



UNIVERSITÀ DELLA CALABRIA
DIPARTIMENTO DI
MATEMATICA
E INFORMATICA

Al Direttore del Dipartimento
di Matematica e Informatica

Oggetto: Richiesta Rinnovo per Assegno di Ricerca
Dott. Veltri Pierfrancesco – SSD INF/01

Il sottoscritto, prof. Nicola Leone,
PREMESSO

che il Dott. Pierfrancesco Veltri è stato titolare di assegno di ricerca per 12 mesi (Bando di Concorso del 02.04.2015 emanato con D.D. 21-2015) e che l'attività di ricerca ha ottenuto ottimi risultati, il cui sviluppo ha evidenziato importanti applicazioni future,

RICHIEDE

il rinnovo dell'assegno di ricerca per ulteriori 12 mesi per l'approfondimento di ricerche correlate al potenziamento del sistema di intelligenza artificiale DLV attraverso lo sviluppo del modulo DLV³. La spesa graverà sui fondi di ricerca intestati al professore Nicola Leone.

Rende, lì 31/03/2016

IL RESPONSABILE SCIENTIFICO
(prof. Nicola LEONE)

IL TITOLARE DEI FONDI DI
RICERCA
(prof. Nicola LEONE)



UNIVERSITÀ DELLA CALABRIA

DIPARTIMENTO DI
MATEMATICA
E INFORMATICA

ATTESTAZIONE DEL TUTOR

Il sottoscritto Prof. Nicola Leone, tutor scientifico del Dott. Pierfrancesco Veltri per l'Assegno per la collaborazione ad attività di ricerca vinto con il progetto di ricerca intitolato "Tecniche ed algoritmi per la service innovation" (Bando di Concorso del 02/04/2015 emanato con D.D. n. 21-2015)

ATTESTA

la regolarità, qualità e coerenza delle attività svolte dal Dott. Pierfrancesco Veltri con il progetto di ricerca finanziato e gli eccellenti risultati ottenuti.

Rende, lì 31/03/2016

IL RESPONSABILE SCIENTIFICO

Prof. Nicola Leone

investiamo nel vostro futuro

PROGRAMMA OPERATIVO NAZIONALE RICERCA E COMPETITIVITÀ - 2007/2013

ASSE I: "SOSTEGNO AI MUTAMENTI STRUTTURALI"

OBIETTIVO OPERATIVO: "RETI PER IL RAFFORZAMENTO DEL POTENZIALE SCIENTIFICO-TECNOLOGICO DELLE REGIONI DELLA CONVERGENZA"

I AZIONE: "DISTRETTI DI ALTA TECNOLOGIA E RELATIVE RETI"

II AZIONE: "LABORATORI PUBBLICO-PRIVATI E RELATIVE RETI"

PROGETTO PON03PE_00001_1

SETTORE : ICT/TECNOLOGIE PER SMART COMMUNITIES

CUP : H24G14000370005

Responsabile Scientifico Progetto PON03PE_00001_1 *Leone Nicola*
Responsabile Amministrativo Progetto PON03PE_00001_1 *Massaro Pasquale*

Assegno di Ricerca **TECNICHE E ALGORITMI PER LA SERVICE INNOVATION**
Dipartimento di MATEMATICA E INFORMATICA dell'Università della Calabria
Anno Accademico 2015/2016
Periodo di svolgimento 01/11/2015 - 01/05/2016

RELAZIONE FINALE

Titolare Assegno di Ricerca PON03PE_00001_1 **VELTRI Pierfrancesco**
Supervisore/Referente Scientifico dell'Attività di Ricerca **LEONE Nicola**
Direttore Dipartimento prof. **LEONE Nicola**

Ente ospitante estero _____

Paese estero _____

Tutor/Referente estero _____ (Cognome e Nome) _____ (per l'eventuale periodo all'estero)

La presente relazione è composta
di n. 7 pagine numerate
da 1 a 7

investiamo nel vostro futuro

Logistica, attrezzature

Durante questo semestre ho svolto il mio lavoro di ricerca nello studio presso il Cubo 31B, Secondo Piano, Dipartimento di Matematica, Università della Calabria. Ho inoltre usufruito delle seguenti attrezzature per testare le performance dei prodotti di ricerca sviluppati:

1. Legione, una macchina con 2 processori Intel Xeon "Woodcrest" (quad core) 3GHz con 4MB di L2 Cache e 4GB di RAM, con sistema operativo Debian GNU Linux 4.0.
2. Efeso, un insieme di macchine a 4 core Intel Xeon CPU X3430 2.4 Ghz e 4GB di RAM, con sistema operativo Linux Debian Lenny (32bit).

Attività svolte e risultati ottenuti

Al giorno d'oggi, l'utilizzo delle ontologie è largamente diffuso in moltissime applicazioni reali, specialmente per quanto riguarda il *knowledge and data management*. In questo contesto, il problema dell'*ontology-based Query Answering* (QA) sta assumendo un ruolo via via sempre più rilevante [4, 5, 6, 7]. Tale rilevanza è testimoniata anche dallo spiccato interesse mostrato, di recente, da varie case produttrici di sistemi per la gestione di basi di dati. In particolare, sistemi commerciali come, per esempio, *Oracle*, *Ontotext* ed *Ontoprise*, oggi offrono all'utente la possibilità di fare ontology-based QA. Anche nel mondo accademico, della ricerca scientifica, sono numerevoli i sistemi che permettono di fare "ragionamento ontologico". Tra questi, possiamo citare, per esempio, *QuOnto* [1], *FaCT++* [2], e *Nyaya* [3].

In generale, il termine "ontologia" si usa per indicare la conoscenza generale di un dominio, la quale, a volte, viene anche chiamata conoscenza terminologica (TBox), in modo che possa essere chiaramente distinta dalla conoscenza asserzionale, che qui indicheremo anche come "dataset". Così, data una *base di conoscenza* $KB = (\Sigma, D)$ composta da un'ontologia Σ ed un dataset D , ed una query q , il problema dell'ontology-based QA consiste nel calcolo dell'insieme delle risposte alla query q che possono essere ottenute tenendo conto della conoscenza implicita modellata dall'ontologia.

investiamo nel vostro futuro

Un problema chiave, in questo contesto, è rappresentato dalla scelta del linguaggio da utilizzare per la specifica della teoria ontologica Σ . Tale linguaggio dovrebbe in qualche modo bilanciare espressività e complessità.

Il linguaggio *Datalog* [8] permette di esprimere diversi assiomi ontologici. Purtroppo, però, l'espressività di *Datalog* non è tale da consentire la "creazione" di nuovi valori. Questa caratteristica è stata riconosciuta come cruciale, specialmente in una prospettiva di mondo aperto, dove non si può assumere che tutti gli individui siano conosciuti a priori. Questo ha motivato il rinnovato interesse per *Datalog*³, che è la naturale estensione di *Datalog* in cui si possono rappresentare variabili quantificate esistenzialmente in testa alle regole. Questo linguaggio è altamente espressivo e consente di modellare domini di conoscenza in maniera semplice e potente. Purtroppo, però, la presenza delle variabili esistenziali rende indecidibile, nel caso generale, il ragionamento su *Datalog*³.

