MINISTERO DELL'ISTRUZIONE, DELL'UNIVERSITÀ E DELLA RICERCA

DIPARTIMENTO PER L'UNIVERSITÀ, L'ALTA FORMAZIONE ARTISTICA, MUSICALE E COREUTICA E PER LA RICERCA SCIENTIFICA E TECNOLOGICA
PROGRAMMI DI RICERCA SCIENTIFICA DI RILEVANTE INTERESSE NAZIONALE
RICHIESTA DI COFINANZIAMENTO (DM n. 287 del 23 febbraio 2005)

PROGETTO DI UNA UNITÀ DI RICERCA - MODELLO B
Anno 2005 - prot. 2005015491_003

PARTE I

1.1 Programma di Ricerca afferente a

1. **Area Scientifico Disciplinare** 01: Scienze matematiche e informatiche 100%

1.2 Durata del Programma di Ricerca

24 Mesi

1.3 Coordinatore Scientifico del Programma di Ricerca

ROSSI **FRANCESCA** **frossi@math.unipd.it**

INF/01 - Informatica

Università degli Studi di PADOVA

Facoltà di SCIENZE MATEMATICHE FISICHE e NATURALI

Dipartimento di MATEMATICA PURA ED APPLICATA

1.4 Responsabile Scientifico dell'Unità di Ricerca

DOVIER **AGOSTINO**

Professore Associato non confermato **28/07/1965** **DVRGTN65L28E125D**

INF/01 - Informatica

Università degli Studi di UDINE

Facoltà di SCIENZE MATEMATICHE FISICHE e NATURALI

Dipartimento di MATEMATICA E INFORMATICA

0432558494 (Prefisso e telefono)	**0432558499** (Numero fax)	**dovier@dimi.uniud.it** (Indirizzo posta elettronica)

1.5 Curriculum scientifico del Responsabile Scientifico dell'Unità di Ricerca

Testo italiano

Agostino Dovier si e' laureato in Scienze dell'Informazione presso l'Universita' di Udine, e ha conseguito nel 1996 il Dottorato di Ricerca in Informatica presso l'Universita' di Pisa, discutendo una tesi intitolata: "Teoria Computabile degli Insiemi e Programmazione Logica".
E' stato ricercatore presso l'Universita' di Verona ed e' attualmente Professore Associato di Informatica presso l'Universita' di Udine.
I suoi interessi di ricerca comprendono la definizione, lo sviluppo e l'utilizzo di linguaggi di programmazione dichiarativi con vincoli, la bioinformatica con particolare riferimento al problema del protein folding, ed il problema del reperimento efficiente di informazioni da basi di dati semistrutturate.
Ha fatto parte di comitati di programma di conferenze nazionali ed internazionali nell'area della programmazione dichiarativa ed e' autore o coautore di piu' 50 pubblicazioni internazionali.

Testo inglese

I received my `Laurea' (Master degree) in `Scienze dell'Informazione' (Computer Science) at the University of Udine, and my PhD degree in `Informatica' (again, Computer Science) from the University of Pisa on October 1996.
The title of my PhD thesis is "Computable Set Theory and Logic Programming".
I was employed as a `ricercatore' (something like `assistant professor') at the University of Verona and I am currently employed as `associate professor' at the University of Udine.
My research interests include: the development and the applications of declarative programming languages with constraints; BioInformatics (in particular the "protein folding problem"); and the problem of fast data retrieval in semistructured Databases.
I have been member of program committees of national and international conferences on declarative programming and I am author or coauthor of more than 50 international publications.

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

1. DOVIER A., PIAZZA C., POLICRITI A. (2004). An efficient algorithm for computing bisimulation equivalence. *THEORETICAL COMPUTER SCIENCE.* vol. 311(1-3) pp. 221-256 ISSN: 0304-3975
2. DOVIER A., PIAZZA. C., PONTELLI E. (2004). Disunification in ACI1 Theories. *CONSTRAINTS.* vol. 9(1) pp. 35-91 ISSN: 1383-7133
3. A. DAL PALU', DOVIER A., AND F. FOGOLARI (2004). Constraint logic programming approach to protein structure prediction. *BMC BIOINFORMATICS.* vol. 5 pp. 1-12 ISSN: 1471-2105
4. F. AVANZINI, D. ROCCHESSO, A. BELUSSI, A. DAL PALU', DOVIER A. (2004). Designing an urban-scale auditory alert system. *COMPUTER.* vol. 37 pp. 55-61 ISSN: 0018-9162

5. DOVIER A., PIAZZA C. (2003). The Subgraph Bisimulation Problem. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING.* vol. 15(4) pp. 1055--1056 ISSN: 1041-4347
6. DOVIER A. (2003). Constraint Approach to the Protein Folding Problem. *AI*IA NOTIZIE.* vol. 2 pp. 17-22
7. CORTESI A., DOVIER A., QUINTARELLI E., TANCA L. (2002). Operational and abstract semantics of a graphical query language. *THEORETICAL COMPUTER SCIENCE.* vol. 275(1/2) pp. 521--560 ISSN: 0304-3975
8. DOVIER A., PONTELLI E., ROSSI G. (2001). Constructive negation and constraint logic programming with sets. *NEW GENERATION COMPUTING.* vol. 19(3) pp. 209-255 ISSN: 0288-3635
9. DOVIER A., PIAZZA C., PONTELLI E., ROSSI G. (2000). Sets and Constraint Logic Programming. *ACM TRANSACTIONS ON PROGRAMMING LANGUAGES AND SYSTEMS.* vol. 22(5) pp. 861--931 ISSN: 0164-0925
10. DOVIER A., PONTELLI E., ROSSI G. (2000). A Necessary Condition for Constructive Negation in Constraint Logic Programming. *INFORMATION PROCESSING LETTERS.* vol. 74 pp. 147--156 ISSN: 0020-0190
11. DOVIER A., OMODEO E., POLICRITI A. (1999). Solvable set/hyperset contexts: II. A goal-driven unification algorithm for the blendedcase. *APPLICABLE ALGEBRA IN ENGINEERING COMMUNICATION AND COMPUTING.* vol. 9 pp. 293-332 ISSN: 0938-1279

1.7 Risorse umane impegnabili nel Programma dell'Unità di Ricerca

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

Personale docente

n°	Cognome	Nome	Dipartimento	Qualifica	Settore Disc.	Mesi Uomo	
						1° anno	2° anno
1.	DOVIER	Agostino	Dip. MATEMATICA E INFORMATICA	Prof. Associato	INF/01	6	6
2.	MONTANARI	Angelo	Dip. MATEMATICA E INFORMATICA	Prof. Ordinario	INF/01	4	4
3.	PIAZZA	Carla	Dip. MATEMATICA E INFORMATICA	Ricercatore Universitario	INF/01	4	4
	TOTALE					**14**	**14**

Altro personale

nº	Cognome	Nome	Dipartimento	Qualifica	Mesi Uomo 1° anno	Mesi Uomo 2° anno
1.	Vitacolonna	Nicola	Dip. MATEMATICA E INFORMATICA	Assegnista	3	
2.	Gentilini	Raffaella	Dip. MATEMATICA E INFORMATICA	Assegnista	3	
	TOTALE				**6**	**0**

1.7.2 Personale universitario di altre Università

Personale docente

nº	Cognome	Nome	Università	Dipartimento	Qualifica	Settore Disc.	Mesi Uomo 1° anno	Mesi Uomo 2° anno
1.	ROSSI	Gianfranco	PARMA	Dip. MATEMATICA	PO	INF/01	5	6
	TOTALE						**5**	**6**

