005-2006.
10. QUARTIERI, B. Implementazione efficiente di domini astratti numerici debolmente relazionali. Tesi di laurea in “Matematica”, Università degli Studi di Parma. Relatore: R. Bagnara; correlatore: E. Zaffanella; a.a. 2005-2006.
11. FRANCHI, E. A contribution to the issue of string cleanness: a design of an automatic program transformation. Tesi di laurea in “Matematica e Informatica”, Università degli Studi di Parma. Relatore: R. Bagnara; correlatore: E. Zaffanella; a.a. 2006-2007.
12. SOFFIA, S. Definizione ed implementazione di una analisi di points-to per linguaggi C-like. Tesi di laurea in “Informatica”, Università degli Studi di Parma. Relatore: R. Bagnara; correlatore: E. Zaffanella; a.a. 2007-2008.
13. BOSSI, F. CORAL: a modern C++-library for the manipulation of Boolean functions. Tesi di laurea in “Informatica”, Università degli Studi di Parma. Relatore: R. Bagnara. a.a. 2007-2008.
14. BOLZONI, P. Automatic checking of coding rules. Tesi di laurea in “Informatica”, Università degli Studi di Parma. Relatore: R. Bagnara; correlatore: E. Zaffanella; a.a. 2008-2009.

3.6. Dottorati di ricerca

1. Membro del collegio dei docenti del dottorato in “Matematica, Statistica, Scienze Computazionali e Informatica”, XVI, XVII, XVIII, XIX, XX, XXI, XXII, XXIII e XXIV ciclo, Dipartimento di Matematica dell’Università degli Studi di Milano.

3.7. Altre attività didattiche

1. Lezioni seminariali per il “Corso di perfezionamento per la formazione degli insegnanti di matematica” organizzato dalla Facoltà di Scienze MM. FF. NN. dell’Università degli Studi di Parma, aa.aa. 1997-1998 e 1998-1999.
2. Corso di alfabetizzazione informatica per gli insegnanti delle scuole materne ed elementari di Suvereto (LI), 1999.

4. Descrizione dell’attività di ricerca

4.1. Interessi di ricerca

- Semantica dei linguaggi di programmazione
- Tecniche formali per l’analisi e la verifica dei programmi

- Interpretazione astratta
- Tecniche avanzate di compilazione;
- Programmazione logica e con vincoli
- Computer algebra

4.2. Descrizione dei lavori pubblicati

- Nella tesi di Laurea [84] (i cui risultati sono stati in seguito estesi e pubblicati in [19, 37, 38]) è stata studiata l'applicabilità di tecniche per la soluzione approssimata di vincoli, note nel campo dell'Intelligenza Artificiale, all'analisi a tempo di compilazione dei programmi logici con vincoli (CLP). L'idea trae origine dalla necessità di studiare il comportamento di un programma logico con vincoli numerici mediante la sua valutazione su un dominio approssimato di vincoli. Le tecniche di inferenza su grafo e di propagazione di vincoli su reti di vincoli trovano quindi una naturale applicazione nel campo dell'analisi statica di programmi mediante interpretazione astratta.
- In [19, 37, 38] sono state studiate le applicazioni di queste tecniche all'analisi statica di linguaggi logici con vincoli su domini finiti e sul dominio dei numeri reali, con particolare riferimento alla determinazione di vincoli impliciti e ridondanti. Questo tipo di analisi ha molteplici applicazioni nell'ottimizzazione del procedimento di compilazione di detti linguaggi.
- In [2, 39] sono state introdotte alcune costruzioni su domini di vincoli. Tali costruzioni consentono, in modo automatico, di "migliorare" un dominio dato incorporando informazioni di tipo implicativo e disgiuntivo. Diversi domini interessanti per l'analisi di programmi logici con vincoli possono essere costruiti e giustificati in modo molto semplice facendo ricorso a dette costruzioni. È stato poi mostrato che tali costruzioni danno luogo ad un'algebra di domini di vincoli con proprietà molto interessanti. Questa metodologia per la definizione di nuovi domini di vincoli è utilizzata nella *Parma Polyhedra Library* [30, 64, 89] ed è stata estesa con operatori di widening generici in [32, 70].
- Per la parte sperimentale del lavoro di ricerca, hanno giocato un ruolo importante il progetto e la realizzazione dell'analizzatore CHINA [53, 88]. CHINA è un analizzatore data-flow/control-flow per programmi logici con vincoli su domini simbolici (alberi finiti o alberi razionali) e numerici. Il prototipo corrente (circa 40.000 righe di codice C++) ha funzionalità molto estese rispetto agli analizzatori attualmente disponibili. In particolare incorpora combinazioni di domini piuttosto sofisticate (vedi sopra), ottimizzazioni della procedura di calcolo dei punti fissi, informazioni su pattern di chiamata e di successo per mezzo di trasformazione del programma eccetera.
- In [20, 40] viene affrontato il problema della determinazione *incrementale* delle variabili *ground* in analisi basate su *alberi binari di decisione* (BDD). Tale problema è di estrema rilevanza per molte applicazioni pratiche. Infatti, l'analisi di *groundness* è di cruciale importanza per l'ottimizzazione

di programmi logici e CLP. Inoltre l'incrementalità è necessaria per poter disporre efficientemente di informazioni sulla groundness delle variabili *durante* l'analisi (e non solo al termine dell'analisi stessa). Tale necessità emerge nell'analisi di linguaggi che impiegano meccanismi di *delay* (che sono solitamente basati su condizioni di groundness). Un esempio è $\text{CLP}(\mathcal{R})$, dove i vincoli non lineari sono sospesi fino al momento in cui diventano lineari. La disponibilità delle informazioni di groundness durante l'analisi permette anche di impiegare, nell'analisi di *aliasing*, domini computazionalmente meno complessi di *Sharing*, a parità di precisione (anzi con l'incremento di precisione dovuto al passaggio da *Def* a *Pos*) e con significativi vantaggi sull'efficienza complessiva dell'analisi. In [20, 40] vengono esplorate alcune possibili soluzioni, a partire da quelle più semplici, fino ad arrivare ad una rappresentazione *ibrida* che distingue le variabili *ground* dalle *dipendenze ground*. Il risultato è sorprendente: la rappresentazione ibrida è sensibilmente più efficiente delle rappresentazioni usuali "pure" (ovvero basate solo su BDD). E questo a prescindere dal fatto che si sia interessati o meno alle informazioni di groundness nel corso dell'analisi. Ad essere migliorata è l'efficienza dell'analisi di groundness in sé stessa, sia in termini di velocità di esecuzione che (ed ancor più) in termini di memoria utilizzata. Il fatto che la rappresentazione ibrida consenta l'accesso in tempo costante (e molto efficiente) alle variabili *ground* dà poi luogo ad ovvi ed ulteriori vantaggi.

In [23, 44] quest'idea viene spinta oltre, separando le informazioni sull'equivalenza di coppie di variabili. Due variabili sono equivalenti se sono entrambe *ground* o entrambe non *ground*. La nuova rappresentazione ibrida che deriva da questa ulteriore fattorizzazione, grazie ad uno sforzo implementativo molto accurato ed attento, è nettamente più efficiente di ogni altro approccio. L'analisi di programmi sul dominio *Pos* è stata resa più veloce, su programmi di media grandezza, di un ordine di grandezza e più. Non solo: programmi di grandi dimensioni che prima non erano assolutamente trattabili con *Pos* ora sono analizzabili in tempi del tutto ragionevoli.

- In [41] viene presentata una caratterizzazione semantica adatta a catturare non solo i linguaggi CLP "ideali", ma anche i linguaggi realmente implementati e, in particolare quei sistemi che, per ragioni di efficienza, impiegano un risolutore di vincoli *incompleto* ($\text{CLP}(\mathcal{R})$ e $\text{clp}(\text{fd})$, ad esempio). La semantica presentata in [41] è una versione, riveduta e corretta in alcuni punti cruciali, di quella introdotta da Jaffar e Maher. In [41] vengono messe in luce l'importanza e la ragionevolezza di un'ipotesi di incrementalità del risolutore di vincoli (il risultato del risolutore non dipende dal fatto che un certo numero di vincoli gli sia presentato in blocco o uno alla volta). Sotto questa condizione la semantica definita in [41], con caratterizzazioni sia top-down che bottom-up, ha le seguenti caratteristiche: è *AND-composizionale* (i vincoli di risposta a goal qualsiasi possono essere ricostruiti a partire dai vincoli di risposta a goal atomici *più generali*); non perde la distinzione tra vincoli attivi (inviati al risolutore di vincoli) e passivi (ovvero sospesi). Viene inoltre mostrato che l'ipotesi di incrementalità è così ragionevole che, senza di essa, è impos-

sibile definire una *semantica basata su atomi* che sia, al tempo stesso, AND-composizionale e indipendente dalla regola di calcolo.

- L'obiettivo dell'*analisi di sharing* per i programmi logici è quello di determinare, per ogni punto del programma, quali *coppie* di variabili logiche possono essere legate a termini che condividono una terza variabile logica. Il dominio normalmente utilizzato per detta analisi (*Sharing*, Jacobs e Langen, 1989) caratterizza invece gli *insiemi* di variabili che possono condividere una variabile. In [3, 21] ci si chiede, per la prima volta, se questo dominio (oramai standard) non sia forse inutilmente complesso per l'analisi di sharing. La risposta è positiva: definendo una relazione di equivalenza su *Sharing* siamo in grado di ottenere un dominio più semplice, riducendo la complessità (nel caso peggiore) della procedura di unificazione da esponenziale a polinomiale. Si dimostra inoltre che il nuovo dominio (in seguito denominato *PSD*, per *Pair-Sharing Dependencies*) non può essere ulteriormente astratto senza incorrere in perdite di precisione dell'analisi corrispondente.
- Naturalmente, è importante che gli analizzatori statici siano basati su risultati teorici dimostrati in modo convincente. In programmazione logica, il dominio *Sharing*, come si è detto, è una scelta pressoché standard per l'analisi di *sharing* e per ulteriori studi teorici. A dispetto di ciò si è riscontrato che non esistevano, prima della pubblicazione dei lavori [4, 22, 45], dimostrazioni soddisfacenti delle proprietà chiave, commutatività ed idempotenza, che rendono *Sharing* ben definito. Inoltre, le prove o congetture pubblicate circa la correttezza di *Sharing* presumono che il linguaggio analizzato implementi l'unificazione con l'*occur-check*, il che è falso nella quasi totalità dei casi. I lavori [4, 22, 45] chiudono queste falle, rimaste aperte per quasi un decennio. In particolare, viene data una generalizzazione della funzione di astrazione di *Sharing* che può essere applicata ad ogni linguaggio, con o senza l'*occur-check*. Oltre a questo, vengono dimostrati i risultati di correttezza, idempotenza e commutatività per l'operazione di unificazione astratta.
- Avendo constatato che né il dominio *Sharing* di Jacobs e Langen, né la sua astrazione *PSD* descritta in [3, 21, 43] consentono di analizzare programmi di grandi dimensioni, in [24, 47, 56] è stato studiato il dominio proposto da C. Fecht, formato dalla combinazione del dominio *Pos* con un'astrazione molto debole di *Sharing*. Sebbene questa combinazione rappresenti un buon compromesso tra precisione ed efficienza, si sono riscontrate perdite di precisione significative nell'analisi di parecchi programmi "reali". Questa perdita di precisione concerne la groundness, il pair-sharing, la linearità, ma non la freeness: prendendo spunto da questa osservazione empirica, si è dimostrato formalmente che una famiglia piuttosto ampia di astrazioni di *Sharing* non comporta perdite di precisione sulla freeness. In [24, 47, 56] si definisce un nuovo dominio per l'analisi di sharing che supporta l'implementazione di svariate tecniche di widening. In particolare, con questo dominio risulta semplice utilizzare l'idea di Fecht per realizzare un widening vero e proprio. Sono stati considerati anche widening più precisi, ma l'estensiva sperimentazione che è stata condotta suggerisce come sia difficile migliorare il primo widening proposto, a patto che il dominio

Pos sia incluso nel dominio complessivo. Quando questo non sia il caso, widening più precisi basati su *clique* nei grafi di pair-sharing consentono di ottenere una migliore precisione a costi assolutamente accettabili.