Nel primo semestre, però, abbiamo dimostrato che è possibile fare reasoning (e quindi query answering) su programmi *Datalog*³ in maniera potente, nonché decidibile ed efficiente. Infatti, abbiamo introdotto la classe dei programmi *Datalog*³ "parsimoniosi" ed abbiamo dimostrato che fare ragionamento su programmi di questo tipo è decidibile ed anche efficiente. Sfortunatamente, però, abbiamo verificato che riconoscere la proprietà di "parsimonia" è indecidibile. Tuttavia, abbiamo individuato un frammento facilmente riconoscibile di programmi parsimoniosi, chiamato *Shy*, che estende in maniera significativa sia *Datalog* che *Linear Datalog*³ [5]. Inoltre, nonostante l'aggiunta dei quantificatori esistenziali, *Shy* preserva la stessa complessità di *Datalog* per quanto riguarda il query answering.

Quindi, considerate le interessanti peculiarità di questa nuova classe di programmi e constatata la mancanza di sistemi scalabili per il ragionamento su ontologie *Datalog*³, in questo semestre abbiamo implementato una strategia di valutazione bottom-up per programmi *Shy* all'interno del ben noto sistema DLV [9].

Per capire meglio la procedura implementata in DLV, dobbiamo partire dall'analisi di una delle procedure più largamente diffuse per la valutazione di programmi *Datalog*³: il *chase*.

Il *chase* è una procedura forward-chaining che parte dall'insieme vuoto e produce un insieme (possibilmente infinito) di atomi, o meglio un'interpretazione, che rappresenta un

investiamo nel vostro futuro

modello *universale* del programma in input. Quindi, sull'output del chase è possibile fare QA in maniera sound and complete.

Nel dettaglio, il chase parte dall'insieme vuoto ed esegue in maniera esaustiva (cioè finché non raggiunge un punto fisso) la seguente operazione, meglio nota come *chase step*: sia C l'interpretazione parziale costruita fino al passo precedente, per ogni regola del programma r che non è soddisfatta (che ha, quindi, il corpo vero e la testa falsa) rispetto a C , il chase inferisce dei nuovi atomi che soddisfano la testa di r . Gli atomi prodotti dal chase contengono dei nulli in luogo delle variabili esistenziali. E' importante assumere che in corrispondenza di chase step differenti (sulle stesse regole o su regole diverse) i nulli introdotti sono sempre "nuovi".

La proprietà semantica di parsimonia, introdotta durante il primo semestre di attività, si basa su una variante del chase che abbiamo appena descritto: il *parsimonious-chase*.

Il *parsimonious-chase* differisce dal chase solo nel processo di inferenza dei nuovi atomi. Il chase standard inferisce sempre nuovi atomi quando trova delle regole non soddisfatte. Il *parsimonious-chase*, invece, inferisce dei nuovi atomi solo quando trova una regola non soddisfatta ed, allo stesso tempo, non ci sono già nell'interpretazione derivata al passo precedente degli atomi isomorfi a quelli che si andrebbero a generare. Questo è anche il motivo per cui abbiamo deciso di chiamare la procedura "parsimoniosa". Si noti che l'output del *parsimonious-chase* non solo potrebbe non essere un modello universale per il programma in input ma, addirittura, potrebbe non essere affatto un modello.

Durante il primo semestre, abbiamo dimostrato che per i programmi che soddisfano la proprietà di parsimonia (quindi anche per Shy) si può fare atomic query answering in maniera sound and complete considerando semplicemente l'output del *parsimonious-chase*. Per quanto riguarda, invece, il conjunctive query answering abbiamo ideato una procedura che riesuma il *parsimonious-chase* al termine della sua esecuzione, dopo aver promosso tutti i nulli introdotti a costanti. Questo processo di riesumazione viene ripetuto tante volte quante sono le variabili in join nella query in input. La procedura finale fornisce in output un'interpretazione sulla quale si può rispondere alla query in input in maniera sound and complete.

Quella appena descritta è, quindi, la procedura che abbiamo implementato all'interno del ben noto sistema DLV durante questo secondo semestre di attività. In realtà, la

investiamo nel vostro futuro

computazione è stata arricchita attraverso lo sviluppo di una serie di tecniche di ottimizzazione. Il sistema ottenuto è stato chiamato DLV³ - un potente ragionatore in grado di rispondere a query congiuntive su programmi Shy, il quale è applicabile proficuamente al problema dell'ontology-based query answering.

Finora, abbiamo condotto anche delle analisi sperimentali dove abbiamo confrontato DLV³ con i principali sistemi che rappresentano lo stato dell'arte nell'ontology-based query answering. I risultati che abbiamo ottenuto hanno dato chiara evidenza del fatto che DLV³ è il sistema più efficiente per il query answering in ambienti dinamici, dove sia i dati che l'ontologia possono variare di frequente, rendendo inapplicabile ogni forma di pre-computazione e di ottimizzazione statica.

Da quanto ci risulta, inoltre, possiamo affermare che DLV³ è il primo sistema che supporta pienamente la semantica first-order per *unrestricted* CQs con variabili esistenziali su ontologie con proprietà avanzate, come per esempio, la transitività sui ruoli, le *gerarchie* di ruoli, i ruoli *inversi*, ed il *prodotto* di concetti.

I risultati ottenuti durante l'intero anno di attività sono raccolti in una prima bozza di articolo per la rivista scientifica internazionale TOCL (Transactions on Computational Logic). Nel prossimo periodo, prevediamo di estendere l'analisi sperimentale su DLV³ in modo da poter completare l'articolo sopra citato.

Pubblicazioni

La descrizione del lavoro di ricerca condotto finora, insieme ai risultati ottenuti dai test sopra descritti sono oggetto di una bozza di articolo per la rivista internazionale TOCL (Transactions on Computational Logic), il quale è attualmente in via di sviluppo.

Durante l'anno, inoltre, è stato condotto un altro studio parallelamente alle attività sopra descritte. Tale studio ha portato alla sottomissione dell'articolo "Modeling and Reasoning about NTU Games via Answer Set Programming" alla conferenza internazionale IJCAI (International Joint Conference on Artificial Intelligence), una delle più autorevoli conferenze nel campo dell'Intelligenza Artificiale. L'articolo è stato accettato dalla commissione di valutazione e verrà presentato a New York nel mese di Luglio 2016.