Altro personale

Nessuno

1.7.3 Titolari di assegni di ricerca

Nessuno

1.7.4 Titolari di borse

nº	Cognome	Nome	Dipartimento	Anno di inizio borsa	Durata (in anni)	Tipologia	Mesi Uomo 1° anno	Mesi Uomo 2° anno
1.	Bortolussi	Luca	Dip. MATEMATICA E INFORMATICA	2004	3	Dottorato	6	6
2.	Bresolin	Davide	Dip. MATEMATICA E INFORMATICA	2004	3	Dottorato	6	6
3.	Casagrande	Alberto	Dip. MATEMATICA E INFORMATICA	2003	3	Dottorato	6	3
4.	Dal Palù	Alessandro	Dip. MATEMATICA E INFORMATICA	2003	3	Dottorato	6	3
5.	Puppis	Gabriele	Dip. MATEMATICA E INFORMATICA	2003	3	Dottorato	6	3
6.	Scalabrin	Simone	Dip. MATEMATICA E INFORMATICA	2004	3	Dottorato	6	6
7.	Zantoni	Marco	Dip. MATEMATICA E INFORMATICA	2004	3	Dottorato	6	6
	TOTALE						**42**	**33**

1.7.5 Personale a contratto da destinare a questo specifico programma

nº	Qualifica	Costo previsto	Mesi Uomo 1° anno	2° anno	Note
1.	Assegnista	18.000	6	6	
2.	Borsista	9.000	3	3	
3.	Borsista	6.000	4		
	TOTALE	**33.000**	**13**	**9**	

1.7.6 Personale extrauniversitario indipendente o dipendente da altri Enti

nº	Cognome	Nome	Nome dell'ente	Qualifica	Mesi Uomo 1° anno	2° anno
1.	Pontelli	Enrico	New Mexico State University	Professore Associato	3	3
	TOTALE				**3**	**3**

PARTE II

2.1 Titolo specifico del programma svolto dall'Unità di Ricerca

Testo italiano

Vincoli per la programmazione con insiemi, l'analisi di sistemi con automi, il ragionamento su intervalli e la bioinformatica.

Testo inglese

Constraints for programming with sets, automata-based system analysis, interval reasoning, and bioinformatics.

2.2 Settori scientifico-disciplinari interessati dal Programma di Ricerca

INF/01 - Informatica

2.3 Parole chiave

Testo italiano

PROGRAMMAZIONE CON VINCOLI ; PROGRAMMAZIONE CON INSIEMI ; RAGIONAMENTO CON INTERVALLI ; BIOINFORMATICA ; VERIFICA DI SISTEMI

Testo inglese

CONSTRAINT PROGRAMMING ; SET BASED PROGRAMMING ; INTERVAL REASONING ; BIOINFORMATICS ; SYSTEMS VERIFICATION

2.4 Base di partenza scientifica nazionale o internazionale

Testo italiano

La Programmazione con Vincoli [JM94,MS98,Apt03] è una metodologia di programmazione che permette di formalizzare dichiarativamente i problemi da risolvere con l'ausilio di formule logiche dette vincoli. La fase di computazione è basata sulla ricerca di una o più soluzioni che soddisfino l'insieme di vincoli imposti. Quest'ultima fase è indipendente da quella di formalizzazione ma viene anch'essa guidata dai vincoli che devono essere rispettati, al fine di consentire una riduzione significativa dello spazio di ricerca.

Un risolutore di vincoli è una procedura effettiva in grado di determinare la doddisfacibilità di vincoli rispetto ad una struttura di interpretazione, oppure di rscrivere insiemi di vincoli in insiemi di vincoli più semplici ma con le stesse soluzioni. Diversi risolutori sono stati proposti dagli anni '80 in poi, per vari domini di vincoli. Molti di questi sono stati proposti nell'ambito della Programmazione Logica con Vincoli (CLP), dove attualmente i sistemi più utilizzati sono SICStus Prolog [SICS] ed ECLiPSe [ECLI]. Tali sistemi mettono a disposizione svariati domini di computazione e corrispondenti risolutori, e sono stati impiegati con successo nella risoluzione di problemi di ottimizzazione o di ricerca di soluzioni ammissibili per problemi in domini complessi e con funzioni obiettivo arbitrarie. Risolutori di vincoli sono presenti anche in ambienti di programmazione più tradizionali, quali ad esempio l'ILOG Solver [PL95,ILOG01], dove vincoli e variabili sono trattati come oggetti e definiti all'interno di una libreria di classi C++. Mozart/Oz [moz99,Smo95] è un linguaggio con vincoli che permette di programmare in stile imperativo, orientato agli oggetti, e logico. Altre proposte sono quelle di Jsolver [Chu99] e JSetL [RP04], entrambe basate sul linguaggio Java invece che sul C++. In tutte queste proposte i risolutori di vincoli sono visti come librerie del linguaggio ospite, che possono essere estese con una relativa facilità. Un approccio ortogonale all'introduzione di vincoli in un contesto di programmazione convenzionale consiste nella definizione di un nuovo linguaggio di programmazione in cui i vincoli siano "cittadini di prima classe" del linguaggio stesso. Questa è la soluzione adottata, ad esempio, nei linguaggi Alma-0 [AS00], Singleton [Ros02] e DJ (Declarative Java) [Zho99,Zho].

L'utilizzo di vincoli su domini finiti (nel caso di linguaggi logici, CLP(FD)) permette di formulare in modo conveniente problemi di ottimizzazione combinatoria, quali problemi di scheduling, timetabling, etc. [MS98,Apt03]. I risolutori di vincoli per tali domini sono basati sul concetto di "propagazione di vincoli" e sulla ricerca nello spazio delle soluzioni controllata dalla propagazione che permette di rilevare assegnamenti parziali dei valori alle variabili che non possono condurre a soluzioni accettabili. Per problemi di ottimizzazione, ben note tecniche quali la branch-and-bound vengono integrate con la propagazione di vincoli per ridurre il numero di soluzioni da investigare. Infine, per problemi in cui una soluzione approssimata è sufficiente sono presenti tecniche approssimate per la scansione dell'albero di ricerca, quali ad esempio la limited discrepancy search.

Nel seguito metteremo in evidenza alcuni ambiti in cui l'utilizzo dei vincoli risulta di estremo interesse per la rappresentazione e la risoluzione di problemi di rilievo.

1. VINCOLI E INSIEMI
In tale ambito sono state studiate le opportunità concesse dall'utilizzo di vincoli di tipo insiemistico che permettono di sviluppare prototipi effettivi funzionanti per un dato problema mantenendo un elevato livello di astrazione. Ad esempio, il problema di stabilire se un grafo con X1,...,Xn e archi {X_i1,X_j1},...{X_ik,X_jk} possa o meno essere colorato mediante 3 colori a, b e c in modo tale che nessun nodo sia adiacente ad un nodo dello stesso colore può essere espresso mediante l'unico vincolo:
{{X_i1,X_j1},...,{X_ik,X_jk}} = {{a,b},{a,c},{b,c}}

Oltre a diversi risultati nell'area dei risolutori di vincoli per insiemi/iperinsiemi [DPR98,DOP99,DFO05,DPP04,DPR01], membri dell'unità di Udine hanno descritto e implementato il linguaggio logico con vincoli CLP(SET) [DPPR00], recentemente mostrato adatto a modellare e risolvere problemi di security [WWJ04]. In [DDPR03] CLP(SET) è combinato con CLP(FD) per ottenere un linguaggio con vincoli su più domini che trae vantaggio dalla dichiaratività dei vincoli insiemistici e dall'efficienza dei risolutori di vincoli su domini finiti. Membri dell'unità di Udine hanno inoltre recentemente sviluppato linguaggi tradizionali provvisti di vincoli insiemistici quali il Singleton [Ros02] e delle librerie Java per la programmazione con vincoli insiemistica [RP04].