- La *complementazione*, e cioè l'operazione inversa del prodotto ridotto, rappresenta una tecnica relativamente nuova per la determinazione di decomposizioni minimali dei domini astratti. Filé e Ranzato hanno introdotto un metodo particolarmente semplice per il calcolo del complemento. Tra le applicazioni del loro metodo, essi hanno considerato la rimozione per complementazione del dominio di pair-sharing (*PS*) dal dominio di set-sharing di Jacobs e Langen's (*Sharing*). Dal momento che il risultato di questa operazione è lo stesso *Sharing*, Filé e Ranzato ne hanno concluso che il dominio *PS* è "troppo astratto" per lo scopo che si erano prefissi.

In [5, 25, 57] si mostra come la radice di queste difficoltà non stia in *PS*, ma in *Sharing* e, più precisamente, nell'informazione ridondante che *Sharing* contiene rispetto alle dipendenze ground e al pair-sharing. Infatti, le difficoltà svaniscono se la nostra versione non ridondante di *Sharing*, *PSD* [3, 5, 21, 43], viene utilizzata in luogo di *Sharing*. Per stabilire questo risultato su *PSD*, abbiamo definito uno schema generale per sottodomini di *Sharing* che include *Def* (un dominio per l'analisi di groundness) e *PSD* come casi particolari. Questo getta nuova luce sulla struttura di *Sharing* e svela una connessione naturale, sebbene inattesa, tra *Def* e *PSD*.

In [5, 25, 57], inoltre, viene data sostanza all'affermazione del fatto che la complementazione, da sola, non è sufficiente per ottenere decomposizioni veramente minimali. La soluzione giusta a questo problema consiste prima nella rimozione delle ridondanze attraverso il calcolo del quoziente del dominio rispetto al comportamento osservabile, e poi nella decomposizione per complementazione.

- In [27, 42, 58] è presentata la costruzione razionale di un dominio generico con informazione strutturale per l'analisi di programmi CLP. Il dominio proposto, $\text{Pattern}(\mathcal{D})$, ha come parametro un qualsiasi dominio astratto, \mathcal{D} , che soddisfi alcuni requisiti assolutamente ragionevoli. Il dominio discende dal dominio parametrico per l'analisi di programmi logici $\text{Pat}(\mathfrak{R})$ introdotto da A. Cortesi e colleghi. D'altra parte, la formalizzazione del nostro dominio astratto per l'analisi di programmi CLP è indipendente dalla tecnica di implementazione: $\text{Pat}(\mathfrak{R})$ (opportunamente esteso per trattare correttamente i linguaggi CLP che omettono l'occurs-check nella procedura di unificazione) è una delle possibili implementazioni. Ragionando ad un più alto livello di astrazione siamo in grado di riferirci a nozioni classiche della teoria dell'unificazione e di consentire all'implementatore una notevole libertà d'azione. Infatti, come dimostriamo in [27, 42, 58], un analizzatore che incorpori informazione strutturale basandosi sul nostro approccio può essere estremamente competitivo, non solo dal punto di vista della precisione, ma anche dal punto di vista dell'efficienza.
- Riguardo la precisione delle combinazioni di domini che includono il dominio *Sharing* di Jacobs e Langen, esiste un piccolo nucleo di tecniche, quali la combinazione standard con informazioni di linearità e di *freeness*, solitamente denotata con *SFL*, che sono oramai ampiamente accettate ed

utilizzate. Esistono diverse altre proposte per il raffinamento di *Sharing*. Queste proposte, che sono circolate nella comunità scientifica più o meno inosservate per anni, sono accomunate dal fatto che nessuna di esse sembra essere stata convalidata sperimentalmente dai rispettivi autori. Per questa ragione, come parte del nostro sforzo di spingere la precisione dell'analisi oltre i limiti attuali, in [8, 26, 46] ci si è posti il problema di verificare se e quanto dette proposte possano contribuire ad un effettivo miglioramento della precisione. In particolare, in [8, 26, 46] sono state discusse e/o valutate sperimentalmente le seguenti possibilità:

1. l'effetto della propagazione su *SFL* delle variabili *ground* identificate da *Pos*;
 2. l'incorporazione di informazione strutturale esplicita nel dominio di analisi;
 3. tecniche più sofisticate per integrare *SFL* e *Pos*;
 4. la questione del riordinamento dei *binding* nel calcolo dell'operatore *mgv* astratto;
 5. una proposta originale concernente l'aggiunta di un dominio che mantenga informazioni sull'insieme di variabili che sono *ground* o *free*;
 6. una tecnica raffinata per utilizzare le informazioni di linearità;
 7. la possibilità di migliorare la precisione dell'analisi mantenendo informazioni sulla *compoundness* delle variabili (in questo contesto, una variabile è detta *compound* se è certamente legata ad un termine che non è una variabile);
 8. il recupero di "informazione nascosta" nella combinazione di *Sharing* con l'usuale dominio per la *freeness*.
- Le attrattive dei linguaggi logici basati sulla teoria degli alberi razionali (come Prolog II ed i suoi successori, SICStus Prolog e Oz) sono:
 1. l'espressività con cui consentono la codifica di grammatiche ed altri oggetti autoreferenziali;
 2. l'efficienza della procedure di unificazione, che non richiede l'*occurs-check* senza per questo rinunciare alla correttezza.

Sfortunatamente, l'uso degli alberi razionali infiniti è causa di problemi seri. Ad esempio, molti predicati *built-in* e di libreria sono indefiniti per tali alberi e la loro applicazione deve quindi essere soggetta a controlli che, se svolti a tempo di esecuzione, comportano costi computazionali significativi. Inoltre, diverse tecniche di analisi e manipolazione dei programmi sono corrette limitatamente a quelle parti di programma o a quelle esecuzioni che, dimostrabilmente, sono interessate solo da alberi finiti. È perciò importante identificare almeno in parte, automaticamente e staticamente, quelle variabili (le *variabili finite*) che, nei punti del programma di interesse, saranno sempre legate a termini finiti.

In [28, 59] viene proposta una nuova analisi data-flow, formalizzata in termini di interpretazione astratta, che cattura questa informazione. Il

dominio è parametrico: una semplice componente per registrare l'insieme delle variabili finite viene accoppiato ad un dominio generico (il parametro della costruzione) che fornisce informazioni di *sharing*. Quest'ultimo dominio è specificato in modo astratto così da garantire la correttezza del dominio composito e, al tempo stesso, la generalità dell'approccio.

In [29, 60] viene introdotto un dominio di funzioni Booleane che esprime, in modo piuttosto preciso, come la finitezza di una variabile influenzi la finitezza di altre variabili. La verifica sperimentale che abbiamo condotto mostra come la combinazione dei domini descritti in [28, 59] e in [29, 60] fornisca un metodo valido (ovvero applicabile a programmi reali) per ottenere precise informazioni sulla finitezza dei termini coinvolti nelle computazioni di programmi logici con vincoli.

I lavori [7, 71] combinano ed estendono le idee di [28, 29, 59, 60].

- Il dominio dei poliedri convessi riveste un ruolo fondamentale in molte applicazioni per l'analisi statica e la verifica (semi-) automatica di sistemi hardware e software. In [30, 64] vengono evidenziate alcune problematiche relative alla rappresentazione e manipolazione dei poliedri convessi *non necessariamente chiusi*, per i quali le librerie software esistenti offrono un supporto inadeguato. Viene affrontato e risolto il problema della rappresentazione ad alto livello di tali poliedri per mezzo del metodo della *Doppia Descrizione* (introdotto da Motzkin e colleghi), rendendola indipendente dalla particolare tecnica di implementazione utilizzata. Inoltre, per l'implementazione basata su ϵ -rappresentazioni introdotta da Halbwachs e colleghi, viene mostrato come l'algoritmo standard di minimizzazione della rappresentazione fornisca risultati insoddisfacenti, aprendo la strada ad inefficienze evitabili e possibili errori di programmazione. Per risolvere il problema evidenziato, viene definito un concetto più forte di minimizzazione, specificando e dimostrando corretto l'algoritmo corrispondente.

In [48, 49] viene proposta una generalizzazione dell'implementazione dei poliedri NNC basata sulla ϵ -rappresentazione, consentendo di apprezzarne meglio le potenzialità ed i limiti. In particolare, si mostra l'esistenza di una rappresentazione alternativa, che risulta avere caratteristiche computazionali duali rispetto a quella originale. I risultati presentati in [30, 48, 49], completi delle dimostrazioni formali ed esposti in un contesto omogeneo, sono pubblicati in [11].

- Gli operatori di *widening*, la cui formalizzazione si deve a P. e R. Cousot, consentono di forzare e/o accelerare la convergenza di un calcolo di punto fisso nel caso in cui siano impiegati domini astratti infiniti o computazionalmente troppo costosi. Nel caso del dominio dei poliedri, l'unico operatore di widening formalizzato ed utilizzato nella pratica è quello proposto da Cousot ed Halbwachs nel 1978, che giustamente prende il nome di widening *standard*. Tale operatore, però, risulta essere troppo impreciso per gli scopi di alcune applicazioni dell'analisi statica. Sebbene alcuni operatori più precisi siano stati proposti in letteratura, questi non sono veri e propri widening, in quanto non garantiscono la convergenza della computazione in un tempo finito. Per ovviare a questo problema, in

[31, 67] si definisce uno schema generale per la definizione di nuovi widening sul dominio dei poliedri. Usando tale schema, una o più euristiche di approssimazione possono essere sistematicamente trasformate in operatori di widening più precisi del widening standard. Lo schema è stato istanziato considerando sia operatori già proposti in letteratura, sia operatori inediti: il widening ottenuto, all'atto pratico, è stato verificato essere più preciso del widening standard per una percentuale significativa dei *benchmark* considerati. La versione estesa del lavoro presentato in [31, 67] è stata recentemente accettata per la pubblicazione su rivista [10].

- Uno dei successi finora ottenuti con la *Parma Polyhedra Library* è costituito dalla suo utilizzo in *cTI*, il primo sistema per l'inferenza di terminazione universale *left* per programmi logici. L'inferenza di terminazione generalizza sia l'analisi che il controllo di terminazione. Tradizionalmente, un analizzatore di terminazione tenta di dimostrare che una data classe di interrogazioni di un programma logico termina. Questa classe deve essere fornita all'analizzatore, generalmente per mezzo di annotazioni che l'utente deve inserire. Naturalmente, l'analisi deve essere rieseguita ogni volta che la classe di interrogazioni viene modificata. L'inferenza di terminazione, al contrario, non richiede né le annotazioni dell'utente, né il ricalcolo. Infatti, con questo approccio, la classe di interrogazioni che l'analisi riuscirebbe a dimostrare essere terminanti viene inferita in un colpo solo. In [9] viene descritta l'architettura della nuova versione di *cTI* e vengono riportati i risultati di un'estensiva valutazione sperimentale del sistema che raccoglie molti esempi classici della letteratura sulla terminazione in programmazione logica e diversi programmi Prolog di dimensione e complessità assolutamente rispettabili. Grazie alla *Parma Polyhedra Library* la nuova versione di *cTI* è fino a due ordini di grandezza più efficiente della precedente implementazione.
- La progettazione ed implementazione di domini astratti espressivi ed efficienti per l'analisi data-flow e la verifica dei sistemi informatici sono compiti ardui. Per questo motivo, continua ad esservi un grande interesse nei confronti di quelle metodologie che consentono di derivare un dominio astratto complesso applicando costruzioni sistematiche a domini astratti più semplici già esistenti. Di importanza chiave sono quelle tecniche che consentono di derivare, in modo automatico o quasi, degli operatori semantici corretti (anche se potenzialmente non ottimali) per i nuovi domini astratti. In questo contesto, la derivazione automatica degli operatori di widening merita una attenzione particolare, in quanto detti operatori devono anche garantire la convergenza dell'analisi, oltre alla sua correttezza. In [32] vengono proposte due tecniche alternative per specificare operatori di widening generici per un dominio astratto ottenuto per mezzo della costruzione *finite powerset* (la quale consente di rappresentare tutte le disgiunzioni finite di elementi del dominio astratto di partenza). Entrambi gli operatori sono ottenuti a partire da un qualunque operatore di widening definito sul dominio sottostante e sono parametrici rispetto alla specifica di alcune operazioni supplementari. Al fine di meglio esemplificare l'uso delle tecniche proposte, ne viene illustrata la loro istanziazione sul dominio *finite powerset* dei poliedri convessi, per il quale nessun operatore di widening (non banale) è stato proposto precedentemente. La

versione estesa del lavoro, pubblicata in [12, 70], oltre alla dimostrazione dei risultati formali enunciati in [32], contiene anche la specifica di una terza tecnica per la derivazione automatica di un operatore di widening.