investiamo nel vostro futuro

Bibliografia

- [1] Andrea Acciarri, Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Mattia Palmieri, and Riccardo Rosati. QUONTO: querying ontologies. In Proc. of the 20th national conference on Artificial intelligence, volume 4, pages 1670–1671. AAAI Press, 2005.
- [2] Dmitry Tsarkov and Ian Horrocks. FaCT++ Description Logic Reasoner: System Description. In Proc. of the 3rd Int. Joint Conf. on Automated Reasoning, volume 4130 of IJCAR '06, pages 292–297, Seattle, WA, USA., 2006.
- [3] Roberto De Virgilio, Giorgio Orsi, Letizia Tanca, and Riccardo Torlone. Semantic Data Markets: A Flexible Environment for Knowledge Management. In Proc. of the 20th ACM international Conference on Information and Knowledge Management, CIKM '11, New York, NY, USA, 2011. ACM.
- [4] Diego Calvanese, Giuseppe Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family. *J. Autom. Reason.*, 39:385–429, October 2007.
- [5] Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. In Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, PODS '09, pages 77–86, New York, NY, USA, 2009. ACM.
- [6] Ilianna Kolli, Birte Glimm, and Ian Horrocks. Sparql query answering over owl ontologies. In Proceedings of the 24th International Workshop on Description Logics, volume 6643 of Lecture Notes in Computer Science, pages 382–396. Springer Berlin/Heidelberg, 2011.
- [7] Andrea Calì, Georg Gottlob, and Andreas Pieris. New Expressive Languages for Ontological Query Answering. In Proceedings of the 25th AAAI Conference on Artificial Intelligence, pages 1541–1546, 2011.
- [8] Serge Abiteboul, Richard Hull, and Victor Vianu. Foundations of Databases: The Logical Level. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.
- [9] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. The DLV System for Knowledge Representation and Reasoning. *ACM TOCL* 7(3):499–562.

investiamo nel vostro futuro

Data e timbro del Dipartimento,

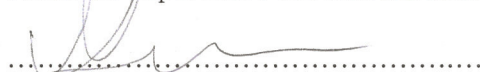
31/03/2016



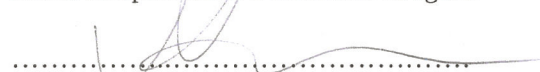
Firma del Titolare dell'Assegno di Ricerca



Firma del Supervisore dell'Attività di Ricerca



Firma Responsabile Scientifico Progetto



Firma del Tutor Ente ospitante (per il periodo all'estero e solo per la relazione semestrale)

.....

investiamo nel vostro futuro

PROGRAMMA OPERATIVO NAZIONALE RICERCA E COMPETITIVITÀ - 2007/2013

ASSE I: "SOSTEGNO AI MUTAMENTI STRUTTURALI"

OBIETTIVO OPERATIVO: "RETI PER IL RAFFORZAMENTO DEL POTENZIALE SCIENTIFICO-TECNOLOGICO DELLE REGIONI DELLA CONVERGENZA"

I AZIONE: "DISTRETTI DI ALTA TECNOLOGIA E RELATIVE RETI"

II AZIONE: "LABORATORI PUBBLICO-PRIVATI E RELATIVE RETI"

PROGETTO PON03PE_00001_1

SETTORE : ICT/TECNOLOGIE PER SMART COMMUNITIES

CUP : H24G14000370005

Responsabile Scientifico Progetto PON03PE_00001_1 *Leone Nicola*
Responsabile Amministrativo Progetto PON03PE_00001_1 *Massaro Pasquale*

Assegno di Ricerca **TECNICHE E ALGORITMI PER LA SERVICE INNOVATION**
Dipartimento di MATEMATICA E INFORMATICA dell'Università della Calabria
Anno Accademico 2015/2016
Periodo di svolgimento 01/05/2015 - 01/11/2015

RELAZIONE SEMESTRALE

Titolare Assegno di Ricerca PON03PE_00001_1 **VELTRI Pierfrancesco**
Supervisore/Referente Scientifico dell'Attività di Ricerca **LEONE Nicola**
Direttore Dipartimento prof. **LEONE Nicola**

Ente ospitante estero _____

Paese estero _____

Tutor/Referente estero _____ (Cognome e Nome) _____ (per l'eventuale periodo all'estero)

La presente relazione è composta
di n. 6 pagine numerate
da 1 a 6

investiamo nel vostro futuro

Logistica, attrezzature

Durante questo semestre ho svolto il mio lavoro di ricerca nello studio presso il Cubo 31B, Secondo Piano, Dipartimento di Matematica, Università della Calabria.

Attività svolte e risultati ottenuti

Al giorno d'oggi, l'utilizzo delle ontologie è largamente diffuso in moltissime applicazioni reali, specialmente per quanto riguarda il *knowledge and data management*. In questo contesto, il problema dell'*ontology-based Query Answering* (QA) sta assumendo un ruolo via via sempre più rilevante [4, 5, 6, 7]. Tale rilevanza è testimoniata anche dallo spiccato interesse mostrato, di recente, da varie case produttrici di sistemi per la gestione di basi di dati. In particolare, sistemi commerciali come, per esempio, *Oracle*, *Ontotext* ed *Ontoprise*, oggi offrono all'utente la possibilità di fare ontology-based QA. Anche nel mondo accademico, della ricerca scientifica, sono numerosi i sistemi che permettono di fare "ragionamento ontologico". Tra questi, possiamo citare, per esempio, *QuOnto* [1], *FaCT++* [2], e *Nyaya* [3].

In generale, il termine "ontologia" si usa per indicare la conoscenza generale di un dominio, la quale, a volte, viene anche chiamata conoscenza terminologica (TBox), in modo che possa essere chiaramente distinta dalla conoscenza asserzionale, che qui indicheremo anche come "dataset". Così, data una *base di conoscenza* $KB = (\Sigma, D)$ composta da un'ontologia Σ ed un dataset D , ed una query q , il problema dell'ontology-based QA consiste nel calcolo dell'insieme delle risposte alla query q che possono essere ottenute tenendo conto della conoscenza implicita modellata dall'ontologia.

Un problema chiave in questo contesto è rappresentato dalla scelta del linguaggio da utilizzare per la specifica della teoria ontologica Σ . Tale linguaggio dovrebbe in qualche modo bilanciare espressività e complessità. In particolare, dovrebbe essere: (1) intuitivo, (2) QA-decidibile (cioè, il QA dovrebbe essere decidibile in questo linguaggio), (3) computabile in maniera efficiente, (4) abbastanza potente in termini di espressività e (5) adatto ad una implementazione efficiente.

investiamo nel vostro futuro

Il linguaggio *Datalog* [8] permette di esprimere diversi assiomi ontologici. Purtroppo, però, l'espressività di *Datalog* non è tale da consentire la “creazione” di nuovi valori. Questa caratteristica è stata riconosciuta come cruciale, specialmente in una prospettiva di mondo aperto, dove non si può assumere che tutti gli individui siano conosciuti a priori. Questo ha motivato il rinnovato interesse per *Datalog*³, che è la naturale estensione di *Datalog* in cui si possono rappresentare variabili quantificate esistenzialmente in testa alle regole. Questo linguaggio è altamente espressivo e consente di modellare domini di conoscenza in maniera semplice e potente. Purtroppo, però, la presenza delle variabili esistenziali rende indecidibile, nel caso generale, il ragionamento su *Datalog*³.