2. VINCOLI E AUTOMI

La verifica automatica di proprietà di sistemi a stati infiniti è un problema rilevante nell'ambito della ricerca in ambito informatico. Esempi significativi di sistemi a stati infiniti sono i sistemi di controllo real-time, i protocolli di sicurezza, i protocolli di comunicazione via canali FIFO, i sistemi parametrizzati e i sistemi ibridi (ai quali dedicheremo particolare attenzione per la loro rilevanza nell'area della systems biology). Un approccio naturale al problema della verifica automatica di sistemi a stati infiniti consiste nella modellazione del sistema come un grafo diretto i cui vertici rappresentano configurazioni del sistema e i cui archi rappresentano transizioni. Il comportamento atteso del sistema è definito attraverso l'unraveling del grafo, vale a dire attraverso un albero bisimile al grafo. Formule della logica temporale, espresse in un opportuno formato (logica temporale modale, logica temporale classica, automi, ...) possono essere usate per esprimere proprietà desiderate del sistema. In tal modo il problema della verifica viene ridotto al problema di stabilire la soddisfacibilità (risp. verità) di una formula (risp. enunciato) rispetto a una data struttura ad albero/grafo. Nel presente progetto concentreremo la nostra attenzione sugli approcci al problema della verifica basati su automi che incorporano una nozione di vincolo, quali gli automi con contatori, che consentono di gestire vincoli di periodicità, gli automi definitivamente periodici, gli automi temporizzati e gli automi ibridi. In particolare, gli Automi Ibridi sono stati introdotti in [Alu92] come linguaggio per la modellazione e la specifica di sistemi ibridi, ovvero sistemi in cui un programma discreto interagisce con un ambiente che evolve nel continuo. Essi sono stati ampiamente utilizzati per la verifica automatica di sistemi sia di tipo ingegneristico che biologico.

3. VINCOLI E INTERVALLI

Gli intervalli occupano un ruolo centrale nell'ambito della rappresentazione e del ragionamento temporali. Proprietà temporali significative di sistemi sviluppati in aree assai diverse, quali, ad esempio, la pianificazione, l'elaborazione del linguaggio naturale, la specifica e la verifica di sistemi hardware e software e la bioinformatica, possono essere descritte in modo naturale utilizzando formalismi basati sugli intervalli. Le logiche temporali basate su intervalli costituiscono uno dei più interessanti e sistematici approcci al trattamento degli intervalli temporali [Gor04]. Mentre vari metodi a tableau sono stati sviluppati per le logiche temporali proposizionali basate sui punti, sia lineari che ramificate, poco è stato fatto per quelle basate sugli intervalli. Goranko e dei membri dell'unità di Udine hanno recentemente proposto un metodo a tableau per la logica CDT interpretata sugli ordini parziali [Gor03b] che combina le caratteristiche dei metodi a tableau classici per la logica al prim'ordine con quelle dei metodi a tableau espliciti per le logiche modali, che prevedono un'apposita componente per la gestione dei vincoli sulle etichette (intervalli, nel caso delle logiche temporali a intervalli). Tale metodo risulta essere estremamente generale e facilmente adattabile alla maggioranza delle logiche temporali a intervalli proposte in letteratura. Un'implementazione di tale metodo è stata fornita in [Gor05].

4. VINCOLI E BIOINFORMATICA

Recentemente diversi problemi di Bioinformatica sono stati
mostrati essere codificabili come problemi di vincoli su domini finiti [CB01]. In particolare, il problema di determinare la conformazione tridimensionale di una proteina una volta nota la sequenza di amino acidi che la compongono (Protein Folding) è stato
codificato come problema vincolato su un dominio discreto (il cubo a facce centrate) in [BW01] usando funzioni di energia approssimate (il modello HP). Tale formalizzazione è stata estesa nel caso di funzioni di energia più precise in [BMC04] alcuni membri
dell'unità di Udine. Tale proposta, pur fornendo risultati incoraggianti, evidenzia dei problemi

legati alla scarsa visibilità delle strutture dati associate alle variabili e ai vincoli concesse all'utente dei linguaggi CLP(FD). Inserire euristiche nella ricerca di soluzioni è altamente inefficiente senza l'accesso diretto alla struttura dati dell'albero di ricerca. In questi lavori si cerca di posizionare l'atomo centrale di ogni amino acido nel reticolo. In [SK01] viene invece studiato sempre come problema CLP(FD) il problema di posizionare le catene laterali di una proteina una
volta che gli atomi centrali siano fissati.

Il problema del protein folding è affrontato in letteratura con vari metodi, ma in particolare mediante simulazione. Tali programmi studiano le interazioni tra gli atomi costituenti gli aminoacidi della proteina. Seppure precise dal punto di vista termodinamico, tali simulazioni sono estremamente lente. Recentemente alcuni membri
dell'unità di Udine hanno mostrato come sia naturale programmare simulazioni in Concurrent Constraint Programming in cui gli elementi di base sono dei processi concorrenti associati ad ogni aminoacido costituente la proteina [BDDF04].

Testo inglese

Constraint Programming [JM94,MS98,Apt03] is a programming methodology where problems are formalized using logic formulas known as "constraints".
Computation is based on the search of solutions that satisfy the set of imposed constraints. The computation phase is independent from the formalization one. However, computation is driven by constraints satisfaction to sensibly reduce the search space.

A constraint solver is an effective procedure that checks the satisfiability of a constraint and/or simplifies a constraint into another constraint with the same solution's set.
Several constraint solvers have been proposed since the eighties, for various constraint domains. Most of them have been developed in the context of Constraint Logic Programming (CLP), where currently the most used systems are SICStus Prolog [SICS] and ECLiPSe [ECLI]. These systems offer several computation domains and corresponding constraint solvers and they have been successfully used for optimization problems and for problems of finding admissible solutions for complex domains and arbitrary objective functions.
Constraint Solvers are embedded in more traditional programming frameworks, such as ILOG Solver [PL95,ILOG01] where constraints and variables are considered as objects and defined within a library of C++ classes. Mozart/Oz [moz99,Smo95] is a constraint-based language where imperative, Object Oriented, and logic programming programming styles are allowed. Other proposals are Jsolver [Chu99] and JSetL [RP04], both based on the Java language instead of on C++.
In all these proposals, constraint solvers are libraries of the host languages, and they can be easily extended. A different approach in imperative programming is that of Alma-0 [AS00], Singleton [Ros02], and DJ (Declarative Java) [Zho99,Zho] where constraints are "first-class citizens" of the host languages.

Constraint programming over finite domains (in the case of logic languages, briefly CLP(FD)) allows to conveniently model combinatoric optimization problems, such as scheduling, timetabling, etc. [MS98,Apt03].
Solvers for those domains are based on "constraint propagation" and on "constraint based" search tree traversing.
As far as optimization problems are concerned, well-known techniques such as branch-and-bound are integrated with constraint propagation. Finally, when approximated solutions are satisfactory (e.g., in a real-life resource allocation problem where a breakdown of a machine or the illness of an operator would make the computed optimum obsolete), approximated techniques such as the limited discrepancy search are incorporated into the solver.

In the following, we will focus on some situations where constraints are extremely interesting for problem representation and solution.

1. CONSTRAINTS AND SETS

In this area, the power of set-based constraints have been investigated. Their use allows to design working prototypes for a given problem, at a high level of abstraction.
For example, the problem of determining whether a graph with nodes X1,...,Xn and edges {X_i1,X_j1},...,{X_ik,X_jk} can be colored with 3 colors a, b, and c so that no nodes are adjacent to another one with the same color, can be expressed by means of the single constraint: {{X_i1,X_j1},...,{X_ik,X_jk}} = {{a,b},{a,c},{b,c}}.

Some members of the Udine team obtained several results in the sets/multisets/hypersets constraint solvers area [DPR98,DOP99,DFO05,DPP04,DPR01]. Moreover, they designed and implemented a constraint logic programming language CLP(SET) [DPPR00] and recently they showed that the language is suitable for modeling and solving security problems [WWJ04].
In [DDPR03] the combination of CLP(SET) and CLP(FD) results in a constraint language over various domains that exploits the declarativity of set-based constraints and the efficiency of finite domains constraint solvers. Members of the Udine team have also recently developed some traditional languages with set constraints, such as Singleton [Ros02] and some Java libraries for set-based constraint [RP04].