- L'obiettivo dell'analisi di complessità dei programmi è la determinazione di approssimazioni inferiori e superiori di misure di complessità di algoritmi, processi e strutture dati. Quando queste analisi sono parzialmente o completamente automatizzate, esse possono essere di supporto al programmatore nella comprensione dei programmi, guidare l'applicazione di trasformazioni per l'ottimizzazione di programmi e condurre alla scoperta di problemi di efficienza dovuti ad errori di programmazione molto difficilmente individuabili in altro modo. Le relazioni di ricorrenza giocano un ruolo molto importante nell'analisi di complessità dei programmi, dal momento che le misure di complessità si possono spesso e convenientemente esprimere per mezzo di sistemi di relazioni di ricorrenza. Il progetto PURRS (Parma University's Recurrence Relation Solver) ha per obiettivo la creazione di una libreria software in grado di risolvere od approssimare le relazioni di ricorrenza, con particolare riferimento all'analisi di complessità dei programmi. Il lavoro di ricerca finora svolto ha portato alla definizione di alcuni algoritmi che, in modo completamente automatico e sostanzialmente efficiente, consentono la soluzione o l'approssimazione (sia dall'alto che dal basso) di un'ampia classe di relazioni di ricorrenza. In particolare, sono state considerate le seguenti classi di relazioni di ricorrenza:

- ricorrenze lineari di ordine finito a coefficienti costanti [68];
- ricorrenze lineari di ordine 1 a coefficienti variabili;
- una sottoclasse delle ricorrenze lineari di ordine infinito;
- forme speciali di ricorrenze non lineari;
- le cosiddette *equazioni funzionali* di rango 1, che scaturiscono dall'analisi di complessità degli algoritmi *divide-et-impera*.

Per fare un esempio, consideriamo la ricorrenza lineare di ordine k data da

$$x(n) = a_1x(n-1) + \dots + a_kx(n-k) + p(n). \quad (1)$$

Al momento attuale, PURRS è in grado di risolvere *esattamente* relazioni di ricorrenza lineari a coefficienti costanti di ordine fino a 4 (oltre a quelle di ordine più grande per cui sia possibile determinare le radici del polinomio caratteristico $\lambda^k - (a_1\lambda^{k-1} + \dots + a_k)$), se la parte non omogenea p è una combinazione lineare di prodotti di polinomi ed esponenziali. In realtà, è già possibile determinare la soluzione di alcune speciali ricorrenze del tipo (1) anche quando l'ordine è 1 e p è una funzione razionale (ad esempio, PURRS è in grado di trovare la formula chiusa per la somma parziale n -esima della serie di Mengoli), ed anche risolvere una classe di relazioni di ricorrenza lineari di ordine 1 a coefficienti variabili, oppure dimostrare automaticamente che non esiste una formula chiusa per la soluzione.

Inoltre, PURRS riesce a determinare l'ordine di grandezza corretto della soluzione di una classe di ricorrenze generalizzate del tipo

$$x(n) = \alpha x(n/\beta) + g(n), \quad (2)$$

dove $\alpha > 0$, $\beta > 1$ è intero, e g è una combinazione lineare di prodotti di polinomi ed esponenziali e di prodotti di polinomi e funzioni logaritmiche. (Qui $x(n/\beta)$ deve essere inteso come $x(\lfloor n/\beta \rfloor)$). Queste ultime ricorrenze scaturiscono naturalmente nell'analisi di complessità di algoritmi *divide et impera*. Si noti che per le ricorrenze del tipo (2) può accadere un fenomeno che non capita alle soluzioni delle ricorrenze del tipo (1): per esempio, se $\alpha = \beta = 2$, $g(n) = 0$ ed $x(1) = 1$, allora

$$\liminf_{n \rightarrow \infty} \frac{x(n)}{2^n} = \frac{1}{2}, \quad \limsup_{n \rightarrow \infty} \frac{x(n)}{2^n} = 1,$$

dato che $x(2^m + a) = 2^m$, se $0 \leq a < 2^m$. In altre parole, il rapporto fra la soluzione di questa ricorrenza generalizzata ed il suo termine dominante non ha limite, ma oscilla fra due quantità positive e finite. Per questo tipo di ricorrenze ed una vasta classe di funzioni g sono state ottenute approssimazioni della soluzione dell'ordine di grandezza corretto, cioè due funzioni "semplici" $x_1(n)$ ed $x_2(n)$ tali che $x_1(n) \leq x(n) \leq x_2(n)$ per ogni $n \geq 1$, e che

$$L_1 = \liminf_{n \rightarrow \infty} \frac{x(n)}{x_1(n)} > 0, \quad L_2 = \limsup_{n \rightarrow \infty} \frac{x(n)}{x_2(n)} < +\infty.$$

Quando $g(n) = n^k \alpha^n$, il sistema PURRS determina le funzioni x_1 ed x_2 in modo che il rapporto L_2/L_1 sia il più piccolo possibile, ferma restando la condizione $x_1(n) \leq x(n) \leq x_2(n)$ per ogni $n \geq 1$. Per fare questo si sono dovuti estendere in varie direzioni i risultati già noti, che riguardavano quasi esclusivamente la determinazione di maggiorazioni per la soluzione di ricorrenze di tipo (7) della forma $x(n) = O(x_2(n))$, senza preoccuparsi né del valore della costante implicita nella notazione di Landau, né di dare le corrispondenti minorazioni.

Uno dei problemi che sono sorti durante lo sviluppo di PURRS è il seguente: dal momento che le soluzioni in forma chiusa di ricorrenze anche modeste possono essere molto complesse, come possiamo acquistare confidenza nel nostro risolutore? Come possiamo validare o forse confutare i risultati che questo fornisce? Inoltre, in quei casi in cui le soluzioni esatte sono così complesse da essere inutilizzabili, come possiamo cedere precisione in cambio di semplicità approssimandole dall'alto e dal basso? E infine, riguardo al problema di gestire *insiemi* di soluzioni di relazioni di ricorrenza: come possiamo confinare tali insiemi per mezzo di limitazioni inferiori e superiori? In [69] vengono date alcune soluzioni a questi problemi, facendo attenzione, ove possibile, ad utilizzare solo aritmetica intera e/o condizioni che possono essere controllate molto efficientemente e in modo completamente automatico. La valutazione sperimentale di queste idee sta dando risultati molto promettenti, con *speedup* di un ordine di grandezza e più rispetto ai metodi tradizionali. Grazie a questo lavoro, PURRS è in grado, in un gran numero di casi, di certificare rapidamente la correttezza della soluzione determinata, verificando che soddisfa le condizioni iniziali e la relazione di ricorrenza.

- Nell'ambito dei domini astratti numerici, la continua ricerca del compromesso ideale tra il costo computazionale e la precisione intrinseca delle

approssimazioni ha portato, negli anni recenti, all'identificazione di un'ampia classe di domini "debolmente relazionali" (*weakly-relational*), nei quali cioè le relazioni tra le variabili dei programmi sono catturate con precisione limitata rispetto a quanto è possibile fare con il dominio dei poliedri convessi. Ne sono esempio i domini astratti delle differenze vincolate, degli ottagoni, degli ottaedri e dei vincoli templatici. Per la maggior parte, tali domini sono definiti in maniera sintattica, per cui elementi diversi possono di fatto rappresentare lo stesso oggetto concreto, portando a potenziali ridondanze, interfacce poco intuitive e, in taluni casi, problemi di convergenza dell'analisi. In [33, 73], estensioni di una proposta inizialmente presentata in [72], si mostra come, mediante la specifica di opportune procedure di minimizzazione, sia possibile definire le versioni semantiche di tali domini astratti, risolvendo tutti i problemi di cui sopra.

- In [34, 74] viene presentata un'analisi statica per la generazione di *disuguaglianze polinomiali* invarianti mediante interpretazione astratta. La tecnica si basa sull'approssimazione di congiunzioni di disuguaglianze polinomiali (di grado limitato) per mezzo di poliedri convessi nei quali l'introduzione di dimensioni ausiliarie, corrispondenti ai monomi di grado superiore ad uno, consente una caratterizzazione parziale delle relazioni non lineari. L'approccio proposto migliora la letteratura esistente, per lo più limitata all'inferenza di *equazioni polinomiali* invarianti, pur evitando la complessità proibitiva dei metodi (teoricamente completi, ma praticamente improponibili) basati sull'eliminazione esatta dei quantificatori. Lo studio sperimentale condotto ha evidenziato come sia possibile produrre invarianti non banali in un tempo ragionevole, consentendo in taluni casi la verifica di proprietà che vanno oltre la potenza espressiva delle classiche invarianti lineari.
- In [35, 63] è stato studiato in dettaglio il dominio astratto delle *griglie*, in grado di rappresentare insiemi di punti ed iperpiani linearmente disposti su di uno spazio vettoriale n -dimensionale. Per tale dominio, utile per l'analisi statica dei pattern di distribuzione dei valori delle variabili del programma, è stato presentato un insieme completo di operatori che include tutto quanto è necessario per definire la semantica astratta e, in particolare, gli operatori di widening che ne garantiscono la calcolabilità in un numero finito di passi. A tal scopo si sfrutta un concetto di rappresentazione duale simile a quello sviluppato per i poliedri convessi, che oltre a consentire il riutilizzo di tecniche standard dell'algebra lineare, rende anche possibile un'implementazione concisa ed efficiente.
- I *vincoli ottagonali interi* (in inglese *Unit Two Variables Per Inequality* o *UTVPI integer constraints*) costituiscono un'interessante classe di vincoli per la rappresentazione e la risoluzione di problemi nel campo della programmazione con vincoli e dell'analisi e verifica formale di sistemi software e hardware. Essi, infatti, uniscono algoritmi di complessità polinomiale ad un potere espressivo apprezzabile. I principali algoritmi richiesti per la manipolazione di tali vincoli sono il test di soddisfacibilità e il calcolo della chiusura inferenziale di un insieme di vincoli. Quest'ultima è detta chiusura *stretta* (*tight*) per marcare la differenza con l'algoritmo di chiusura (incompleto) che non sfrutta l'integralità delle variabili. In [36, 77] vie-

ne presentato e pienamente giustificato un algoritmo $O(n^3)$ per il calcolo della chiusura stratta di un insieme di vincoli ottagonali interi.

- I poliedri convessi costituiscono la base per varie astrazioni usate in analisi statica e per la verifica automatica di sistemi complessi e talora critici. Per tali applicazioni l'identificazione di un adeguato compromesso tra precisione e complessità dell'analisi è un problema particolarmente acuto, per il quale la disponibilità di un ampio spettro di soluzioni alternative è irrinunciabile. In [14, 76]: viene presentata una rassegna degli impieghi del calcolo poliedrale in queste applicazioni; vengono illustrate le varie classi di poliedri che possono essere adottate, nonché le principali operazioni su poliedri richieste da analizzatori e verificatori automatici; e vengono, infine, considerate combinazioni di poliedri con altre astrazioni numeriche volte a migliorare la precisione dell'analisi. In ogni caso vengono evidenziati quegli aspetti dove l'ulteriore indagine teorica potrebbe portare contributi importanti.
- Dai suoi inizi come progetto per studenti nel 2001 —inizialmente solo per il trattamento, come il nome suggerisce, dei poliedri convessi— la *Parma Polyhedra Library* è stata continuamente migliorata ed estesa unendo una scrupolosa ricerca sui fondamenti teorici delle astrazioni numeriche (anche non convesse) ad una totale aderenza alle migliori pratiche dello sviluppo software. Sebbene la libreria non sia ancora pienamente matura e funzionalmente completa, la Parma Polyhedra Library già offre una combinazione di funzionalità, affidabilità, semplicità d'uso ed efficienza che non ha uguali in librerie simili e che siano liberamente disponibili. In [13, 75] vengono presentate le principali caratteristiche della versione 0.9 della libreria, concentrandosi su quelle che la distinguono da librerie analoghe e su quelle che sono importanti per le applicazioni nel campo dell'analisi e della verifica di sistemi hardware e software.
- Il progetto e l'implementazione di analizzatori statici precisi per linguaggi come C, C++, Java e Python sono problemi fortemente impegnativi. In [50, 78] si considera un “nucleo” di linguaggio imperativo che ha molte delle caratteristiche che si trovano in linguaggi di largo uso: funzioni possibilmente ricorsive, errori ed eccezioni sollevate dal supporto a tempo di esecuzione e/o definite dall'utente, un modello realistico dei dati e della memoria. Per questo linguaggio viene fornita una semantica concreta (che caratterizza sia le computazioni finite che quelle infinite) e una semantica astratta (davvero) generica che si dimostra essere corretta rispetto a quella concreta. La semantica astratta è generica in quanto progettata per essere completamente parametrica rispetto ai domini di analisi: in particolare, a differenza di altri schemi pubblicati, supporta domini *relazionali* (domini astratti che catturano relazioni non banali tra oggetti dato). In [50, 78] si mostra inoltre come la metodologia proposta può essere estesa per trattare linguaggi più complessi che includono puntatori, oggetti dato composti e meccanismi non strutturati di controllo del flusso. Tale metodologia è basata sull'interpretazione astratta di una semantica operazionale strutturata *big-step* (in particolare, sull'estensione G^∞ SOS che semplifica il trattamento delle computazioni infinite). Di particolare rilievo è la modularità dell'approccio, nel quale l'analizzatore statico viene

naturalmente partizionato in componenti con interfacce e responsabilità ben definite: questo aspetto semplifica grandemente la dimostrazione di correttezza e l'implementazione. Alcuni aspetti dell'implementazione di queste tecniche nel sistema ECLAIR sono trattati in [51]

4.3. Software recente

Gran parte del lavoro di ricerca teorica è stato integrato, verificato e, in casi particolari, ispirato dalla progettazione e successivo sviluppo di alcuni sistemi software che implementano (o utilizzano) tecniche di analisi statica dei programmi. Tutti i progetti software qui menzionati sono coordinati da Roberto Bagnara.