In questo contesto, è sempre maggiore l'interesse rivolto alla famiglia di linguaggi *Datalog*[±], proposta da Cali, Gottlob e Lukasiewicz [5, 7]. In particolare, *Datalog*[±] ha come obiettivo quello di inglobare tutte le estensioni espressive di *Datalog* che sono basate sui ben noti *tuple-generating dependencies* (o TGD, che sono regole *Datalog*³ con eventuali congiunzioni di atomi in testa), *equality-generating dependencies* e *negative constraints*. Il simbolo “+” si riferisce a tutte le possibili combinazioni di queste estensioni, mentre il simbolo “-” ricorda che c'è bisogno di imporre alcune restrizioni per poter garantire quantomeno la “decidibilità” del QA, visto che *Datalog*³ è di per sé indecidibile.

In letteratura, sono già stati proposti diversi linguaggi *Datalog*[±] che garantiscono la decidibilità del QA. Tuttavia, anche se molti di essi preservano la semplicità espressiva di *Datalog* e sono arricchiti di alcune proprietà effettivamente utili ad un linguaggio per la specifica di ontologie, nessuno soddisfa contemporaneamente le condizioni (1)-(5) summenzionate.

Durante il primo semestre di attività, quindi, ci siamo concentrati su questo problema ed abbiamo individuato una nuova classe di programmi *Datalog*[±]: *Shy*. Questa classe di programmi soddisfa una nuova proprietà semantica, denominata *parsimonia*, e risulta essere un linguaggio QA-decidabile molto potente per la specifica di ontologie. *Shy* combina gli aspetti positivi di tutti i linguaggi esistenti della famiglia *Datalog*[±]. Rispetto alle proprietà (1)-(5) introdotte in precedenza, *Shy* si comporta nel seguente modo: (1) eredita pienamente la semplicità e la naturalezza di *Datalog*, (2) garantisce la decidibilità del QA, (3) è computabile in maniera efficiente (in particolare, la data complexity è trattabile), (4) offre un'ottima espressività in quanto estende contemporaneamente sia

investiamo nel vostro futuro

Datalog che Linear Datalog[±] [5] ed, infine, (5) è particolarmente adatto ad una implementazione efficiente. In particolare, i programmi Shy possono essere valutati da una procedura forward-chaining “parsimoniosa”, la quale verrà studiata ed implementata nel prossimo semestre di ricerca.

Da un punto di vista tecnico, quello che è stato fatto finora può essere riassunto, nell'ordine, come segue:

- abbiamo proposto una nuova proprietà semantica, chiamata *parsimonia*, ed abbiamo dimostrato che il QA sulla classe dei programmi Datalog[±] che soddisfano questa proprietà è decidibile ed, inoltre, computabile in maniera efficiente; tale classe di programmi è stata chiamata *Parsimonious-Sets*;
- dopo aver dimostrato che verificare se un programma Datalog[±] soddisfa la proprietà di parsimonia è indecidibile, abbiamo individuato la classe *Shy*, una sottoclasse di *Parsimonious-Sets* che è facilmente riconoscibile e che dà la possibilità di fare QA in maniera efficiente (anche su query congiuntive);
- abbiamo dimostrato che entrambe le classi, *Parsimonious-Sets* e *Shy*, preservano la stessa complessità (data complexity e combined complexity) di Datalog per il query answering su query atomiche: ciò significa che, in questo caso, l'aggiunta dei quantificatori esistenziali non apporta alcun overhead computazionale;
- abbiamo analizzato i related work ed abbiamo delineato una tassonomia precisa di tutte le classi più importanti della famiglia Datalog[±]. E' venuto fuori che, sia *Parsimonious-Sets* che *Shy*, estendono strettamente e contemporaneamente sia Datalog che Linear Datalog[±] [5] (insieme a tutte le loro sottoclassi ovviamente), mentre sono incomparabili con tutti gli altri linguaggi del framework considerato.

I risultati ottenuti in questo semestre, pur se ancora non in veste del tutto definitiva, sono già raccolti in una prima bozza di articolo per la rivista scientifica internazionale TOCL (Transactions on Computational Logic).

Nel prossimo periodo, lo scheduling delle nostre attività di ricerca prevede l'implementazione di una procedura forward-chaining “parsimoniosa” per la valutazione di programmi Shy all'interno del ben noto sistema DLV [9].

investiamo nel vostro futuro

Bibliografia

- [1] Andrea Acciarri, Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Mattia Palmieri, and Riccardo Rosati. QUONTO: querying ontologies. In Proc. of the 20th national conference on Artificial intelligence, volume 4, pages 1670–1671. AAAI Press, 2005.
- [2] Dmitry Tsarkov and Ian Horrocks. FaCT++ Description Logic Reasoner: System Description. In Proc. of the 3rd Int. Joint Conf. on Automated Reasoning, volume 4130 of IJCAR '06, pages 292–297, Seattle, WA, USA., 2006.
- [3] Roberto De Virgilio, Giorgio Orsi, Letizia Tanca, and Riccardo Torlone. Semantic Data Markets: A Flexible Environment for Knowledge Management. In Proc. of the 20th ACM international Conference on Information and Knowledge Management, CIKM '11, New York, NY, USA, 2011. ACM.
- [4] Diego Calvanese, Giuseppe Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family. *J. Autom. Reason.*, 39:385–429, October 2007.
- [5] Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. In Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, PODS '09, pages 77–86, New York, NY, USA, 2009. ACM.
- [6] Ilianna Kollia, Birte Glimm, and Ian Horrocks. Sparql query answering over owl ontologies. In Proceedings of the 24th International Workshop on Description Logics, volume 6643 of Lecture Notes in Computer Science, pages 382–396. Springer Berlin/Heidelberg, 2011.
- [7] Andrea Calì, Georg Gottlob, and Andreas Pieris. New Expressive Languages for Ontological Query Answering. In Proceedings of the 25th AAAI Conference on Artificial Intelligence, pages 1541–1546, 2011.
- [8] Serge Abiteboul, Richard Hull, and Victor Vianu. Foundations of Databases: The Logical Level. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.
- [9] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello. The DLV System for Knowledge Representation and Reasoning. *ACM TOCL* 7(3):499–562.

investiamo nel vostro futuro

Data e timbro del Dipartimento,

31/03/2016




Firma del Titolare dell'Assegno di Ricerca



Firma del Supervisore dell'Attività di Ricerca



Firma Responsabile Scientifico Progetto



Firma del Tutor Ente ospitante (per il periodo all'estero e solo per la relazione semestrale)

.....



UNIVERSITÀ DELLA CALABRIA
DIPARTIMENTO DI
MATEMATICA
E INFORMATICA

Al Direttore del Dipartimento di
Matematica e Informatica

Oggetto: Richiesta Rinnovo per Assegno di Ricerca.
Dott. Dodaro Carmine - SSD INF/01

Il sottoscritto, prof. Francesco Ricca,

PREMESSO

che il Dott. Carmine Dodaro è stato titolare di assegno di ricerca per 12 mesi (D.R. 2648 del 22.12.2014) e che l'attività di ricerca ha ottenuto ottimi risultati, il cui sviluppo ha evidenziato importanti applicazioni future,

RICHIEDE

il rinnovo dell'assegno di ricerca per ulteriori 12 mesi per l'approfondimento di ricerche correlate al potenziamento del sistema di intelligenza artificiale DLV attraverso l'integrazione ed il miglioramento del modulo WASP. La spesa graverà sui fondi di ricerca intestati al professore Nicola Leone.