2. CONSTRAINTS AND AUTOMATA

The automatic verification of properties of infinite state systems is a relevant problem in computer science research. Meaningful examples of infinite state systems are real-time control systems, security protocols, communication protocols for FIFO channels, parameterized systems, and hybrid systems (we will devote a special attention to the latter given their relevance in systems biology). A natural approach to the problem of automatic verification of infinite state systems is to model a system as a directed graph, whose vertices (resp. edges) represent system configurations (resp. transitions). The expected behavior of the system is defined as the unraveling of the graph, i.e., a tree bisimilar to the graph itself. Temporal logic formulas, expressed in a suitable form (modal temporal logic, classical temporal logic, automata, ...) can then be exploited to express desired properties of the system. In such a way the verification problem reduces to the problem of deciding the satisfiability (resp. truth) of a formula (resp. sentence) over a given graph/tree structure. In this project, we will focus our attention on the approaches to the verification problem based on automata provided with a notion of constraint, such as automata with counters, that manage periodicity constraints, ultimately periodic automata, timed automata, and hybrid automata. In particular, hybrid automata were first introduced in [Alu92] as a model and specification language for hybrid systems, i.e., systems consisting of a discrete program within a continuously changing environment. Since their introduction, they have been widely used for the automatic verification of both engineering and biological systems.

3. CONSTRAINTS AND INTERVALS

Intervals have a central role in temporal representation and reasoning. Temporal properties of systems developed in different areas, such as planning, natural language processing, definition and verification of hardware and software systems, and Bioinformatics can be naturally formalized using interval-based formalisms.
Interval-based temporal-logics represent one of the more interesting systematic approaches to temporal intervals handling [Gor04]. While several tableau-like methods have been developed for point-based propositional temporal logics (either linear or branching time) very few has been done for interval-based logics.
Goranko and some members of the Udine unit have recently proposed a tableau method for the logics CDT interpreted on partial orders [Gor03b].
This method combines the features of classical tableau methods for first-order logics with explicit tableau methods for modal logics, that include a specific component for handling constraints on labels (in this case, intervals). Such a method is extremely general and easy to be adapted on the majority of interval-based temporal-logics in literature. An imperative implementation of this method can be found in [Gor05].

4. CONSTRAINTS AND BIOINFORMATICS

Recently, various Bioinformatics problems have been shown to be successfully implemented as problems over Finite Domains [CB01]. In detail, the problem of determining the three dimensional conformation of a specific protein, whose sequence of composing aminoacids is

known (Protein Folding), has been modeled over a discrete domain (Face Centered Cube) in [BW01] using approximated energy functions (HP model). This formalization has been extended to a more refined energy model in [BMC04] by some members of the Udine unit. The proposal provides encouraging results, but reveals, at the same time, the limits caused by a scarce accessibility of data structures associated to the variables and constraints, when implementing the ideas in CLP(FD) languages.
For proteins made of hundreds of aminoacids, the presence of some problem-dependent search heuristics is fundamental. Unfortunately, the implementation of heuristics is highly inefficient without a direct access to the underlying data structures. In these abstractions, we map the central carbon of each aminoacid to a lattice point. In [SK01], CLP(FD) is again employed to place the side chains of a protein, after each central carbon has been placed.

The protein folding problem is tackled in the literature by various methods, mainly simulations. These programs consider the interactions among the atoms that form the protein. The thermodynamic laws encoded are precise as well as compute-intensive. Recently, some members of the Udine team showed how to naturally design Concurrent Constraint Programming programs for simulation, where the base elements are concurrent processes associated to each aminoacid in the protein [BDDF04].

2.4.a Riferimenti bibliografici

[Alu92] R. Alur, C. Courcoubetis, T. A. Henzinger, and P. H. Ho.
Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems.
Hybrid Systems, LNCS 736:209-229, 1992.

[Ana99] H. Anai.
Algebraic Approach to Analysis of Discrete-Time Polynomial Systems.
European Control Conference (ECC'99), 1999.

[Apt03] K.R. Apt.
Principles of Constraint Programming. Cambridge, 2003.

[AS00] K.R. Apt and A. Schaerf.
Programming in Alma-0, or Imperative and Declarative Programming
FROCOS 2:1-16, 2000.

[BW01] R. Backofen and S. Will.
Fast, constraint-based threading of HP-sequences to hydrophobic cores.
CP'01, LNCS 2239:494-508, 2001.

[Bas97] S. Basu.
An Improved Algorithm for Quantifier Elimination Over Real Closed Fields.
IEEE Symp. on Foundations of Computer Science 1997:56-65.

[proIV] F. Benhamou et al.
Le manuel de Prolog IV, PrologIA, Giugno 1996.

[BDDF04] A. Dal Palù, A. Dovier, F. Fogolari.
Protein Folding Simulation in CCP.
Proc. of ICLP 2004, LNCS 3132, Saint Malo, France.

[CB01] P. Clote and R. Backofen.
Computational Molecular Biology: An Introduction.
John Wiley & Sons, 2001.

[Col90] A. Colmerauer.
An introduction to Prolog III.

Comm. of ACM 33(7):69-90, 1990.

[Chu99] A.Chun. Constraint programming in Java with JSolver.
Proc. of Practical Applications of Constraint Logic Programming, 1999.

[CD96] P.Codognet, D.Diaz.Compiling constraints in CLP(FD).
J. of Logic Programming 27(3):185-226, 1996.

[BMC04] A. Dal Palù, A. Dovier, and F. Fogolari.
Constraint logic programming approach to protein structure prediction.
BMC Bioinformatics, 5(186):1-12, November 2004.

[DDPR03] A. Dal Palù, A. Dovier, E. Pontelli, and G. Rossi.
Integrating Finite Domain Constraints and CLP with Sets.
Proc. of 5thPPDP:219-229. ACM, 2003.

[DC93] D.Diaz, P.Codognet.
A minimal extension of the WAM for CLP(FD).
Proc. of the 10th ICLP, 1993.

[DVSA88] M.Dincba et al.
The constraint logic programming CHIP,
Proc. of the 2nd Int. Conf. on Fifth Generation Computer Systems, 683-702, 1988.

[DPPR00] A.Dovier, C.Piazza, E.Pontelli, G.Rossi.
Sets and Constraint Logic Programming.
ACM Toplas, 22(5) 2000, 861-931.

[DFO05] A. Dovier, A. Formisano, and E.G. Omodeo.
Decidability Results for Sets with Atoms.
ACM TOCL, 2005.

[DOP99] A. Dovier, E.G. Omodeo, and A. Policriti.
Solvable set/hyperset contexts: II. A goal-driven unification algorithm for the blended case.
AAECC 9(4):1-48, 1999.

[DPP04] A. Dovier, C. Piazza, and E. Pontelli.
Disunification in ACI1 Theories.
Constraints, 9(1):35-91, 2004.

[DPR98] A. Dovier, A. Policriti, and G. Rossi.
A Uniform Axiomatic View of Lists, Multisets, and Sets, and the Relevant Unification Algorithms.
Fundamenta Informaticae 36(2/3):201-234, 1998.

[DPR01] A. Dovier, E. Pontelli, and G. Rossi.
Constructive Negation and Constraint Logic Programming with Sets.
New Generation Computing 19(3):209-255, 2001.

[ECLI] ECLiPSe, User Manual. Tech. rep., Imperial
College, London. August 1999. Available at www.icparc.ic.ac.uk/eclipse.

[Fra99] M. Franzle.
Analysis of Hybrid Systems: An ounce of realism can save an infinity of states, in Computer Science Logic
(CSL'99), LNCS 1683:126-140, 1999.