4.3.1. Parma Polyhedra Library

La *Parma Polyhedra Library* (PPL) è una libreria di astrazioni numeriche specialmente concepita per applicazioni nel campo dell'analisi e della verifica di sistemi hardware (digitali e/o analogici) e software. La PPL fornisce: poliedri convessi non necessariamente chiusi, griglie, un'ampia famiglia di sistemi di differenze vincolate e vincoli "ottagonali", la più estesa e potente famiglia di intervalli oggi implementata nel mondo del software libero (intervalli chiusi, possibilmente non chiusi, con limitazioni date da un qualsiasi tipo intero o floating point nativo o intero o razionale illimitato, possibilmente con restrizioni come *modulo interval* e *strided interval*), nonché prodotti e powerset finiti dei domini sopra citati. Altre caratteristiche peculiari della PPL sono la disponibilità di operatori di widening e di estrapolazione innovativi, l'elevata portabilità, la facilità d'uso, la totale dinamicità delle strutture dati utilizzate, la robustezza rispetto alle eccezioni, la disponibilità di interfacce C, Prolog, OCaml e Java, e una documentazione molto accurata [89]. La PPL è software libero distribuito nei termini della *GNU General Public License* (e dunque liberamente disponibile, in perpetuo, a chiunque voglia usarla, studiarla e modificarla) ed è disponibile, insieme alla relativa documentazione, all'URI <http://www.cs.unipr.it/pp1>. La PPL è utilizzata in numerosi progetti presso i più prestigiosi centri di ricerca nel campo dell'analisi e della verifica formale. In particolare, è utilizzata da GCC —la *GNU Compiler Collection*—, probabilmente la suite di compilatori più utilizzata al mondo.

4.3.2. ECLAIR

`eclair` è una nuova piattaforma professionale per la verifica della famiglia di linguaggi che deriva dal C. Al momento può analizzare vari dialetti del C, ma il lavoro per l'estensione a C++ è già iniziato. Le principali caratteristiche della versione corrente di `eclair` sono riassunte nei paragrafi seguenti.

Emulazione della *toolchain* e *build system* immutato Uno schema potente e completamente generico di elaborazione delle opzioni consente ad `eclair` di sostituirsi ad ogni componente della *toolchain*, come `gcc`, con le sue oltre 1000 opzioni, ed ogni altro compilatore, assembler o linker. Grazie a questo tipo di mimetismo, `eclair` può accedere a tutto il codice che compone un'applicazione o una libreria senza richiedere alcuna modifica del *build system*. In particolare,

se l'applicazione/libreria può essere costruita in parallelo (ad esempio, usando `make -j`), allora può essere analizzata con `eclair` in parallelo. Ecco come la versione 2.2.11 di `httpd` può essere analizzata, una volta che le verifiche da eseguire siano state specificate nel file `my_setup`:

```
$ tar jxf httpd-2.2.11.tar.bz2
$ cd httpd-2.2.11
$ ./configure
$ eclair_env -eval-file=my_setup -- make -j 6
```

Doppia interfaccia verso l'AST Il parser di `eclair` costruisce un albero di sintassi astratta (AST) che è disponibile alle altre componenti del sistema via due diverse interfacce (tra le quali è comunque garantita una completa interoperabilità): (1) come un termine Prolog che codifica direttamente tutti e soli gli aspetti sintattici del programma, e che incorpora come sottotermini delle “handle” che danno accesso a tutte le informazioni non sintattiche; (2) come insieme di classi C++ correlate: una rappresentazione più complessa che però dà accesso diretto a tutta l'informazione. La “vista Prolog” rende molto semplice la codifica di algoritmi basati sulla sintassi, come i controllori di regole di codifica, gli analizzatori ed i trasformatori di programmi.

Parsing veloce e preciso L'AST costruito dal parser codifica informazioni precise sul codice sorgente originale, così che è possibile recuperare, quasi per ogni token, informazioni complete circa la catena di inclusioni e, ortogonalmente, la catena di espansioni di macro che hanno portato quel token nel sorgente preprocessato. Ciò significa che possiamo costruire strumenti in grado di identificare il punto preciso del sorgente originale che è responsabile per un dato stato di cose. Il parser, che supporta i dialetti ISO C90 e C99 insieme a numerose estensioni GNU e Microsoft, è molto efficiente: oggi è in grado di elaborare più di 100,000 linee di codice al secondo, e c'è spazio per ulteriori miglioramenti.

Forma intermedia semplificata `eclair` include un trasformatore di programmi che, preservando la semantica, semplifica un programma in un sottoinsieme di C (o C++). La principale semplificazione effettuata consiste nella rimozione degli effetti collaterali dalle espressioni. La forma semplificata rende molto più semplice lo sviluppo di manipolatori di programmi basati sulla semantica.

Controllori di regole di codifica I controllori supportano i seguenti insiemi di regole: MISRA-C:2004, CERT C Secure Coding Standard, JSF C++, High-Integrity C++. La semplicità e l'eleganza con le quali i controllori possono essere definiti è disponibile agli utenti che desiderino svilupparne di propri.

Analisi precisa e scalabile Stiamo portando l'analizzatore semantico da un precedente prototipo di `eclair`. Questo include un'analisi dei puntatori *flow-*, *context-* e *field-sensitive* insieme ad analisi dei valori numerici basate sulle astrazioni fornite dalla *Parma Polyhedra Library*. Una caratteristica chiave di `eclair` sarà la riduzione dei falsi positivi grazie a sofisticate analisi semantiche e non attraverso l'introduzione di falsi negativi.

4.3.3. PURRS

PURRS (*Parma University's Recurrence Relation Solver*) è un prototipo di risolutore automatico di relazioni di ricorrenza, in grado di risolvere automaticamente equazioni alle differenze ed altre ricorrenze in senso generalizzato, dando la formula chiusa quando questo è possibile e buone stime per l'andamento asintotico della soluzione in caso contrario. Più precisamente, la soluzione della ricorrenza viene espressa in termini di funzioni note (polinomi, esponenziali, funzioni razionali, funzioni trigonometriche elementari, funzioni ipergeometriche, altre funzioni trascendenti) o eventualmente come somma parziale della serie che definisce qualcuna delle funzioni sopra elencate. La possibilità di ottenere delle vere e proprie limitazioni (superiori e inferiori) per la soluzione di relazioni di ricorrenza è il primo passo verso la definizione di un'analisi automatica di complessità che sia in grado di fornire garanzie effettive sull'utilizzo delle risorse. PURRS è software libero, distribuito nei termini della *GNU General Public License*. Maggiori informazioni sono disponibili all'URI <http://www.cs.unipr.it/purrs>. Allo stesso indirizzo è disponibile un prototipo interattivo che risolve esattamente, oppure approssima dall'alto e dal basso, le classi di ricorrenze più sopra descritte.

4.3.4. CHINA

CHINA (*Clp(H, N) Analyzer*) è un analizzatore statico parametrico per i linguaggi logici (con vincoli). L'analizzatore, che accetta programmi logici scritti in accordo allo standard ISO Prolog, comprende numerosi domini astratti che consentono di calcolare approssimazioni delle seguenti informazioni: groundness, sharing, freeness, linearità, compoundness, informazione strutturale, finiteness, tipi semplici, approssimazioni di variabili numeriche basate sul dominio degli intervalli, relazioni poliedrali tra le dimensioni dei termini. Ulteriori informazioni sono disponibili all'URI <http://www.cs.unipr.it/china>.

4.3.5. OCRA

OCRA (*Occur-Check Reduction Analyzer*) è un prototipo che utilizza le informazioni prodotte dall'analizzatore CHINA per la riduzione dell'occur-check nei linguaggi logici. Dato, ad esempio, un programma Prolog, produce un nuovo programma Prolog che può essere eseguito correttamente anche sui sistemi che omettono tale controllo.

4.3.6. CLAIR

CLAIR è un programma che è stato sviluppato per studiare in dettaglio alcune tematiche relative ai linguaggi di programmazione, vale a dire: analisi lessicale; analisi sintattica e generazione dell'albero di sintassi astratta (parsing); controllo statico della correttezza rispetto ai tipi; semantica operativa mediante sistema di transizioni; interpretazione (concreta). CLAIR supporta due linguaggi: un semplice linguaggio funzionale e un linguaggio imperativo che per certi versi ricorda Pascal. Entrambi i linguaggi usano la regola di scoping statico. L'approccio è basato sulla semantica operativa strutturata alla Plotkin, per la parte formale, e su Prolog, per la parte implementativa. Uno dei vantaggi di questo approccio combinato sta nella relativa facilità con la

quale si può estendere il sistema per supportare linguaggi con caratteristiche differenti. CLAIR è software libero, distribuito nei termini della *GNU General Public License* ed è disponibile, insieme alla relativa documentazione, all'URI <http://www.cs.unipr.it/clair>.

Il sistema e la sua documentazione sono stati usati nel corso di *Analisi e verifica del software*, corso di laurea in “Informatica” dell'Università di Parma, e nel corso *Programming Language Semantics (CS31)* della *School of Computing*, Università di Leeds (Regno Unito).

4.4. Periodi di ricerca congiunta all'estero

1. *Technical Student Fellow* presso il *CERN* (European Organization for Nuclear Research) di Ginevra (Svizzera), dall'1 gennaio al 30 settembre 1988.
2. *Visiting researcher* presso il *Department of Computer Science* della *Monash University* di Melbourne (Australia), dal 14 gennaio al 16 febbraio 1995.
3. *Research fellow* presso la *School of Computer Studies* della *University of Leeds* (Inghilterra), dall'1 gennaio al 31 ottobre 1997.
4. A partire dal 1998, ha visitato numerose volte (almeno una settimana ogni anno) la *School of Computing* della *University of Leeds*, dove collabora ad attività di ricerca con Patricia Hill.
5. Nell'ambito del programma di scambio docenti previsto dal protocollo culturale tra Italia e Spagna, ha effettuato una visita di studio, finanziata dal MURST, presso la *Facultad de Informática, Universidad Politécnica de Madrid*, collaborando con il gruppo del Prof. Manuel Hermenegildo, dal 27 gennaio al 2 febbraio 2001.
6. Nell'ambito del progetto “Ambienti Avanzati per lo Sviluppo di Programmi Logici” (Azioni Integrate Italia-Spagna 2001, codice IT229), ha effettuato visite di studio di una settimana l'una presso la *Facultad de Informática, Universidad Politécnica de Madrid*, nel novembre 2001, settembre 2002 e maggio 2003.
7. *Visiting scientist* presso il *Laboratoire d'Informatique* della *École Polytechnique* (LIX, Parigi) dal 15 giugno al 15 luglio 2008.
8. *Professeur invité* presso il *Département de Mathématiques et Informatique* della *Université de La Réunion*, S^t Denis (Oceano Indiano, Francia), per i mesi di maggio 2002, giugno 2005, maggio 2006, maggio 2007 e marzo 2010.

4.5. Seminari

1. “Introduzione alla Warren Abstract Machine”, Dipartimento di Informatica, Università di Pisa, marzo 1991.