IL RESPONSABILE SCIENTIFICO
(prof. Francesco RICCA)

IL TITOLARE DEI FONDI DI
RICERCA
(prof. Nicola LEONE)

Rende, 31/03/2016



UNIVERSITÀ DELLA CALABRIA
DIPARTIMENTO DI
MATEMATICA
E INFORMATICA

ATTESTAZIONE DEL TUTOR

Il sottoscritto Prof. Francesco Ricca, tutor scientifico del Dott. Carmine Dodaro per l'Assegno per la collaborazione ad attività di ricerca vinto con il progetto di ricerca intitolato "Aggregati Ricorsivi in Answer Set Programming: Algoritmi e Implementazione" (Bando di Concorso emanato con D.R. n.2648 del 22/12/2014)

ATTESTA

la regolarità, qualità e coerenza delle attività svolte dal Dott. Carmine Dodaro con il progetto di ricerca finanziato e gli eccellenti risultati ottenuti.

Rende, lì 31/03/2016

IL RESPONSABILE SCIENTIFICO

Prof. Francesco Ricca

RELAZIONE FINALE DEL PROGETTO DI RICERCA

AGGREGATI RICORSIVI IN ANSWER SET PROGRAMMING: ALGORITMI E
IMPLEMENTAZIONE

(Bando di Concorso emanato con D.R. n.2648 del 22/12/2014)

DOTT. CARMINE DODARO
dodaro@mat.unical.it

1 Logistica, attrezzature

Durante questo semestre ho svolto il mio lavoro di ricerca nello studio presso il Cubo 31B, Secondo Piano, Dipartimento di Matematica, Università della Calabria. Ho inoltre usufruito delle seguenti attrezzature per testare le performance dei prodotti di ricerca sviluppati: 1. Le-gione, una macchina con 2 processori Intel Xeon "Woodcrest" (quad core) 3GHz con 4MB di L2 Cache e 4GB di RAM, con sistema operativo Debian GNU Linux 4.0. 2. Efeso, un insieme di macchine a 4 core Intel Xeon CPU X3430 2.4 Ghz e 4GB di RAM, con sistema operativo Linux Debian Lenny (32bit).

2 Attività svolte e risultati ottenuti

L'Answer Set Programming (ASP, in breve) è un paradigma di programmazione dichiarativa basato sulla semantica dei modelli stabili. L'idea alla base di ASP è di codificare un problema computazionale in un programma logico i cui modelli stabili, anche detti answer set, corrispondono alle soluzioni del problema. L'espressività e l'intuitività della semantica di ASP hanno contribuito alla sua diffusione in diversi contesti accademici, dall'intelligenza artificiale fino ad applicazioni di bioinformatica. Inoltre, negli ultimi anni, ASP è stato largamente utilizzato anche per applicazioni industriali. Molti dei problemi industriali presentati finora possono essere modellati in un modo semplice e naturale usando aggregati ricorsivi. Durante questo ultimo anno ci siamo occupati della realizzazione di nuovi algoritmi per la valutazione di programmi ASP in presenza di aggregati ricorsivi. In un articolo recente dal titolo "Rewriting recursive aggregates in answer set programming: back to monotonicity" pubblicato sulla rivista scientifica "Theory and Practice of Logic Programming", Alviano, Faber, e Gebser hanno presentato una riscrittura di aggregati ricorsivi non monotoni in aggregati ricorsivi monotoni utilizzando programmi logici disgiuntivi. Questa efficiente riscrittura è stata implementata nel grounder gringo e quindi tutti i solver esistenti possono beneficiare delle sue potenzialità. Tuttavia, la riscrittura ha evidenziato una debolezza dei sistemi ASP esistenti nella valutazione dei programmi disgiuntivi. In particolare, i solver ASP efficienti come CLASP e WASP utilizzano una tecnica di riscrittura chiamata *Clark's completion*. Tuttavia, la *Clark's completion* è definita su programmi non disgiuntivi, quindi è di solito preceduta da un'altra tecnica chiamata *shift*, che si occupa di trasformare un programma disgiuntivo in un programma non disgiuntivo. Lo svantaggio principale di questa tecnica di *shift* è l'esplosione quadratica che causa nel programma in input. In particolare, per ogni regola disgiuntiva contenente n atomi la tecnica di *shift* produce n regole non disgiuntive. Un'accurata valutazione empirica ha dimostrato come questa tecnica sia inefficiente in pratica. Consideriamo come esempio un programma contenente una sola regola disgiuntiva. La figura 1 mostra il consumo della memoria dei solver CLASP e WASP al variare del numero di atomi in disgiunzione. Come si può osservare, una disgiunzione contenente 30000 atomi è già fuori dalla portata dei due solver allo stato dell'arte CLASP e WASP.

Risultati ottenuti. Una regola disgiuntiva r è del tipo:

$$p_1 \vee \dots \vee p_m \leftarrow \ell_1 \wedge \dots \wedge \ell_n \quad (1)$$

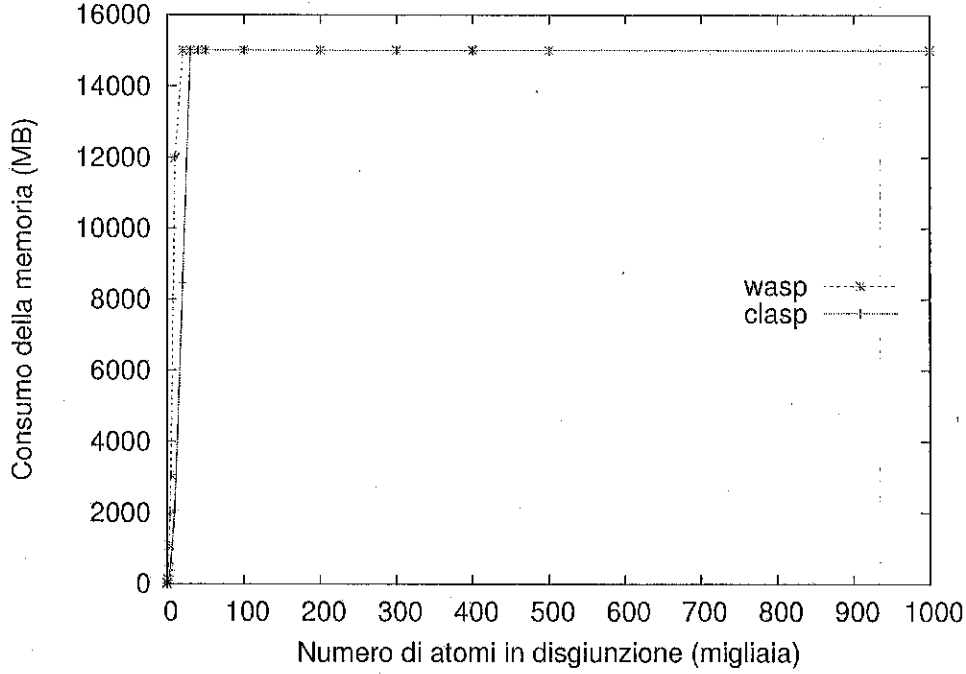


Figura 1: Consumo della memoria al variare degli atomi in disgiunzione

dove $m \geq 1$, $n \geq 0$, p_1, \dots, p_m sono atomi distinti, e ℓ_1, \dots, ℓ_n sono letterali distinti. I modelli supportati di un programma senza disgiunzione possono essere computati attraverso la tecnica chiamata *Clark's completion*. Sia Π un programma senza disgiunzione. La completion di Π , denotata $comp(\Pi)$, è l'insieme delle seguenti clausole:

$$p_1^r \leftrightarrow \ell_1 \wedge \dots \wedge \ell_n \quad (2)$$

per tutte le regole di $r \in \Pi$ della forma (1) con $m = 1$, dove p_1^r è un atomo nuovo (p_1^r è vero se e solo se r è un supporto di p_1), insieme con

$$p \leftrightarrow \bigvee_{r \in heads(\Pi, p)} p^r \quad (3)$$

per tutti gli atomi p del programma.

Esempio 1. Sia Π_1 il seguente programma:

$$\begin{array}{ll} a \leftarrow \neg b \wedge \neg c & b \leftarrow a \\ b \leftarrow \neg a \wedge \neg c & c \leftarrow \neg a \\ c \leftarrow \neg a \wedge \neg b & \end{array}$$

La completion di Π_1 è:

$$\begin{array}{lll} a^1 \leftrightarrow \neg b \wedge \neg c & b^2 \leftrightarrow a & a \leftrightarrow a^1 \\ b^1 \leftrightarrow \neg a \wedge \neg c & c^3 \leftrightarrow \neg a & b \leftrightarrow b^1 \vee b^2 \\ c^1 \leftrightarrow \neg a \wedge \neg b & & c \leftrightarrow c^1 \vee c^3 \end{array}$$

Come detto, per applicare la Clark's completion a programmi disgiuntivi una trasformazione nota come *shift* è applicata al programma in input Π , in modo da ottenere un programma $shift(\Pi)$ con gli stessi modelli supportati. Formalmente, $shift(\Pi)$ è il programma che contiene le seguenti regole:

$$p_i \leftarrow l_1 \wedge \dots \wedge l_n \wedge \bigwedge_{j \in [1..n], j \neq i} \neg p_j \quad \forall i \in [1..m] \quad (4)$$

per tutte le regole $r \in \Pi$ della forma (1). Il problema principale dello *shift* è che la sua costruzione non è lineare, bensì quadratica nella dimensione del programma. Per questo motivo abbiamo presentato la completion di programmi disgiuntivi. La completion di un programma disgiuntivo Π , denotata $comp^\vee(\Pi)$, è l'insieme delle seguenti clausole:

$$d_i^r \leftrightarrow p_i \vee d_{i+1}^r \quad \forall i \in [2..m-1] \quad (5)$$

$$d_m^r \leftrightarrow p_m \quad \text{if } m \geq 2 \quad (6)$$

$$s_1^r \leftrightarrow l_1 \wedge \dots \wedge l_n \quad (7)$$

$$s_i^r \leftrightarrow s_{i-1}^r \wedge \neg p_{i-1} \quad \forall i \in [2..m] \quad (8)$$

$$p_i^r \leftrightarrow s_i^r \wedge \neg d_{i+1}^r \quad \forall i \in [1..m-1] \quad (9)$$

$$p_m^r \leftrightarrow s_m^r \quad (10)$$

per tutte le regole $r \in \Pi$ della forma (1), insieme con (3) per tutti gli atomi p del programma. Si noti che la costruzione in questo caso è lineare nella dimensione del programma in input.

Esempio 2. Si consideri il seguente programma Π_2 :

$$a \vee b \vee c \leftarrow \quad b \leftarrow a \quad c \leftarrow \neg a$$

La completion del programma Π_2 è la seguente $comp^\vee(\Pi_2)$:

$$\begin{array}{lll} d_2^1 \leftrightarrow b \vee d_3^1 & d_3^1 \leftrightarrow c & \\ \{s_1^1\} & s_2^1 \leftrightarrow s_1^1 \wedge \neg a & s_3^1 \leftrightarrow s_2^1 \wedge \neg b \\ a^1 \leftrightarrow s_1^1 \wedge \neg d_2^1 & b^2 \leftrightarrow a & a \leftrightarrow a^1 \\ b^1 \leftrightarrow s_2^1 \wedge \neg d_3^1 & c^3 \leftrightarrow \neg a & b \leftrightarrow b^1 \vee b^2 \\ c^1 \leftrightarrow s_3^1 & & c \leftrightarrow c^1 \vee c^3 \end{array}$$

Come contributo aggiuntivo abbiamo implementato la completion disgiuntiva nell'ASP solver WASP, insieme con una gestione efficiente delle regole (5)–(10) in memoria attraverso l'uso di un propagatore.

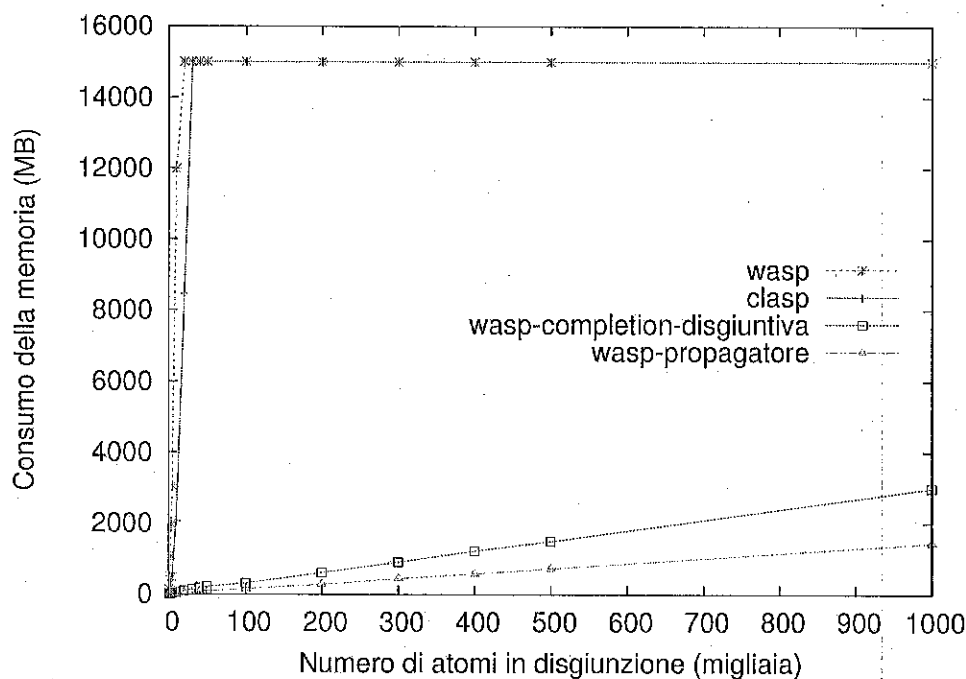


Figura 2: Consumo della memoria al variare degli atomi in disgiunzione.