[Fra01] M. Franzle.
What Will Be Eventually True of Polynomial Hybrid Automata?
Proc of TACS 01, LNCS 2215:340-359, 2001.

[Fran04] M. Franzle and C. Herde.
Efficient Proof Engines for Bounded Model Checking of Hybrid Systems.
Proc. of FMICS, 2004.

[GL88] M. Gelfond and V. Lifschitz.
The Stable Model Semantics for Logic Programming.
Proc. of the ICLP'88.

[Gor05] V. Goranko, A. Montanari, P. Sala, and G. Sciavicco.
A general tableau method for propositional interval temporal logics.
J. of Applied Logic, 2005.

[Gor03b] V. Goranko, A. Montanari, and G. Sciavicco.
A general tableau method for propositional interval temporal logics.
Proc. of the International Conference TABLEAUX 2003, LNAI 2796:102-116, 2003.

[Gor03c] V. Goranko, A. Montanari, and G.Sciavicco.
Proof systems for propositional neighborhood logics.
Proc. of the Third Workshop on Methods for Modalities (M4M):125-140, 2003.

[Gor03a] V. Goranko, A. Montanari, and G. Sciavicco.
Propositional interval neighborhood temporal logics.
J. of Universal Computer Science, 9(9):1137-1167, 2003.

[Gor04] V. Goranko, A. Montanari, and G. Sciavicco.
A road map of interval temporal logics and duration calculi.
J. of Applied Non-Classical Logics, 14(1-2):9-54, 2004.

[Gri88] D. Grigoriev.
Complexity of Deciding Tarski Algebra.
J. of Symbolic Computation, vol. 5, 65-108, 1988.

[ILOG01] ILOG Optimisation Suite-White Paper.
http://www.ilog.com/products/optimisation/tech/optimisation/whitepaper.pdf.

[JM94] J. Jaffar and M. J. Maher.
Constraint Logic Programming: A Survey.
J. of Logic Programming, 19/20:503-581, 1994.

[JMSY92] J.Jaffar, S.Michaylov, P.J.Stuckey, and R.H.C.Yap.
The CLP(R) language and system.
ACM TOPLAS 14(3), 1992, 339-395.

[Laf00] G. Lafferriere, G. J. Pappas, and S. Sastry.
O-minimal Hybrid Systems, Mathematics of Control, Signals, and
Systems, 13:1-21, 2000.

[Laf01] G. Lafferriere, G. J. Pappas, and S. Yovine.
Symbolic Reachability Computation for Families of Linear Vector Fields.
J. Symb. Comput., 32(3):231-253, 2001.

[MT91] V.W. Marek and M. Truszczynski.
Autoepistemic Logic. J. of the ACM, 38(3):588-619, 1991.

[MS98] K. Marriott and P. J. Stuckey.
Programming with Constraints. MIT, 1998.

[MR04] M. Milano and A. Roli.
MAGMA: A Multiagent Architecture for Metaheuristics.

IEEE Trans. on Systems, Man and Cybernetics - Part B, 34(2), 2004.

[MSV02] A. Montanari, G. Sciavicco, and N. Vitacolonna.
Decidability of interval temporal logics over split-frames via granularity.
Proc. of European Conference on Logic in Artificial Intelligence 2002, LNAI 2424, 259-270, 2002.

[moz99] Mozart Consortium. The Mozart programming system, 1999.
Disponibile in http://www.mozart-oz.org.

[PM05] C. Piazza and B. Mishra.
Stability of Hybrid Systems and Related Questions from Systems Biology.
Systems & Control: Foundations & Applications, T. Basar Ed., Birkhauser, 2005.

[Pra05] I. Pratt-Hartmann.
Temporal Prepositions and their Logic.
Artificial Intelligence, 2005 (to appear).

[PL95] J.-F. Puget and M. Leconte.
Beyond the Glass Box: Constraints as Objects.
Proc. of the 1995 ILPS, MIT, 513-527.

[Ren92] J. Renegar.
On the Computational Complexity and Geometry of the First-order Theory of the Reals, parts I-III.
J. of Symbolic Computation 13:255-352, 1992.

[RS05] S. Ratschan and Z. She.
Safety verification of hybrid systems by constraint propagation based abstraction refinement.
Hybrid Systems: Computation and Control (HSCC'05), LNCS 3114:573-589, 2005.

[Ros02] G. Rossi.
Set-based Nondeterministic Declarative Programming in SINGLETON.
Proc of 11th WFLP. Grado (IT), June 20-22, 2002.

[RP04] G.Rossi and E.Poleo.
JSetL: Declarative Programming in Java with Sets.
Proc. of CF'04, 2004 ACM International Conference on Computing Frontiers, 2004.

[Smo95] G.Smolka.
The OZ programming model.
Computer Science Today, LNCS 1000, 1995.

[SK01] M. T. Swain and G. J. L. Kemp.
A CLP approach to the protein side-chain placement problem.
Proc. of CP'01, LNCS 2239:479-493, 2001.

[SICS] Swedish Institute of Computer Science.
The SICStus Prolog Home Page. www.sics.se

[Tar51] A. Tarski.
A Decision Method for Elementary Algebra and Geometry.
Univ. California, 1951.

[VSD92] P.Van Hentenryck, H.Simonis, and M.Dincbas.
Constraint satisfaction using constraint logic programming.
Artificial Intelligence 58, 1992, 113-159.

[Vit05] N. Vitacolonna.
Intervals: logics, algorithms, and games.

PhD thesis, Univ. of Udine, DIMI, 2005.

[WWJ04] L. Wang, D. Wijesekera, and S. Jajodia.
A Logic-based Framework for Attribute based Access Control.
Proc. of 2nd ACM Workshop on Formal Methods in Security Engineering (FMSE 2004), 2004, 45-55

[Zho99] N.-F. Zhou.
DJ: Declarative Java, Version 0.5, User's manual.
Kyushu Institute of Tecnology, 1999.http://www.cad.mse.kyutech.ac.jp/people/zhou/dj.htlm.

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

Testo italiano

L'unità di ricerca di Udine articolerà il proprio contributo alla ricerca nei seguenti quattro filoni:

1. VINCOLI E INSIEMI
2. VINCOLI E AUTOMI
3. VINCOLI E INTERVALLI
4. VINCOLI E BIOINFORMATICA

Nel filone dei VINCOLI E INSIEMI si intende procedere ad investigare le problematiche connesse alla progettazione ed alla integrazione di linguaggi di programmazione con vincoli di tipo insiemistico/iperinsiemistico/multiinsiemistico con particolare attenzione agli aspetti computazionali che garantiscano la loro effettiva utilizzabilità. In particolare si desidera analizzare le problematiche relative alla integrazione o cooperazione dei risolutori di vincoli insiemistici di CLP(SET) e a quelli su domini finiti di CLP(FD). Ciò permette di ottenere un ambiente di programmazione con vincoli che permette la dichiaratività della programmazione insiemistica, adatta alla rapida prototipazione del software e l'efficienza data dai risolutori su domini finiti. Si desidera inoltre, usando tecniche di interpretazione astratta, sviluppare dei compilatori di parti di codice CLP(SET) in CLP(FD).

Si desidera inoltre completare e mettere a punto la libreria JSetL, una libreria Java che offre caratteristiche proprie dei linguaggi logici a vincoli, in particolare con vincoli insiemistici, in un ambito di programmazione "object-oriented". In questo contesto verrà anche affrontato il problema della cooperazione di più risolutori di vincoli, utilizzando ed adattando tecniche di "cooperative constraint solving" note in letteratura, allo specifico ambiente offerto da JSetL. Questo porterà tra l'altro ad una riprogettazione dell'architettura complessiva di JSetL, che, sfruttando i meccanismi di multitasking e comunicazione tra processi offerti da Java, permetta una strutturazione del sistema che faciliti l'integrazione tra diversi risolutori di vincoli.