2. “Data-Flow Analysis for Constraint Logic-Based Languages”, *School of Computer Studies, University of Leeds*, marzo 1997.
3. “Análisis eficiente de información estructural para lenguajes de programación lógica y de restricciones”, *Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid*, gennaio 2001.
4. “Widening Sharing”, *Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid*, gennaio 2001.
5. “The Parma Polyhedra Library”, *Institut de Recherche en Mathématiques et Informatique Appliquées, Université de La Réunion*, maggio 2002.
6. “Symbolic Computation Support for Complexity Analysis and the PURRS Project”, *Facultad de Informática, Universidad Politécnica de Madrid*, maggio 2003.
7. “Convex Polyhedra for the Analysis and Verification of Hardware and Software Systems: the ‘Parma Polyhedra Library’”, *Dipartimento di Matematica, Università di Parma*, dicembre 2003.
8. “Representation and Manipulation of Not Necessarily Closed Convex Polyhedra”, *Dipartimento di Matematica, Università di Parma*, febbraio 2004.
9. “Abstract Interpretation and the Parma Polyhedra Library: from Theory to Practice and Vice Versa”, *Dipartimento di Informatica, Sistemi e Produzione, Università degli Studi di Roma “Tor Vergata”*, novembre 2004.
10. “Análisi e verifica di sistemi software a mezzo interpretazione astratta: un’introduzione informale”, *Dipartimento di Ingegneria dell’Informazione, Università degli Studi di Siena*, gennaio 2006.
11. “Applications of Polyhedral Computations to the Analysis and Verification of Hardware and Software Systems”, *Polyhedral Computation Workshop (relazione invitata), Centre de recherches mathématiques, Université de Montréal, Montréal (Québec), Canada*, ottobre 2006.
12. “The Parma Polyhedra Library: A Library of Numerical Abstractions for Analysis and Verification”, *Department of Informatics and Mathematical Modelling, Technical University of Denmark (DTU), Lyngby, Danimarca*, giugno 2007.
13. “Numerical Abstract Domains”, *Department of Communication, Business and Information Technologies (CBIT), Roskilde University, Roskilde, Danimarca*, giugno 2007.
14. “On the Design of Generic Static Analyzers for Modern Imperative Languages”, *Copenhagen Programming Language Seminar (COPLAS), Department of Computer Science (DIKU), University of Copenhagen, Copenhagen, Danimarca*, giugno 2007.
15. “On the Design of Generic Static Analyzers for Modern Imperative Languages”, *Facultad de Informática, Universidad Politécnica de Madrid, Madrid, Spagna*, settembre 2007.

16. “Ranking Functions for Automatic Termination Analysis”, Dipartimento di Scienze dell’Informazione, Università di Bologna, dicembre 2007.
17. “On the Design of Generic Static Analyzers for Imperative Languages”, Dipartimento di Informatica, Università di Pisa, marzo 2008.
18. “Computer Aided Verification of Complex Systems: An Introduction”, Dipartimento di Matematica, Università di Milano, marzo 2008.
19. “Numerical Abstract Domains and the Parma Polyhedra Library”, Dipartimento di Informatica, Università di Verona, gennaio 2010.

4.6. Comitati di programma

1. Membro del comitato di programma della “2000 Joint Conference on Declarative Programming”, La Habana, Cuba, 4–7 dicembre, 2000.
2. Membro del comitato di programma del tredicesimo “Workshop on Logic Programming Environments”, Mumbai, India, 8 dicembre 2003.
3. Membro del *comité de lecture* della tredicesima edizione delle “Journées Francophones de Programmation en Logique et de programmation par Contraintes” (JFPLC 2004), Angers, Francia, 21–23 giugno 2004,
4. Membro del comitato di programma del quattordicesimo “International Symposium on Logic-based Program Synthesis and Transformation” (LOPSTR’05), Londra, Regno Unito, 7–9 settembre 2005.
5. Membro del comitato di programma della ventunesima “International Conference on Logic Programming” (ICLP’05), Barcellona, Spagna, 2–5 ottobre 2005.
6. Membro del comitato di programma del sedicesimo “International Symposium on Logic-based Program Synthesis and Transformation” (LOPSTR 2006), S. Servolo, Venezia, 12–14 luglio 2006.
7. Membro del comitato scientifico del *II Workshop su Open Source, Free Software e Open Format nei processi di ricerca archeologica*, Genova, 11 maggio 2007 (<http://workshop07.iossa.it/>).
8. Membro del comitato scientifico del *III Workshop su Open Source, Free Software e Open Format nei processi di ricerca archeologica*, Padova, 8-9 maggio 2008 (<http://www.archeologia.unipd.it/workshop08/>).
9. Membro del comitato di programma della *Technical Track on Software Verification* del ventitreesimo “Annual ACM Symposium on Applied Computing” (SAC 2008), Fortaleza, Ceará, Brasile, 16–20 marzo 2008.
10. Membro del comitato di programma della nona “International Conference on Verification, Model Checking, and Abstract Interpretation” (VMCAI 2008), San Francisco, USA, 7–9 gennaio 2008.
11. Membro del comitato di programma del quindicesimo “International Static Analysis Symposium” (SAS 2008), Valencia, Spagna, 16–18 luglio 2008.

12. Membro del comitato di programma del “Colloquium on Implementation of Constraint and Logic Programming Systems” (CICLOPS 2008), Udine, 12–13 dicembre 2008.
13. Membro del comitato di programma del diciannovesimo “International Symposium on Logic-based Program Synthesis and Transformation” (LOPSTR 2009), Coimbra, Portogallo, 9–11 settembre 2009.
14. Membro del comitato scientifico di *ArcheoFOSS 2010*, Foggia, Museo Civico, May 6-7, 2010 (<http://www.archeologiadigitale.it/archeofoss/2010.html>).

4.7. Organizzazione di scuole, conferenze, workshop e seminari

1. *Workshop Chair* per i “Joint International Symposia SAS’98 and PLILP-ALP’98”, Pisa, Italia, 14–18 settembre 1998.
2. Organizzatore, insieme a Patricia Hill della *University of Leeds* della “Second International Summer School on Computational Logic”, Maratea, Italia, 25–30 agosto 2002 (<http://www.cs.unipr.it/ISCL02>).
3. Organizzatore del ciclo di seminari su “Convex Polyhedra for the Analysis and Verification of Hardware and Software Systems”, Dipartimento di Matematica, Università di Parma, novembre 2003 — febbraio 2004 (http://www.cs.unipr.it/ppl/seminars_2003_2004).
4. Membro del comitato organizzatore del *CILC’04 – Convegno Italiano di Logica Computazionale*, diciannovesimo incontro annuale della “Associazione Italiana Gruppo Ricercatori e Utenti di Logic Programming” (GULP), Dipartimento di Matematica, Università di Parma, 16–17 giugno 2004 (<http://www.cs.unipr.it/CILC04>).
5. Membro del comitato scientifico e organizzatore (con Giancarlo Macchi) del *I Workshop su Open Source, Free Software e Open Format nei processi di ricerca archeologica*, Grosseto, 8 maggio 2006 (<http://www.archeogr.unisi.it/asiaa/open/>).
6. Membro del comitato scientifico e organizzatore (con Giancarlo Macchi) della *I-QMDAA Summer School on Quantitative Methods and Data Analysis in Archaeology*, Villa Lanzi, Campiglia Marittima, 10–17 settembre 2006 (<http://www.archeogr.unisi.it/qmdaa/>).

4.8. Attività editoriali

1. *Implementation Area Editor* e curatore della sezione *Net Talk* per l’*ALP newsletter*, il bollettino della *Association for Logic Programming*.

4.9. Attività di referee per riviste internazionali

1. *Journal of Logic Programming*;
2. *Journal of Functional and Logic Programming*;
3. *Information Processing Letters*;

4. Theory and Practice of Logic Programming;
5. Theoretical Computer Science.
6. Journal of Automated Reasoning.

4.10. Attività di referee per conferenze internazionali

1. GULP'93, "Ottavo Convegno sulla Programmazione Logica", Gizzeria Lido (CZ), giugno 1993.
2. AMAST'93, "Algebraic Methodology and Software Technology", Enschede, Olanda, giugno 1993.
3. WSA'93, "3rd International Workshop on Static Analysis", Padova, settembre 1993.
4. ILPS'93, "International Logic Programming Symposium", Vancouver, British Columbia, Canada, ottobre 1993.
5. AI*IA'93, "Terzo Congresso dell'Associazione Italiana per l'Intelligenza Artificiale", Torino, ottobre 1993.
6. SAC'94, "ACM Symposium on Applied Computing", Phoenix, Arizona, USA, marzo 1994.
7. ICLP'94, "International Conference on Logic Programming", S. Margherita Ligure, giugno 1994.
8. PLILP'94, "Programming Languages Implementation and Logic Programming", Madrid, Spagna, settembre 1994.
9. ALP'94, "Third International Conference on Algebraic and Logic Programming", Madrid, Spagna, settembre 1994.
10. GULP-PRODE'94, "Joint Conference on Declarative Programming", Peñíscola, Spagna, settembre 1994.
11. SAS'94, "International Static Analysis Symposium", Namur, Belgio, settembre 1994.
12. PLILP'95, "International Symposium on Programming Languages, Implementations, Logics and Programs", Utrecht, Paesi Bassi, settembre 1995.
13. ILPS'95, "International Logic Programming Symposium", Portland, Oregon, USA, dicembre 1995.
14. ESOP'96, "European Symposium on Programming", Linköping, Svezia, aprile 1996.
15. ECAI'96, "European Conference on Artificial Intelligence" Budapest, Ungheria, agosto 1996.
16. PLILP'96, "Eighth International Symposium on Programming Languages, Implementations, Logics, and Programs", Aachen, Germania, settembre 1996.

17. SAS'96, "Third International Static Analysis Symposium", Aachen, Germania, settembre 1996.
18. LOPSTR'97, "Seventh International Workshop on Logic Program Synthesis and Transformation", Leuven, Belgio, luglio 1997.
19. JICSLP'98, "Joint International Conference and Symposium on Logic Programming" Manchester, Regno Unito, giugno 1998.
20. SAS'98, "Fifth International Static Analysis Symposium", Pisa, Italia, settembre 1998.
21. PLILP/ALP'98, "Tenth International Symposium on Programming Languages, Implementations, Logics and Programs" and "Seventh International Conference on Algebraic and Logic Programming", Pisa, Italia, settembre 1998.
22. SAS'99, "International Static Analysis Symposium", Venezia, Italia, settembre 1999.
23. PPDP'99, "International Conference on Principles and Practice of Declarative Programming", Parigi, Francia, settembre/ottobre 1999.
24. SAS 2000, "Seventh International Static Analysis Symposium", Santa Barbara, USA, giugno/luglio 2000.
25. ESSLLI-2000, "Twelfth European Summer School in Logic, Language and Information", student session, Birmingham, Regno Unito, agosto 2000.
26. APPIA-GULP-PRODE'00, "2000 Joint Conference on Declarative Programming", La Havana, Cuba, dicembre 2000.
27. LPAR 2000, "Seventh International Conference on Logic for Programming and Automated Reasoning", Isola della Réunion, Oceano Indiano, novembre 2000.
28. LPAR 2001, "Eighth International Conference on Logic for Programming, Artificial Intelligence and Reasoning", La Havana, Cuba, dicembre 2001.
29. SAS'02, "Ninth International Static Analysis Symposium", Madrid, Spagna, settembre 2002.
30. ESOP'03, "European Symposium on Programming", Varsavia, Polonia, aprile 2003.
31. SAS'03, "Tenth International Static Analysis Symposium", San Diego, California, USA, giugno 2003.
32. PSI'03, "Perspectives of System Informatics", Novosibirsk, Akademgorodok, Russia, luglio 2003.
33. APPIA-GULP-PRODE'03, "2003 Joint Conference on Declarative Programming", Reggio Calabria, Italia, settembre 2003.
34. WLPE'03, "13th Workshop on Logic Programming Environments", Mumbai, India, dicembre 2003.