Discussione dei risultati. Nel corso del 2015 Alviano et al. hanno presentato una riscrittura efficiente di aggregati ricorsivi non monotoni in aggregati ricorsivi monotoni utilizzando programmi logici disgiuntivi. Questa efficiente riscrittura ha evidenziato una debolezza dei sistemi ASP esistenti nella valutazione dei programmi disgiuntivi. In particolare, lo shift di programmi disgiuntivi causa un'esplosione quadratica nel programma in input. Quindi, durante questo anno abbiamo proposto una nuova riscrittura con le stesse proprietà dello shift ma che è lineare nella dimensione del programma in input. Inoltre, abbiamo proposto l'uso di un propagatore per mantenere in memoria le strutture dati richieste dalla riscrittura. Abbiamo implementato le due nuove tecniche nel solver WASP e valutato il loro impatto nella risoluzione pratica dei problemi. In figura 1 abbiamo mostrato le debolezze dei sistemi esistenti nella valutazione di una sola regola disgiuntiva molto lunga. La figura 2 mostra l'efficacia delle nuove tecniche per risolvere questo problema. Dal grafico si può notare come la nuova versione di WASP sia in grado di processare regole contenenti un milione di atomi in disgiunzione, sia quando viene applicata la riscrittura e sia quando viene usato il propagatore. Inoltre, allo scopo di valutare le performance delle nuove tecniche su problemi computazionali complessi, abbiamo eseguito un'analisi sperimentale su istanze del Minimal Hitting Set, un noto problema computazionale che è facilmente modellabile utilizzando Answer Set Programming. La figura 3 è un cactus plot che riporta i risultati della valutazione sperimentale. Intuitivamente, un punto (i,j) del grafico rappresenta che l'istanza i -esima è stata risolta in j secondi. Quindi il solver che si trova più a destra è quello che risolve più istanze nel minore tempo. Si può osservare come WASP con l'utilizzo del propagatore sia molto più efficiente di WASP con la tecnica dello shift e di CLASP.

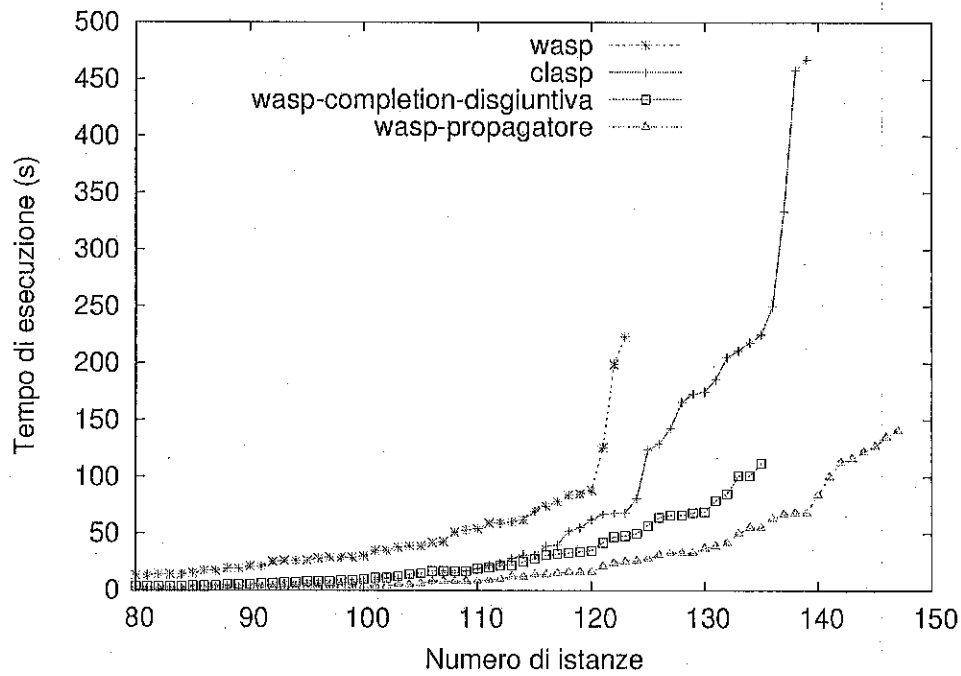


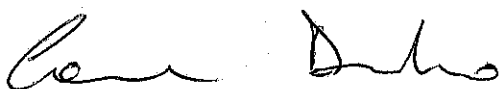
Figura 3: Risultati sulle istanze del minimal hitting set

Pubblicazioni. Abbiamo inviato un articolo contenente la descrizione delle nuove tecniche e dei risultati ottenuti per la pubblicazione negli atti di convegno della conferenza IJCAI 2016.

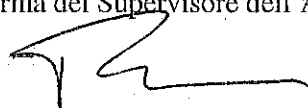
Data

31/03/2016

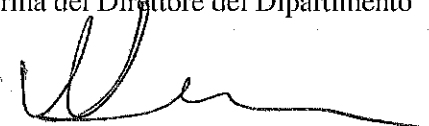
Firma del Titolare dell'Assegno di Ricerca

A handwritten signature in black ink, appearing to be 'C. Deho'.

Firma del Supervisore dell'Attività di Ricerca

A handwritten signature in black ink, appearing to be 'R.'.

Firma del Direttore del Dipartimento

A handwritten signature in black ink, appearing to be 'U.'.



UNIVERSITÀ DELLA CALABRIA
DIPARTIMENTO DI
MATEMATICA
E INFORMATICA

Al Direttore del Dipartimento di
Matematica e Informatica

Oggetto: Richiesta Proroga per Assegno di Ricerca.
Dott.sa Fionda Valeria - SSD INF/01

Il sottoscritto, prof. Gianluigi Greco,

PREMESSO

che la Dott.ssa Valeria Fionda è stata titolare di assegno di ricerca per 12 mesi (D.R. 2648 del 22.12.2014) e che l'attività di ricerca ha ottenuto ottimi risultati, il cui sviluppo ha evidenziato importanti applicazioni future,

RICHIEDE

La proroga dell'assegno di ricerca per ulteriori 4 mesi per l'approfondimento di ricerche correlate all'applicazione della Logica Temporale Lineare su tracce finite al process mining. La spesa graverà sui fondi di ricerca intestati al professore Gianluigi Greco.

Rende, lì 01/04/2016

IL RESPONSABILE SCIENTIFICO
(prof. Gianluigi GRECO)

IL TITOLARE DEI FONDI DI RICERCA
(prof. Gianluigi GRECO)



UNIVERSITÀ DELLA CALABRIA

DIPARTIMENTO DI
MATEMATICA
E INFORMATICA

ATTESTAZIONE DEL TUTOR

Il sottoscritto Prof. Gianluigi Greco, tutor scientifico della Dott.ssa Valeria Fionda per l'Assegno per la collaborazione ad attività di ricerca vinto con il progetto di ricerca intitolato "La Logica Temporale Lineare su tracce finite: identificare la frontiera di trattabilità della soddisfacibilità di formule rispetto a restrizioni sintattiche e applicazioni al process mining" (Bando di Concorso emanato con D.R. n.2648 del 22/12/2014).