Una metodologia di programmazione dichiarativa che sta riscuotendo un crescente interesse è l'Answer Set Programming (in breve, ASP). I programmi ASP sono programmi logici privi di simboli funzionali ma con arbitraria presenza di letterali negati. Per tali programmi l'esistenza di un modello non garantita; modelli interessanti sono i modelli stabili [GL88] per il calcolo dei quali sono stati sviluppati opportuni risolutori (p. es., SMODELS, DLV, ASSAT). E' dimostrato che la classe dei problemi codificabili mediante ASP equivale esattamente alla classe NP [MT91]. Pertanto nei risolutori di ASP vi sono tecniche di risoluzione simili a quelle utilizzate, ad esempio, nei risolutori di vincoli CLP(FD) che tipicamente affrontano problemi simili. Per la codifica di problemi con vincoli, la metodologia ASP va dunque tenuta in cosiderazione. Nell'ambito del progetto si desidera integrare le idee utilizzate nei risolutori per answer set programming e nei risolutori per vincoli su domini finiti in CLP(FD) in un unico risolutore adatto ad affrontare, in generale, problemi NP-completi.

Nel filone VINCOLI E AUTOMI, studieremo il problema della decidibilità per opportune varianti ed estensioni delle classi di automi considerate nella base di partenza; in particolare, le loro

proprietà di chiusura. Inoltre, affronteremo il problema di determinare le controparti logiche di tali classi di automi. Ciò consentirà di usare le prime come un'interfaccia di alto livello per la specifica dei sistemi e/o delle loro proprietà e i secondi come un formalismo di basso di livello per eseguire la verifica. Dedicheremo particolare attenzione all'utilizzo di tecniche di programmazione con vincoli per gestire in modo efficiente i vincoli (temporali). Oltre che in ambito biologico, intendiamo sperimentare tali strumenti in altri contesti applicativi, quali quello delle basi di dati temporali e spaziali, quello dei sistemi di workflow e quello più tradizionale dei sistemi reattivi e in tempo reale richiamati in precedenza. Intendiamo inoltre sviluppare un confronto sistematico tra le soluzioni da noi proposte, che sfruttano le potenzialità offerte dalla nozione di vincolo, e gli approcci ai sistemi a stati infiniti presenti in letteratura (approccio trasformazionale, sviluppato da Caucal e altri, sistemi di riscrittura, sistemi equazionali, uso di transducer).

Per quanto riguarda lo studio degli automi ibridi, un problema fondamentale è quello della raggiungibilità. Uno stato dell'automa ibrido è individuato da uno stato della componente discreta e da un assegnamento di valori reali alle componenti continue. Si vuole caratterizzare l'insieme degli stati raggiungibili attraverso transizioni discrete e continue dagli stati iniziali [Alu92]. Tra gli approcci proposti per lo studio di automi ibridi emergono approcci simbolici basati sulla traduzione dell'automa in vincoli (formule) sui reali e sullo sviluppo di risolutori per tali vincoli. Gli approcci simbolici si fondano sulla decidibilità delle formule al primo ordine nella teoria dei reali provata da Tarski in [Tar51]. Citiamo come esempi [Ana99] e [Fra99] che hanno indipendentemente suggerito l'utilizzo dell'eliminazione dei quantificatori per la verifica di sistemi ibridi polinomiali. [Fra04] ha sviluppato "sistemi di prova" per il bounded model checking di questi sistemi. [Laf01] propone un metodo basato sull'eliminazione dei quantificatori per il calcolo della raggiungibilità simbolica nel caso di campi vettoriali lineari. [RS05] ha suggerito un nuovo metodo di raffinamento astratto basato su propagazione di vincoli per la verifica di sistemi ibridi con equazioni differenziali autonome.
Tali tecniche di verifica simbolica sono attualmente applicabili a classi molto ristrette di automi ibridi. Spesso si considerano sistemi di equazioni differenziali lineari [Laf01], o comunque sistemi autonomi per i quali è possibile dimostrare esistenza ed unicità della soluzione [Laf00]. In molti casi questi automi ibridi sono traducibili in un unico sistema di equazioni differenziali [PM05] e tecniche sviluppate in analisi possono essere utilizzate per il loro studio. Queste assunzioni non bastano per assicurare la convegenza del processo di traduzione in vincoli [Fran01].
Si intendono pertanto scoprire delle ipotesi sugli automi ibridi che garantiscano la convergenza della traduzione in vincoli sui reali, ovvero ipotesi che garantiscano di ottenere un'unica formula e non una sequenza infinita di formule. Le ipotesi che intendiamo studiare saranno ottenute come combinazione di restrizioni sulla componente continua e sulla componente discreta del sistema. Tale combinazione dovrebbe consentire di studiare anche sistemi di inclusioni differenziali non autonomi. Si intende inoltre analizzare in quali casi i vincoli ottenuti dalla traduzione possono essere verificati in modo efficiente o possono essere approssimati con limitazioni sull'errore. Recentemente sono stati proposti numerosi metodi [Gri88, Ren92, Bas97] che aumentano l'applicabilità del risultato di decidibilità di Tarski [Tar51] abbassandone la complessità per particolari classi di formule. Tuttavia numerose tecniche di geometria algebrica quali le basi di Groebner a gli insiemi caratteristici non sono ancora state pienamente sfruttate.
Infine, si desidera implementare un tool in grado di effettuare la traduzione e la verifica eventualmente interfacciandosi con tool già esistenti per la verifica di vincoli sui reali, come ad esempio QepCad.

Nell'area del filone VINCOLI E INTERVALLI, si intende trasformare il metodo a tableau proposto in [Gor03b,Gor05] in una procedura di decisione per classi particolari di logiche temporali, quali la famiglia delle Split Logic (SL) [MSV02] e la famiglia delle Propositional Neighborhood Logic (PNL) [Gor03a,Gor03c].

In particolare, verranno esplorate opportune restrizioni di tipo sintattico o semantico, che garantiscano la terminazione del processo di costruzione e verifica del tableau. Per quanto riguarda le restizioni sintattiche, si imporranno delle condizioni sul numero e la tipologia degli operatori temporali utilizzati; per quanto riguarda le restrizioni semantiche, si imporranno opportune condizioni sulla struttura temporale (strutture finite, strutture split, strutture

discrete, numeri naturali) e/o sull'interpretazione (ad esempio, il principio di localita'). Esempi di logiche decidibili ottenute imponendo tali restrizioni sono riportati in [Gor04].

Di recente, abbiamo ottenuto dei risultati preliminari promettenti per il frammento della logica PNL che contiene i soli operatori futuri, vale a dire gli operatori che consentono di predicare solo sugli intervalli alla destra dell'intervallo corrente, interpretato sul dominio dei naturali. L'approccio proposto combina restrizioni sintattiche (operatori temporali: frammento futuro) e restrizioni semantiche (struttura temporale: numeri naturali) e verrà utilizzato com punto di partenza per lo studio della decidibilita' di altre logiche ad intervalli, quali, ad esempio, la logica PNL completa, interpretata sui naturali o su altre classi di strutture temporali, e la logica dei sottointervalli, o logica D, il cui unico operatore temporale consente di esprimere la relazione di inclusione tra intervalli. Studieremo, inoltre, gli effetti dell'introduzione della restrizione alle sole intepretazioni/modelli finite su logiche che, senza tale restrizione, sono state dimostrate indecidibili, sulla base di risultati ottenuti di recente da Pratt [Pra05].

Infine, per supportare in maniera adeguata il lavoro di sviluppo, testing e validazione di tali procedure, verranno utilizzate tecniche e strumenti mutuati dalla programmazione (logica) con vincoli. Le procedure di decisione così ottenute verranno sperimentate in ambito biologico per la specifica e la verifica di rilevanti proprietà temporali, in linea con quanto fatto in [Vit05].