35. VMCAI'04, "Fifth International Conference on Verification, Model Checking and Abstract Interpretation", Venezia, Italia, gennaio 2004.
36. ESOP'04, "European Symposium on Programming", Barcellona, Spagna, marzo–aprile 2004.
37. SAS'04, "Eleventh International Static Analysis Symposium", Verona, Italia, agosto 2004.
38. CAV'05, "Seventeenth International Conference on Computer Aided Verification", The University of Edinburgh, Scozia, UK, luglio 2005.
39. SAS'05, "Twelfth International Static Analysis Symposium", Londra, Regno Unito, settembre 2005.
40. LOPSTR'05, "International Symposium on Logic-based Program Synthesis and Transformation", Londra, Regno Unito, settembre 2005.
41. ICLP'05, "Twenty First International Conference on Logic Programming", Barcellona, Spagna, ottobre 2005.
42. LOPSTR 2006, "International Symposium on Logic-based Program Synthesis and Transformation", S. Servolo, Venezia, Italia, luglio 2006.
43. POPL 2007, "Thirtyfourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages", Nizza, Francia, gennaio 2007.
44. ESOP'07, "Sixteenth European Symposium on Programming", Braga, Portogallo, marzo, 2007.
45. SAS 2007, "Fourteenth International Static Analysis Symposium", Kongens Lyngby, Danimarca, agosto 2007.
46. SAC 2008, "Twentythird Annual ACM Symposium on Applied Computing", *Technical Track on Software Verification*, Fortaleza, Ceará, Brasile, marzo 2008.
47. VMCAI 2008, "Nineth International Conference on Verification, Model Checking, and Abstract Interpretation", San Francisco, USA, gennaio 2008.
48. SAS 2008, "Fifteenth International Static Analysis Symposium", Valencia, Spagna, luglio 2008.

4.11. Partecipazione a progetti di ricerca

4.11.1. Progetti Terminati o in Corso

1. Partecipa al progetto ESPRIT Basic Research Action Project n. 6707, "ParForce".
2. Partecipa al programma di ricerca cofinanziato dal MURST intitolato "Certificazione automatica di programmi mediante interpretazione astratta" (1999–2001, coordinatore nazionale: Prof. Roberto Giacobazzi, Università di Verona).

3. Partecipa al programma di ricerca cofinanziato dal MURST intitolato “Interpretazione astratta, sistemi di tipo e analisi control-flow” (2000–2002, coordinatore nazionale: Prof. Giorgio Levi, Università di Pisa).
4. Nell’ambito delle *Azioni Integrate Italia-Spagna 2001* coordina, per la parte italiana, il progetto biennale dal titolo “Ambienti avanzati per lo sviluppo di programmi logici” (IT229). Coordinatore per la parte spagnola: Prof. Germán Puebla Sánchez, *Facultad de Informática, Universidad Politécnica de Madrid*.
5. Partecipa al programma di ricerca cofinanziato dal MURST intitolato “Ragionamento su aggregati e numeri a supporto della programmazione e relative verifiche” (2001–2003, coordinatore nazionale: Prof. Domenico Cantone, Università di Catania).
6. Partecipa, come responsabile dell’unità di Parma al programma di ricerca cofinanziato dal MURST intitolato “Verifica di sistemi reattivi basata su vincoli” (2002–2004, coordinatore nazionale: Prof. Maurizio Gabbrielli, Università di Bologna).
7. Partecipa, come responsabile dell’unità di Parma al programma di ricerca cofinanziato dal MIUR intitolato “AIDA — Interpretazione Astratta: Progettazione e Applicazioni” (2004–2006, coordinatore nazionale: Prof. Roberto Giacobazzi, Università di Verona).
8. Nell’ambito del programma ITEA (*Information Technology for European Advancement*), è responsabile dell’unità di Parma del progetto GlobalGCC (GGCC, <http://www.ggcc.info/>), il cui obiettivo è quello di estendere la *GNU Compiler Collection* (GCC) con tecniche di analisi statica globale.
9. Partecipa, come responsabile dell’unità di Parma al programma di ricerca cofinanziato dal MIUR intitolato “AIDA2007 — Interpretazione Astratta: Progettazione e Applicazioni” (2008–2010, coordinatore nazionale: Prof. Francesco Ranzato, Università di Padova).

4.11.2. Progetti Presentati

1. Ha partecipato, come rappresentante nazionale per l’Italia e come coordinatore (insieme a François Irigoien, *École des Mines de Paris*) del *workpackage* su *Numerical Abstract Domains*, alla proposta IST (VI programma quadro) intitolata “AINoE: Network of Excellence on Abstract Interpretation” (004456). Tale proposta non è stata finanziata.
2. Nell’ambito del programma del programma ITEA2 (*Information Technology for European Advancement*), e come responsabile dell’unità che fa capo all’Università di Parma, ha presentato —insieme ad ADACORE, Bertin Technologies, CEA LIST, Fundació European Software Institute (ESI), META, NXP Semiconductors, THALES Communications France, Universidad Politécnica de Madrid, Universidad de Las Palmas, Open Sistemas, VTT Technical Research Centre of Finland— il progetto SAFE-GCC (*Static Analysis Free & Everywhere for GCC*). Tale proposta è in corso di valutazione.

4.12. Partecipazione a scuole

1. “Fifth International School for Computer Science Researchers”, Lipari, Italia, 21 giugno – 3 luglio 1993.
2. NATO Advanced Study Institute “CONSTRAINT PROGRAMMING”, Pärnu, Estonia, 13–24 agosto 1993.
3. “2000 International Summer School in Computational Logic”, Acquafredda di Maratea, Basilicata, Italia, 3–8 settembre 2000.

4.13. Partecipazione a conferenze e workshop nazionali

- GULP’91, “Sesto Congresso sulla Programmazione Logica”, Pisa, giugno 1991.
- COMPULOG 2 meeting, Roma, dicembre 1992. Presentazione del lavoro:

BAGNARA, R., GIACOBAZZI, R., AND LEVI, G. Applications of Constraint Propagation to Data-Flow Analysis.
- GULP’93, “Ottavo Convegno sulla Programmazione Logica”, Gizzeria Lido (CZ), giugno 1993.
- Workshop progetto nazionale “Modelli della Computazione e dei Linguaggi di Programmazione”, Volterra (PI), settembre 1993. Presentazione del lavoro:

BAGNARA, R. Detection of Future Redundant Constraints in $CLP(\mathcal{R})$.
- ICTCS’05, “Ninth Italian Conference on Theoretical Computer Science”, Certosa di Pontignano (Siena), ottobre, 2005.
- “Sesto Workshop Automotive SPIN Italia”, Milano, dicembre 2009.

4.14. Partecipazione a conferenze e workshop internazionali

- “Third International Workshop on Exstensions of Logic Programming”, Bologna, Italia, febbraio 1992.
- WSA’92, “Second International Workshop on Static Analysis”, Bordeaux, Francia, settembre 1992.
- ALP’92, “Third International Conference on Algebraic and Logic Programming”, Volterra, Italia, settembre 1992.
- CAIA’93, “Ninth IEEE Conference on Artificial Intelligence for Applications”, Orlando (Florida), Stati Uniti, marzo 1993.
- WSA’93, “Third International Workshop on Static Analysis”, Padova, Italia, settembre 1993.
- Workshop su “Constraints for Program Analysis”, Århus, Danimarca, febbraio 1994. Presentazione del lavoro:

CURRICULUM VITAE ET STUDIORUM DI ROBERTO BAGNARA

BAGNARA, R. Detection of Redundant Numeric Constraints in CLP Languages.

- ICLP'94, "International Conference on Logic Programming", S. Margherita Ligure, giugno 1994.
- GULP-PRODE'94, "Joint Conference on Declarative Programming", Peñíscola, Spagna, settembre 1994.
- CCP'95, "First International Workshop on Concurrent Constraint Programming", Cà Dolfin, Venezia, maggio 1995.
- GULP-PRODE'95, "Joint Conference on Declarative Programming", Marina di Vietri, settembre 1995.
- "Special Workshop on Abstract Interpretation of Logic Languages", Eilat, Israele, giugno 1995.
- APPIA-GULP-PRODE'96, "1996 Joint Conference on Declarative Programming", Donostia-San Sebastián, Spagna, luglio 1996.
- PLILP'96, "Eighth International Symposium on Programming Languages, Implementations, Logics, and Programs", Aachen, Germania, settembre 1996.
- SAS'96, "Third International Static Analysis Symposium", Aachen, Germania, settembre 1996.
- SAS'97, "Third International Static Analysis Symposium", Parigi, Francia, settembre 1997.
- APPIA-GULP-PRODE'98, "1998 Joint Conference on Declarative Programming", A Coruña, Spagna, luglio 1998.
- SAS'98, "Fifth International Static Analysis Symposium", Pisa, Italia, settembre 1998.
- PLILP/ALP'98, "Tenth International Symposium on Programming Languages, Implementations, Logics and Programs" and "Seventh International Conference on Algebraic and Logic Programming", Pisa, Italia, settembre 1998.
- AMAST'98, "Seventh International Conference on Algebraic Methodology and Software Technology" Amazzonia, Brasile, gennaio 1999.
- APPIA-GULP-PRODE'99, "1999 Joint Conference on Declarative Programming", L'Aquila, Italia, settembre 1999.
- SAS'99, "International Static Analysis Symposium", Venezia, Italia, settembre 1999.
- PPDP'99, "International Conference on Principles and Practice of Declarative Programming", Parigi, Francia, settembre/ottobre 1999.
- PPDP'00, "Second International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming", Montreal, Canada, settembre 2000.

- LPAR 2000, “Seventh International Conference on Logic for Programming and Automated Reasoning”, Réunion, Oceano Indiano, novembre 2000.
- SAS 2001, “Eighth International Symposium on Static Analysis”, Parigi, Francia, luglio 2001.
- APPIA-GULP-PRODE 2001, “2001 Joint Conference on Declarative Programming”, Évora, Portogallo, settembre 2001.
- SAS’02, “Ninth International Static Analysis Symposium”, Madrid, Spagna, settembre 2002.
- ITCLS’02, “Implementation Technology for Computational Logic Systems”, Madrid, Spagna, settembre 2002.
- AVoCS 2003, “Third Workshop on Automated Verification of Critical Systems”, Southampton, Regno Unito, aprile 2003.
- SAS’03, “Tenth International Static Analysis Symposium”, San Diego, California, USA, giugno 2003.
- ITCLS’03, “Implementation Technology for Computational Logic Systems”, Pisa, Italia, settembre, 2003.
- VMCAI’04, “Fifth International Conference on Verification, Model Checking and Abstract Interpretation”, Venezia, Italia, gennaio 2004.
- ETAPS’04, “European Joint Conferences on Theory and Practice of Software”, Barcellona, Spagna, marzo–aprile 2004.
- ICLP’04, “Twentieth International Conference on Logic Programming”, Saint-Malo, Francia, settembre 2004.
- SAS’05, “Twelfth International Static Analysis Symposium”, Londra, Regno Unito, settembre 2005.
- LOPSTR’05, “Fifteenth International Symposium on Logic-based Program Synthesis and Transformation”, Londra, Regno Unito, settembre 2005.
- FMICS 2009, “Fourteenth International Workshop on Formal Methods for Industrial Critical Systems”, Eindhoven, Olanda, novembre 2009.
- GROW’10, “2nd International Workshop on GCC Research Opportunities”, Pisa, Italia, gennaio 2010.

5. Elenco delle pubblicazioni

Le pubblicazioni elencate, tranne alcuni rapporti tecnici, sono disponibili all’URI <http://www.cs.unipr.it/bagnara>. I rapporti tecnici sono disponibili all’URI <http://www.cs.unipr.it>.

5.1. Riviste internazionali

- [1] BAGNARA, R. A unified proof for the convergence of Jacobi and Gauss-Seidel methods. *SIAM Review* 37, 1 (1995), 93–97.
- [2] BAGNARA, R. A hierarchy of constraint systems for data-flow analysis of constraint logic-based languages. *Science of Computer Programming* 30, 1–2 (1998), 119–155.
- [3] BAGNARA, R., HILL, P. M., AND ZAFFANELLA, E. Set-sharing is redundant for pair-sharing. *Theoretical Computer Science* 277, 1-2 (2002), 3–46.
- [4] HILL, P. M., BAGNARA, R., AND ZAFFANELLA, E. Soundness, idempotence and commutativity of set-sharing. *Theory and Practice of Logic Programming* 2, 2 (2002), 155–201.
- [5] ZAFFANELLA, E., HILL, P. M., AND BAGNARA, R. Decomposing non-redundant sharing by complementation. *Theory and Practice of Logic Programming* 2, 2 (2002), 233–261.
- [6] HILL, P. M., ZAFFANELLA, E., AND BAGNARA, R. 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* 4, 3 (2004), 289–323.
- [7] BAGNARA, R., GORI, R., HILL, P. M., AND ZAFFANELLA, E. Finite-tree analysis for constraint logic-based languages. *Information and Computation* 193, 2 (2004), 84–116.
- [8] BAGNARA, R., ZAFFANELLA, E., AND HILL, P. M. Enhanced sharing analysis techniques: A comprehensive evaluation. *Theory and Practice of Logic Programming* 5, 1&2 (2005), 1–43.
- [9] MESNARD, F., AND BAGNARA, R. cTI: A constraint-based termination inference tool for ISO-Prolog. *Theory and Practice of Logic Programming* 5, 1&2 (2005), 243–257.
- [10] BAGNARA, R., HILL, P. M., RICCI, E., AND ZAFFANELLA, E. Precise widening operators for convex polyhedra. *Science of Computer Programming* 58, 1–2 (2005), 28–56.
- [11] BAGNARA, R., HILL, P. M., AND ZAFFANELLA, E. Not necessarily closed convex polyhedra and the double description method. *Formal Aspects of Computing* 17, 2 (2005), 222–257.
- [12] BAGNARA, R., HILL, P. M., AND ZAFFANELLA, E. Widening operators for powerset domains. *Software Tools for Technology Transfer* 8, 4/5 (2006), 449–466.
- [13] BAGNARA, R., HILL, P. M., AND ZAFFANELLA, E. 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* 72, 1–2 (2008), 3–21.