Nel filone VINCOLI E BIOINFORMATICA tre saranno le tematiche principali su cui si focalizzerà il lavoro dell'unità di Udine.

La prima di queste riguarda la predizione della struttura tridimensionale di una proteina data le sequenza di aminoacidi che la compongono. Si intende affrontare tale problema come problema di minimizzazione di una funzione di energia le cui variabili siano le posizioni spaziali dei centri degli aminoacidi che possono assumere valori su alcuni punti di un reticolo discreto, principalmente il Face Centered Cubic (FCC) lattice. La funzione di energia da minimizzare dipende dalle distanze tra i vari aminoacidi e dai tipi di aminoacidi in gioco. In particolare, ci si vuole concentrare sullo sviluppo di un risolutore di vincoli ad hoc per vincoli su reticoli adatti a discretizzare le proteine, al fine di ottimizzare il codice presentato in [BMC04]. Il codice, da svilupparsi in C++, andrà a rimpiazzare il risolutore di vincoli su domini finiti del SICStus Prolog attualmente utilizzato per tale problema. La disponibilità delle strutture dati a basso livello permetterà di programmare in modo efficiente nuove euristiche per il problema. Sarà inoltre possibile utilizzare la metodologia a vincoli per determinare il ripiegamento di grosse proteine costituite da sequenze di proteine note e più piccole. Si tratta, dunque, di vedere queste ultime come corpi rigidi di cui si vuole trovare la posizione spaziale che minimizza la funzione di energia globale. Tale approccio non risulta attualmente praticabile con i risolutori disponibili in SICStus Prolog a causa dell'elevato numero di variabili in gioco. I codici prodotti saranno poi portati su macchina parallela.

La seconda tematica di questo filone riguarda sempre il problema del protein folding, ma sarà affrontato in modo diverso. Anzichè utilizzare un reticolo discreto, si lasceranno le proteine libere di assumere ogni posizione nello spazio e la minimizzazione della funzione d'energia avviene per mezzo di una simulazione concorrente. In questo contesto si desidera investigare l'applicabilità del modello di ottimizzazione multiagente [MR04] al nostro problema e la sua possibile generalizzazione ad altre simulazioni biologiche. In questo contesto si desidera passare da una formalizzazione nel linguaggio di coordinamento Linda ad un modello basato su MPI che permetta una effettiva computazione concorrente su Cluster di PC.

La terza tematica del filone VINCOLI E BIOINFORMATICA è costituita dalla simulazione e verifica di sistemi biologici e sarà basata sull'utilizzo dei risultati del filone VINCOLI E AUTOMI sopra descritto.

Testo inglese

In the scope of the current research project, the Udine research unit will work in the following four research fields:

1. CONSTRAINTS AND SETS
2. CONSTRAINTS AND AUTOMATA
3. CONSTRAINTS AND INTERVALS
4. CONSTRAINTS AND BIOINFORMATICS

In the field CONSTRAINTS and SETS, we will investigate the problem of designing and integrating program languages with constraints over sets/hypersets/multisets, with particular attention to the computational aspects and to the effective usability of these languages.

In particular, we will analyze the problem of the integration and cooperation of the constraint solver over sets of CLP(SET) with the constraint solver over finite domains of CLP(FD). This will allow us to obtain a constraint programming environment that combines declarative set programming, very useful in rapid software prototyping, with the efficiency of finite domains solvers. Using abstract interpretation techniques, we will develop compilers for translating CLP(SET) code into CLP(FD).

Another objective of the project is to conclude the development and to tune the library JSetL. JSetL is a Java library that offers the features of constraint logic programming (in particular, set constraint programming) to object-oriented programming environments. In this context, we will face the problem of the cooperation of different constraint solvers, developing and adapting techniques known in literature as "cooperative constraint solving" to the particular case of JSetL. This will lead to the complete redeveloping and rewriting of the architecture of JSetL, in order to use the multitasking and process communications primitives of Java to ease integration between constraint solvers.

Another interesting declarative programming methodology is Answer Set Programming (ASP, for short). ASP programs are logic programs without function symbols, but with an arbitrary presence of negated literals. The existence of a model for this kind of programs is (in general) not guaranteed. An interesting class of models for ASP programs is the one of stable models [GL88], for which many solvers have been developed (for example, SMODELS, DLV, ASSAT). It has been proved that the class of problems that can be coded by ASP is exactly the class NP [MT91]. Thus, in ASP solvers many techniques coming, for example, from CLP(FD) constraint solvers has been adopted. Since ASP and CLP(FD) face similar problems, it will be fruitful to consider also the ASP approach. In this project, we will aim to the integration of answer set programming solvers with finite domain constraint solvers into a single solver, suited to face NP-complete problems.

In the branch CONSTRAINTS AND AUTOMATA, we will focus our attention on the decidability problem for variants and extensions of the classes of automata introduced in the Scientific Background. In particular, we will study their closure properties. Furthermore, we will address the problem of identifying their temporal logic counterparts. Pairing the logical formalism with the automaton-based one would allow one to use the former as a high-level interface for system and/or property specification and the latter as an internal formalism to implement verification. A special attention will be devoted to the use of constraint programming techniques to efficiently manage (temporal) constraints. Beside the area of life sciences, we intend to test the proposed tools in other application domains, such as, for instance, those of temporal and spatial databases and of workflow systems as well as the above-mentioned traditional area of reactive and real-time systems.
Furthermore, we will systematically compare the proposed solution, that take advantage of the potentialities of the notion of constrain, and the approaches to the specification and verification of infinite state systems in the literature (the transformational approach, developed by Caucal and others, rewriting systems, equational systems, transducers).

As for hybrid automata, one of the main focus is the so-called reachability problem. A state of a hybrid automaton is specified by providing a state of the discrete component and an assignment for the continuous variables. The problem is that of characterizing the set of states reachable through continuous and discrete transitions from the initial states [Alu92].
Recently, symbolic methods based on the translation of the automaton into constraints (formulae) over the reals and on the development of specific constraints solvers in the study of hybrid automata emerged.

Such approaches exploit Tarski's decidability result for the first-order theory over the real numbers [Tar51].
For instance, [Ana99] and [Fra99] independently suggested the use of quantifier elimination for the verification of polynomial hybrid systems. [Fran04] developed "proof engines" for bounded model checking of these systems.
[Laf01] described a method based upon quantifier elimination for symbolic reachability computation of linear vector fields.
[RS05] have suggested a new constraint propagation based abstraction refinement for the safety verification of hybrid systems with autonomous differential equations.
These symbolic verification techniques are applicable only to restricted classes of hybrid automata. In many cases only linear differential equations [Laf01] or autonomous systems with a unique solution [Laf00] are allowed. These hybrid automata can be usually mapped into one system of differential equations [PM05] and techniques developed in real analysis can be applied to their study. These restrictions are not enough to ensure the convergence of the constraint translation process [Fran01].
Hence, we intend to study conditions which guarantee the convergence of the translation of hybrid automata into constraints over the reals, i.e., conditions which guarantee that the translation produces one formula instead of an infinite sequence of formulae. The conditions we plan to study should be combinations of restrictions on the continuous and discrete components of the system. Such combinations should allow to study also non-autonomous systems of differential inclusions.
Moreover, we intend to analyze the computational complexity of the constraints we get from the translation process and to classify classes of constraints which can be efficiently verified/approximated. Recently, a number of methods [Gri88, Ren92, Bas97] which augment the applicability of Tarski's decidability result [Tar51] have been proposed. However, many algebraic geometry techniques, such as Groebner basis and characteristic sets, have not yet been exploited.
To complete our project we should implement a tool to automatize the translation and verification process. The tool could exploit already existent modules for symbolic real computations such as QepCad.