- [14] BAGNARA, R., HILL, P. M., AND ZAFFANELLA, E. Applications of polyhedral computations to the analysis and verification of hardware and software systems. *Theoretical Computer Science* 410, 46 (2009), 4672–4691.
- [15] BAGNARA, R., HILL, P. M., AND ZAFFANELLA, E. Weakly-relational shapes for numeric abstractions: Improved algorithms and proofs of correctness. *Formal Methods in System Design* 35, 3 (2009), 279–323.
- [16] BAGNARA, R., HILL, P. M., AND ZAFFANELLA, E. Exact join detection for convex polyhedra and other numerical abstractions. To appear on *Computational Geometry: Theory and Applications* 43, 5 (2010), 453–473.

5.2. Atti di conferenze internazionali

- [17] PERRIN, Y., BAGNARA, R., BERNERS-LEE, T. J., CARENA, W., DIVIA, R., PARKMAN, C., PETERSEN, J., TREMBLET, L., AND WESSELS, B. The VALET-PLUS embedded into large physics experiments. In *Proceedings of the “VMEbus in Research International Conference”* (Zürich, Oct. 1988), C. Eck and C. Parkman, Eds., Elsevier Science Publishers B.V. (North-Holland), Amsterdam, pp. 59–68.
- [18] HEYES, G., WESSELS, B., PERRIN, Y., BAGNARA, R., BERNERS-LEE, T. J., CARENA, W., DIVIA, R., PARKMAN, C., PETERSEN, J., AND TREMBLET, L. The integration of VAX and VALET-PLUS data acquisition software. Contribution to the “Sixth Conference on Real-Time Computer Applications in Nuclear, Particle, and Plasma Physics” (Williamsburg, VA, May 1989). *IEEE Transactions on Nuclear Science* 36, 5 (1989), 1572–1576.
- [19] BAGNARA, R., GIACOBAZZI, R., AND LEVI, G. An application of constraint propagation to data-flow analysis. In *Proceedings of “The Ninth Conference on Artificial Intelligence for Applications”* (Orlando, Florida, March 1993), IEEE Computer Society Press, Los Alamitos, CA, pp. 270–276.
- [20] BAGNARA, R. A reactive implementation of *Pos* using ROBDDs. In *Programming Languages: Implementations, Logics and Programs, Proceedings of the Eighth International Symposium* (Aachen, Germany, 1996), H. Kuchen and S. D. Swierstra, Eds., vol. 1140 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 107–121.
- [21] BAGNARA, R., HILL, P. M., AND ZAFFANELLA, E. Set-sharing is redundant for pair-sharing. In *Static Analysis: Proceedings of the 4th International Symposium* (École Normale Supérieure, Paris, France, 1997), P. Van Hentenryck, Ed., vol. 1302 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 53–67.
- [22] HILL, P. M., BAGNARA, R., AND ZAFFANELLA, E. The correctness of set-sharing. In *Static Analysis: Proceedings of the 5th International Symposium* (Pisa, Italy, 1998), G. Levi, Ed., vol. 1503 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 99–114.

- [23] BAGNARA, R., AND SCHACHTE, P. Factorizing equivalent variable pairs in ROBDD-based implementations of *Pos*. In *Proceedings of the "Seventh International Conference on Algebraic Methodology and Software Technology (AMAST'98)"* (Amazonia, Brazil, 1999), A. M. Haeberer, Ed., vol. 1548 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 471–485.
- [24] ZAFFANELLA, E., BAGNARA, R., AND HILL, P. M. Widening Sharing. In *Principles and Practice of Declarative Programming* (Paris, France, 1999), G. Nadathur, Ed., vol. 1702 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 414–431.
- [25] ZAFFANELLA, E., HILL, P. M., AND BAGNARA, R. Decomposing non-redundant sharing by complementation. In *Static Analysis: Proceedings of the 6th International Symposium* (Venice, Italy, 1999), A. Cortesi and G. Filé, Eds., vol. 1694 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 69–84.
- [26] BAGNARA, R., ZAFFANELLA, E., AND HILL, P. M. Enhanced sharing analysis techniques: A comprehensive evaluation. In *Proceedings of the 2nd International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming* (Montreal, Canada, 2000), M. Gabbrielli and F. Pfenning, Eds., Association for Computing Machinery, pp. 103–114.
- [27] BAGNARA, R., HILL, P. M., AND ZAFFANELLA, E. Efficient structural information analysis for real CLP languages. In *Proceedings of the 7th International Conference on Logic for Programming and Automated Reasoning (LPAR 2000)* (Reunion Island, France, 2000), M. Parigot and A. Voronkov, Eds., vol. 1955 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 189–206.
- [28] BAGNARA, R., GORI, R., HILL, P. M., AND ZAFFANELLA, E. Finite-tree analysis for constraint logic-based languages. In *Static Analysis: Proceedings of the 8th International Symposium (SAS 2001)*, (Paris, France, 2001), P. Cousot, Ed., vol. 2126 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 165–184.
- [29] BAGNARA, R., ZAFFANELLA, E., GORI, R., AND HILL, P. M. Boolean functions for finite-tree dependencies. In *Proceedings of the 8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 2001)* (Havana, Cuba, 2001), R. Nieuwenhuis and A. Voronkov, Eds., vol. 2250 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, Berlin, pp. 575–589.
- [30] BAGNARA, R., RICCI, E., ZAFFANELLA, E., AND HILL, P. M. Possibly not closed convex polyhedra and the Parma Polyhedra Library. In *Static Analysis: Proceedings of the 9th International Symposium* (Madrid, Spain, 2002), M. V. Hermenegildo and G. Puebla, Eds., vol. 2477 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 213–229.
- [31] BAGNARA, R., HILL, P. M., RICCI, E., AND ZAFFANELLA, E. Precise widening operators for convex polyhedra. In *Static Analysis: Proceedings of the 10th International Symposium* (San Diego, California, USA, 2003),

- R. Cousot, Ed., vol. 2694 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 337–354.
- [32] BAGNARA, R., HILL, P. M., AND ZAFFANELLA, E. Widening operators for powerset domains. In *Proceedings of the Fifth International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI 2004)* (Venice, Italy, 2003), B. Steffen and G. Levi, Eds., vol. 2937 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 135–148.
- [33] BAGNARA, R., HILL, P. M., MAZZI, E., AND ZAFFANELLA, E. Widening operators for weakly-relational numeric abstractions. In *Static Analysis: Proceedings of the 12th International Symposium* (London, UK, 2005), C. Hankin and I. Siveroni, Eds., vol. 3672 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 3–18.
- [34] BAGNARA, R., RODRÍGUEZ-CARBONELL, E., AND ZAFFANELLA, E. Generation of basic semi-algebraic invariants using convex polyhedra. In *Static Analysis: Proceedings of the 12th International Symposium* (London, UK, 2005), C. Hankin and I. Siveroni, Eds., vol. 3672 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 19–34.
- [35] BAGNARA, R., DOBSON, K., HILL, P. M., MUNDELL, M., AND ZAFFANELLA, E. Grids: A domain for analyzing the distribution of numerical values. In *Logic-based Program Synthesis and Transformation, 16th International Symposium* (Venice, Italy, 2007), G. Puebla, Ed., vol. 4407 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 219–235.
- [36] BAGNARA, R., HILL, P. M., AND ZAFFANELLA, E. An improved tight closure algorithm for integer octagonal constraints. In *Verification, Model Checking and Abstract Interpretation: Proceedings of the 9th International Conference (VMCAI 2008)* (San Francisco, USA, 2008), F. Logozzo, D. Peled, and L. Zuck, Eds., vol. 4905 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 8–21.

5.3. Atti di workshop internazionali

- [37] BAGNARA, R., GIACOBAZZI, R., AND LEVI, G. Static analysis of CLP programs over numeric domains. In *Actes “Workshop on Static Analysis ’92”* (Bordeaux, September 1992), M. Billaud, P. Castéran, M. Corsini, K. Musumbu, and A. Rauzy, Eds., vol. 81–82 of *Bigre*, Atelier Irisa, IRISA Campus de Beaulieu, pp. 43–50.
- [38] BAGNARA, R. On the detection of implicit and redundant numeric constraints in CLP programs. In *Proceedings of the “1994 Joint Conference on Declarative Programming (GULP-PRODE’94)”* (Peñíscola, Spain, September 1994), M. Alpuente, R. Barbuti, and I. Ramos, Eds., pp. 312–326.
- [39] BAGNARA, R. Constraint systems for pattern analysis of constraint logic-based languages. In *Proceedings of the “1995 Joint Conference on Declarative Programming (GULP-PRODE’95)”* (Marina di Vietri, Italy, September 1995), M. Alpuente and M. I. Sessa, Eds., pp. 581–592.

- [40] BAGNARA, R. Straight ROBDDs are not the best for *Pos*. In *Proceedings of the “1996 Joint Conference on Declarative Programming (APPIA-GULP-PRODE’96)”* (Donostia-San Sebastián, Spain, 1996), Lucio, P., Martelli, M., and Navarro, M., Eds., pp. 493–496.
- [41] BAGNARA, R., COMINI, M., SCOZZARI, F., AND ZAFFANELLA, E. The *AND*-compositionality of CLP computed answer constraints. In *Proceedings of the “1996 Joint Conference on Declarative Programming (APPIA-GULP-PRODE’96)”* (Donostia-San Sebastián, Spain, 1996), Lucio, P., Martelli, M., and Navarro, M., Eds., pp. 355–366.
- [42] BAGNARA, R. Structural information analysis for CLP languages. In *Proceedings of the “1997 Joint Conference on Declarative Programming (APPIA-GULP-PRODE’97)”* (Grado, Italy, 1997), Falaschi, M., Navarro, M., and Policriti, A., Eds., pp. 81–92.
- [43] BAGNARA, R., HILL, P. M., AND ZAFFANELLA, E. Sharing revisited. In *Proceedings of the “1997 Joint Conference on Declarative Programming (APPIA-GULP-PRODE’97)”* (Grado, Italy, 1997), Falaschi, M., Navarro, M., and Policriti, A., Eds., pp. 69–80.
- [44] BAGNARA, R., AND SCHACHTE, P. Factorizing equivalent variable pairs in ROBDD-based implementations of *Pos*. In *Proceedings of the “1998 Joint Conference on Declarative Programming (APPIA-GULP-PRODE’98)”* (A Coruña, Spain, 1998), Freire-Nistal, J. L., Falaschi, M., and Vilares-Ferro, M., Eds., pp. 227–239.
- [45] HILL, P. M., BAGNARA, R., AND ZAFFANELLA, E. The correctness of set-sharing. In *Proceedings of the “1998 Joint Conference on Declarative Programming (APPIA-GULP-PRODE’98)”* (A Coruña, Spain, 1998), Freire-Nistal, J. L., Falaschi, M., and Vilares-Ferro, M., Eds., pp. 255–267.
- [46] BAGNARA, R., ZAFFANELLA, E., AND HILL, P. M. Enhancing Sharing for precision. In *Proceedings of the “APPIA-GULP-PRODE’99 Joint Conference on Declarative Programming”* (L’Aquila, Italy, 1999), Meo, M. C., and Ferro, M. V., Eds., pp. 213–227.
- [47] ZAFFANELLA, E., BAGNARA, R., AND HILL, P. M. Widening Sharing. In *Proceedings of the “APPIA-GULP-PRODE’99 Joint Conference on Declarative Programming”* (L’Aquila, Italy, 1999), Meo, M. C., and Ferro, M. V., Eds., pp. 559–573.
- [48] BAGNARA, R., HILL, P. M., AND ZAFFANELLA, E. A new encoding of not necessarily closed convex polyhedra. In *Proceedings of the 1st CoLogNet Workshop on Component-based Software Development and Implementation Technology for Computational Logic Systems* (Madrid, Spain, 2002), M. Carro, C. Vacheret, and K.-K. Lau, Eds., pp. 147–153. Published as TR Number CLIP4/02.0, Universidad Politécnica de Madrid, Facultad de Informática.
- [49] BAGNARA, R., HILL, P. M., AND ZAFFANELLA, E. A new encoding and implementation of not necessarily closed convex polyhedra. In *Proceedings of the 3rd Workshop on Automated Verification of Critical Systems*

(Southampton, UK, 2003), M. Leuschel, S. Gruner, and S. Lo Presti, Eds., pp. 161–176. Published as TR Number DSSE-TR-2003-2, University of Southampton.

- [50] BAGNARA, R., HILL, P. M., PESCEZZI, A., AND ZAFFANELLA, E. Verification of C programs via natural semantics and abstract interpretation. In *Proceedings of the C/C++ Verification Workshop* (Oxford, UK, 2007), H. Tews, Ed., pp. 75–80. Extended abstract. Published as Technical Report ICIS-R07015, Institute for Computing and Information Sciences (iCIS), Radboud University Nijmegen, Nijmegen, The Netherlands.
- [51] BAGNARA, R., HILL, P. M., AND ZAFFANELLA, E. A Prolog-based environment for reasoning about programming languages. Tech. Rep. [arXiv:cs.PL/0711.0345](https://arxiv.org/abs/cs.PL/0711.0345), Dipartimento di Matematica, Università di Parma, Italy, 2007. Extended abstract. Contribution to the *17th International Workshop on “Logic-based methods in Programming Environments”* (WLPE’07, Porto, Portugal, September 13, 2007). Available from <http://arxiv.org/>.

5.4. Curatele

- [52] BAGNARA, R., AND MACCHI JÁNICA, G., Eds. *Open Source, Free Software e Open Format nei processi di ricerca archeologici (Atti del I Workshop)* (2007), Centro Editoriale Toscano, Firenze.

5.5. Tesi di dottorato

- [53] BAGNARA, R. *Data-Flow Analysis for Constraint Logic-Based Languages*. PhD thesis, Dipartimento di Informatica, Università di Pisa, Corso Italia 40, I-56125 Pisa, Italy, Mar. 1997. Printed as Report TD-1/97.

5.6. Rapporti tecnici

- [54] BAGNARA, R., HILL, P. M., AND ZAFFANELLA, E. Sharing revisited. Tech. Rep. 97.19, School of Computer Studies, University of Leeds, U.K., 1997.
- [55] BAGNARA, R., HILL, P. M., AND ZAFFANELLA, E. Set-sharing is redundant for pair-sharing. Quaderno 172, Dipartimento di Matematica, Università di Parma, Italy, 1998. Available at <http://www.cs.unipr.it/Publications/>.
- [56] ZAFFANELLA, E., BAGNARA, R., AND HILL, P. M. Widening set-sharing. Quaderno 188, Dipartimento di Matematica, Università di Parma, 1999.
- [57] ZAFFANELLA, E., HILL, P. M., AND BAGNARA, R. Decomposing non-redundant sharing by complementation. Tech. Rep. 99.07, School of Computer Studies, University of Leeds, U.K., 1999.

- [58] BAGNARA, R., HILL, P. M., AND ZAFFANELLA, E. Efficient structural information analysis for real CLP languages. Quaderno 229, Dipartimento di Matematica, Università di Parma, 2000. Available at <http://www.cs.unipr.it/bagnara/>.
- [59] BAGNARA, R., GORI, R., HILL, P. M., AND ZAFFANELLA, E., Finite-tree analysis for constraint logic-based languages, Quaderno 251, Dipartimento di Matematica, Università di Parma, 2001. Available at <http://www.cs.unipr.it/bagnara/>.
- [60] BAGNARA, R., ZAFFANELLA, E., GORI, R., AND HILL, P. M., Boolean functions for finite-tree dependencies, Quaderno 252, Dipartimento di Matematica, Università di Parma, 2001. Available at <http://www.cs.unipr.it/bagnara/>.
- [61] HILL, P. M., ZAFFANELLA, E., AND BAGNARA, R. A correct, precise and efficient integration of set-sharing, freeness and linearity for the analysis of finite and rational tree languages. Quaderno 273, Dipartimento di Matematica, Università di Parma, Italy, 2001. Available at <http://www.cs.unipr.it/Publications/>. Also published as technical report No. 2001.22, School of Computing, University of Leeds, U.K.
- [62] BAGNARA, R., AND CARRO, M. Foreign language interfaces for Prolog: A terse survey. Quaderno 283, Dipartimento di Matematica, Università di Parma, Italy, 2002. Version 1. Available at <http://www.cs.unipr.it/Publications/>.
- [63] R. Bagnara, K. Dobson, P. M. Hill, M. Mundell, and E. Zaffanella. A linear domain for analyzing the distribution of numerical values. Technical Report 2005.06, School of Computing, University of Leeds, U.K., 2005. Available at <http://www.comp.leeds.ac.uk/research/pubs/reports.shtml>.
- [64] BAGNARA, R., RICCI, E., ZAFFANELLA, E., AND HILL, P. M. Possibly not closed convex polyhedra and the Parma Polyhedra Library. Quaderno 286, Dipartimento di Matematica, Università di Parma, Italy, 2002. Available at <http://www.cs.unipr.it/Publications/>.
- [65] BAGNARA, R., HILL, P. M., AND ZAFFANELLA, E. A new encoding and implementation of not necessarily closed convex polyhedra. Quaderno 305, Dipartimento di Matematica, Università di Parma, Italy, 2002. Available at <http://www.cs.unipr.it/Publications/>.
- [66] HILL, P. M., ZAFFANELLA, E., AND BAGNARA, R. On the analysis of set-sharing, freeness and linearity for finite and rational tree languages. Tech. Rep. 2003.08, School of Computing, University of Leeds, U.K., 2003. Available at <http://www.comp.leeds.ac.uk/research/pubs/reports.shtml>.
- [67] BAGNARA, R., HILL, P. M., RICCI, E., AND ZAFFANELLA, E. Precise widening operators for convex polyhedra. Quaderno 312, Dipartimento di Matematica, Università di Parma, Italy, 2003. Available at <http://www.cs.unipr.it/Publications/>.

- [68] BAGNARA, R., ZACCAGNINI, A., AND ZOLO, T. The automatic solution of recurrence relations. I. Linear recurrences of finite order with constant coefficients. Quaderno 334, Dipartimento di Matematica, Università di Parma, Italy, 2003. Available at <http://www.cs.unipr.it/Publications/>.
- [69] BAGNARA, R., AND ZACCAGNINI, A. Checking and bounding the solutions of some recurrence relations. Quaderno 344, Dipartimento di Matematica, Università di Parma, Italy, 2004. Available at <http://www.cs.unipr.it/Publications/>.
- [70] BAGNARA, R., HILL, P. M., AND ZAFFANELLA, E. Widening operators for powerset domains. Quaderno 349, Dipartimento di Matematica, Università di Parma, Italy, 2004. Available at <http://www.cs.unipr.it/Publications/>.
- [71] BAGNARA, R., GORI, R., HILL, P. M., AND ZAFFANELLA, E. Finite-tree analysis for constraint logic-based languages: the complete unabridged version. Quaderno 363, Dipartimento di Matematica, Università di Parma, Italy, 2004. Available at <http://www.cs.unipr.it/Publications/>.
- [72] BAGNARA, R., HILL, P. M., MAZZI, E., AND ZAFFANELLA, E. Widening operators for weakly-relational numeric abstractions. Report [arXiv:cs.PL/0412043](http://arxiv.org/abs/cs.PL/0412043), 2004. Extended abstract. Contribution to the *International workshop on "Numerical & Symbolic Abstract Domains"* (NSAD'05, Paris, January 21, 2005). Available at <http://arxiv.org/> and <http://www.cs.unipr.it/pp1/>.
- [73] BAGNARA, R., HILL, P. M., MAZZI, E., AND ZAFFANELLA, E. Widening operators for weakly-relational numeric abstractions. Quaderno 399, Dipartimento di Matematica, Università di Parma, Italy, 2005. Available at <http://www.cs.unipr.it/Publications/>.
- [74] BAGNARA, R., RODRÍGUEZ-CARBONELL, E., AND ZAFFANELLA, E. Generation of basic semi-algebraic invariants using convex polyhedra. Report de recerca LSI-05-14-R, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Barcelona, Spain, 2005. Available at <http://www.lsi.upc.es/dept/techreps/techreps.html>.
- [75] BAGNARA, R., HILL, P. M., AND ZAFFANELLA, E. 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, Italy, 2006. Available at <http://www.cs.unipr.it/Publications/>. Also published as [arXiv:cs.MS/0612085](http://arxiv.org/abs/cs.MS/0612085), available from <http://arxiv.org/>.
- [76] BAGNARA, R., HILL, P. M., AND ZAFFANELLA, E. Applications of polyhedral computations to the analysis and verification of hardware and software systems. Quaderno 458, Dipartimento di Matematica, Università di Parma, Italy, 2007. Available at <http://www.cs.unipr.it/Publications/>. Also published as [arXiv:cs.CG/0701122](http://arxiv.org/abs/cs.CG/0701122), available from <http://arxiv.org/>.
- [77] BAGNARA, R., HILL, P. M., AND ZAFFANELLA, E. An improved tight closure algorithm for integer octagonal constraints. Quaderno

467, Dipartimento di Matematica, Università di Parma, Italy, 2007. Available at <http://www.cs.unipr.it/Publications/>. Also published as arXiv:0705.4618v2 [cs.DS], available from <http://arxiv.org/>.

- [78] BAGNARA, R., HILL, P. M., PESCHETTI, A., AND ZAFFANELLA, E. On the design of generic static analyzers for imperative languages. Quaderno 485, Dipartimento di Matematica, Università di Parma, Italy, 2008. Available at <http://www.cs.unipr.it/Publications/>.
- [79] BAGNARA, R., HILL, P. M., AND ZAFFANELLA, E. Exact join detection for convex polyhedra and other numerical abstractions. Quaderno 492, Dipartimento di Matematica, Università di Parma, Italy, 2009. Available at <http://www.cs.unipr.it/Publications/>. Also published as arXiv:0904.1783v2 [cs.CG], available from <http://arxiv.org/>.

5.7. Lavori sottomessi a riviste internazionali

- [80] COVINGTON, M. A., BAGNARA, R., O'KEEFE, R. A., WIELEMAKER, J., AND PRICE, S. Coding guidelines for Prolog. Submitted to *Theory and Practice of Logic Programming*. Also published as arXiv:cs.PL/0911.2899, available from <http://arxiv.org/>, 2009.

5.8. Altri scritti

- [81] BAGNARA, R. Announcing Kermit68K, a Portable 68000 Kermit Program. *Info-Kermit Digest 6*, 15 (July 1987).
- [82] BAGNARA, R. *A General Event Handling System for the VALET-PLUS*. CERN Data Handling Division, Online Computing Group, September 1988.
- [83] BAGNARA, R. Remote Procedure Call. *CERN Mini & Micro Computer Newsletter 20* (October 1988).
- [84] BAGNARA, R. Interpretazione astratta di linguaggi logici con vincoli su domini finiti. M.Sc. dissertation, University of Pisa, July 1992. In Italian.
- [85] BAGNARA, R. Analysing with CHINA. *The Association for Logic Programming Newsletter 12*, 1 (1999), 9.
- [86] BAGNARA, R. Is the ISO Prolog standard taken seriously? *The Association for Logic Programming Newsletter 12*, 1 (1999), 10–12.
- [87] BAGNARA, R. On the quality of available Prolog implementations. *The Association for Logic Programming Newsletter 12*, 2 (1999), 12–14.
- [88] BAGNARA, R. Precise and Practical Mode Analysis with the CHINA Analyzer. *Computational Logic 7*, November 1999.
- [89] BAGNARA, R., HILL, P. M., AND ZAFFANELLA, E. *The Parma Polyhedra Library User's Manual*, release 0.10.2 ed. Department of Mathematics, University of Parma, Parma, Italy, Apr. 2009. Available at <http://www.cs.unipr.it/pp1/>.