As far the area of CONSTRAINTS AND INTERVALS is concerned, our work will aim at transforming the tableau method proposed in [Gor03b,Gor05] into a decision procedure for specific classes of interval temporal logics, such as, for instance, the family of Split Logics (SL) [MSV02] and that of Propositional Neighborhood Logics (PNL) [Gor03a,Gor03c].

In particular, we will investigate suitable syntactic and semantic restrictions, that guarantee the termination of the process of tableau construction and verification. As for syntactic restrictions, we will constrain the number and type of temporal operators to be used; as for semantic ones, we will impose suitable constraints on the temporal structure (finite structures, split structures, discrete structures, natural numbers) as well as on the semantic interpretation (e.g., the locality principle). Examples of decidable interval temporal logics obtained by imposing suitable syntactic and/or semantic restrictions have been reported in [Gor04].

Promising preliminary results have been obtained for the future fragment of PNL, that is, for the PNL fragment whose operators only allow one to talk about the truth of formulas over intervals to the right of the current one, interpreted over the domain of natural numbers. The proposed approach combines syntactic restrictions (temporal operators: future fragment) and semantic ones (temporal structure: natural numbers) and it will be used as a starting point for the investigation of the decidability problem for other interval temporal logics, such as, for instance, full PNL, interpreted over natural numbers or other meaningful classes of temporal structures, and the logics of subintervals (the so-called D logics), provided with a unique operator that encodes the inclusion relation between pairs of intervals. Furthermore, we will study the effects of restricting to finite interpretations/models only on logics that in the general case turned out to be undecidable, taking advantage of recent results by Pratt [Pra05].

Finally, to adequately support the processes of development, testing, and validation of these

procedures, we will use techniques and tools borrowed from constraint (logic) programming. The obtained decision procedures will be tested in a Bioinformatics setting for the specification and verification of relevant temporal properties, along the directions outlined in [Vit05].

Within the CONSTRAINTS AND BIOINFORMATICS area, the work of the Udine research unit will focus on three main branches.

The first of them is the prediction of the three dimensional structure of a protein, given its sequence of amino acids that form it. We plan to model the problem as the minimization of an energy function, whose variables are the spatial position of the centers of the amino acids. Each variable ranges over the points of a discrete lattice, mainly the Face Centered Cubic (FCC) lattice. The energy function to be minimized depends on the distances between amino acids and on their type. In detail, we want to focus on the development of an ad-hoc constraint solver for constraints over lattices that are suitable for protein discretization, in order to optimize the code presented in [BMC04]. The new code, written in C++, should replace the constraint solver over finite domains in SICStus Prolog, currently employed for this problem. The accessibility of data structures at low level will allow to encode efficiently new heuristics for the problem. It will be also possible to use the constraint methodology to determine the folding of big complexes made of sequences of known and smaller proteins. The problem here is to model the components as rigid elements and to place them in the space, while minimizing the global energy function. This approach is currently limited by solvers contained in SICStus Prolog, because of the high number of variables in use. The new code will be ported on a parallel machine.

The second branch of this research area is again about the protein folding problem, tackled with a different model. Instead of using a discrete lattice, the proteins are free to move in every spatial position in the continuous space. The energy function minimization is realized by a concurrent simulation. In this context, we plan to investigate the applicability of the multi-agent optimization model [MR04] to our problem and to generalize it to other biological simulations. In detail, we want to move from a formalization with Linda coordination language to a model based on MPI that allows an effective concurrent computation on a PC cluster.

The third branch of this research area is the simulation and checking of biological systems. It will be based on the results of CONSTRAINT AND SYSTEMS described above.

2.6 Descrizione delle attrezzature già disponibili ed utilizzabili per la ricerca proposta con valore patrimoniale superiore a 25.000 Euro

Testo italiano

Nessuna

Testo inglese

Nessuna

2.7 Descrizione delle Grandi attrezzature da acquisire (GA)

Testo italiano

Nessuna

Testo inglese

Nessuna

2.8 Mesi uomo complessivi dedicati al programma

Testo italiano

		Numero	Mesi uomo 1° anno	Mesi uomo 2° anno	Totale mesi uomo
Personale universitario dell'Università sede dell'Unità di Ricerca		5	20	14	34
Personale universitario di altre Università		1	5	6	11
Titolari di assegni di ricerca		0			
Titolari di borse	Dottorato	7	42	33	75
	Post-dottorato	0			
	Scuola di Specializzazione	0			
Personale a contratto	Assegnisti	1	6	6	12
	Borsisti	2	7	3	10
	Dottorandi	0			
	Altre tipologie	0			
Personale extrauniversitario		1	3	3	6
TOTALE		**17**	**83**	**65**	**148**

Testo inglese

		Numero	Mesi uomo 1° anno	Mesi uomo 2° anno	Totale mesi uomo
University Personnel		5	20	14	34
Other University Personnel		1	5	6	11
Work contract (research grants, free lance contracts)		0			
PHD Fellows & PHD Students	PHD Students	7	42	33	75
	Post-Doctoral Fellows	0			
	Specialization School	0			
Personnel to be hired	Work contract (research grants, free lance contracts)	1	6	6	12
	PHD Fellows & PHD Students	2	7	3	10
	PHD Students	0			
	Other tipologies	0			
No cost Non University Personnel		1	3	3	6
TOTALE		**17**	**83**	**65**	**148**

PARTE III

3.1 Costo complessivo del Programma dell'Unità di Ricerca

Testo italiano

Voce di spesa	Spesa in Euro	Descrizione
Materiale inventariabile	14.000	Acquisto di PC desktop o laptop per i partecipanti al progetto e per il personale a contratto richiesto.
Grandi Attrezzature		
Materiale di consumo e funzionamento	1.000	Acquisto di cancelleria, carta e toner per stampanti per i due anni.
Spese per calcolo ed elaborazione dati		
Personale a contratto	33.000	Si desidera bandire un assegno di ricerca annuale per dottorandi/dottori di ricerca e altri collaboratori (ad esempio neolaureati)per ulteriori 6 mesi e 4 mesi.
Servizi esterni		
Missioni	18.000	Spese per periodi di missione all'italia e all'estero anche in concomitanza con conferenze.
Pubblicazioni		
Partecipazione / Organizzazione convegni	4.000	Spese per la partecipazione a conferenze nazionali o internazionali.
Altro		
TOTALE	**70.000**	

Testo inglese

Voce di spesa	Spesa in Euro	Descrizione
Materiale inventariabile	14.000	Purchase of laptop/desktp PC for the participants to the project and the required extra researchers.
Grandi Attrezzature		
Materiale di consumo e funzionamento	1.000	Purchase of stationary, paper and toner for printers in the two years of the project.
Spese per calcolo ed elaborazione dati		
Personale a contratto	33.000	One post-doctoral or doctoral position (for one year) and some supports (6 months and 4 months) for graduated students.
Servizi esterni		
Missioni	18.000	National and International missions for the members of the reasearch unit.
Pubblicazioni		

Partecipazione / Organizzazione convegni	4.000	Funds for participation to national and international conferences.
Altro		
TOTALE	**70.000**	

3.2 Costo complessivo del Programma di Ricerca

		Descrizione
Costo complessivo del Programma dell'Unità di Ricerca	70.000	
Fondi disponibili (RD + RA) ***comprensivi dell'8% max per spese di gestione***	21.000	5250 a cura dei proponenti e 15750 a carico dell'università di Udine
Cofinanziamento di altre amministrazioni		
Cofinanziamento richiesto al MIUR	49.000	

3.3.1 Certifico la dichiarata disponibilità e l'utilizzabilità dei fondi di Ateneo (RD e RA)

SI

(per la copia da depositare presso l'Ateneo e per l'assenso alla diffusione via Internet delle informazioni riguardanti i programmi finanziati e la loro elaborazione necessaria alle valutazioni; D. Lgs, 196 del 30.6.2003 sulla "Tutela dei dati personali")

Firma _____ Data 05/04/2005 ore 15:24

.