ATTESTA

La regolarità, qualità e coerenza delle attività svolte dalla Dott.ssa Valeria Fionda con il progetto di ricerca finanziato e gli eccellenti risultati ottenuti.

Rende, lì 01/09/2016

IL RESPONSABILE SCIENTIFICO

Prof. Gianluigi Greco

RELAZIONE FINALE DEL PROGETTO DI RICERCA

LA LOGICA TEMPORALE LINEARE SU TRACCE FINITE: IDENTIFICARE LA
FRONTIERA DI TRATTABILITÀ DELLA SODDIFACIBILITÀ DI FORMULE
RISPETTO A RESTRIZIONI SINTATTICHE E APPLICAZIONI AL PROCESS
MINING

(Bando di Concorso emanato con D.R. n.2648 del 22/12/2014)

DOTT.SSA VALERIA FIONDA

INDICE

| | |
|--|----------|
| Obiettivi | 3 |
| Attività svolte..... | 3 |
| Ricerca Bibliografica..... | 3 |
| Descrizione delle Attività..... | 4 |
| Risultati Ottenuti | 6 |
| Pubblicazioni | 8 |
| Conclusioni | 8 |

Obiettivi

L'obiettivo del progetto era quello di creare un quadro chiaro della complessità del ragionamento su formule LTLf e di sviluppare un reasoner per il controllo della soddisfacibilità. Lo scopo era di fornire i seguenti contributi:

- Identificare le classi di formule per le quali esiste sempre almeno un modello la cui lunghezza è lineare nella dimensione della formula.
- Studiare per quali classi di formule il problema di verificare se una data traccia è un modello può essere risolto in tempo polinomiale.
- Effettuare uno studio sistematico della complessità di verificare la soddisfacibilità di ogni classe di formule, verificando se tale problema è PSPACE-completo, NP-completo o trattabile.
- Progettare e implementare un reasoner per LTLf.

Attività svolte

Di seguito sono riportati l'elenco del materiale bibliografico studiato e una descrizione delle attività svolte e dei risultati ottenuti.

Ricerca Bibliografica

Di seguito l'elenco del materiale bibliografico studiato:

1. Artale, R. Kontchakov, V. Ryzhikov, and M. Zakharyashev. The complexity of clausal fragments of ltl. In LPAR , pages 35–52, 2013.
2. G. Audemard and L. Simon. Predicting learnt clauses quality in modern SAT solvers. In IJCAI , pages 399–404, 2009.
3. M. Bauland, T. Schneider, H. Schnoor, I. Schnoor, and H. Vollmer. The complexity of generalized satisfiability for linear temporal logic. Log Meth Comput Sci , 5(1), 2009.
4. J.A. Baier and S. A. McIlraith. Planning with First-Order Temporally Extended Goals using Heuristic Search. In AAAI , 2006.
5. A. Biere, K. Heljanko, T.A. Junttila, T. Latvala, and V. Schuppan. Linear Encodings of Bounded LTL Model Checking. Log Meth Comput Sci , 2(5), 2006.
6. C.-C. Chen and I-P. Lin. The Computational Complexity of Satisfiability of Temporal Horn Formulas in Propositional Linear-Time Temporal Logic. Inf. Process. Lett. , 45(3):131–136, 1993.
7. G. De Giacomo and M. Y. Vardi. Linear Temporal Logic and Linear Dynamic Logic on Finite Traces. In IJCAI , 2013.
8. G. De Giacomo and M. Y. Vardi. Synthesis for LTL and LDL on finite traces. In IJCAI , pages 1558–1564, 2015.
9. G. De Giacomo, R. De Masellis, M. Grasso, F. M. Maggi, and M. Montali. Monitoring Business Metaconstraints Based on LTL and LDL for Finite Traces. In BPM , 2014.
10. G. De Giacomo, R. De Masellis, and M. Montali. Reasoning on LTL on Finite Traces: Insensitivity to Infiniteness. In AAAI , 2014.
11. S. Demri and Ph. Schnoebelen. The Complexity of Propositional Linear Temporal Logics in Simple Cases. Inf. Comput. , 174(1):84–103, 2002.
12. C. Dixon, M. Fisher, and B. Konev. Tractable temporal reasoning. In IJCAI , pages 318–323, 2007.
13. S. Edelkamp. On the Compilation of Plan Constraints and Preferences. In ICAPS , pages 374–377. AAAI, 2006.
14. M. Garey and D. Johnson. Computers and Intractability – A guide to the Theory of NP-

- Completeness . Freeman, 1979.
15. A. Gerevini, P. Haslum, D. Long, A. Saetti, and Y. Dimopoulos. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artif. Intell.* , 173(5-6):619–668, 2009.
 16. G. Gottlob and G. Greco. Decomposing combinatorial auctions and set packing problems. *J. ACM* , 60(4):24, 2013.
 17. G. Greco, A. Guzzo, F. Lupia and L. Pontieri. Process discovery under precedence constraints. *TKDD* , 9(4):32, 2015.
 18. E. Hemaspaandra. The complexity of poor man's logic. *J. Log. Comput.* , 11(4):609622, 2001.
 19. Li, L. Zhang, G. Pu, M. Y. Vardi, and J. He. LTLf satisfiability checking. In *ECAI* , pages 513–518, 2014.
 20. N. Markey. Past is for free: on the complexity of verifying linear temporal properties with past. *Acta Inf.* , 40(6-7):431–458, 2004.
 21. H. Ono and A. Nakamura. On the size of refutation Kripke models for some linear modal and tense logics. *Studia Logica* , 39(4):325 – 333, 1980.
 22. M. Pesic and W.M.P. van der Aalst. DecSerFlow: Towards a Truly Declarative Service Flow Language. In *The Role of Business Processes in Service Oriented Architectures* , number 6291 in *Dagstuhl Seminar Proceedings*, July 2006.
 23. M. Pesic, H. Schonenberg, and W.M.P. van der Aalst. DECLARE: Full Support for Loosely-Structured Processes. In *EDOC* , pages 287–298, 2007.
 24. M. Pesic, D. Bosnacki, and W.M.P. van der Aalst. Enacting Declarative Languages Using LTL: Avoiding Errors and Improving Performance. In *SPIN* , pages 146–161, 2010.
 25. A. Pnueli. The temporal logic of programs. In *FOCS* , pages 46–57. IEEE Computer Society, 1977.
 26. A. Pnueli. The Temporal Semantics of Concurrent Programs. *Theor. Comput. Sci.* , 13:45–60, 1981.
 27. P.-Y. Schobbens and J.-F. Raskin. The Logic of Initially and Next, Complete Axiomatisation and Complexity Issues. *Inform Process Lett* , 69(5):221–225, 1999.
 28. A. P. Sistla and E. M. Clarke. The Complexity of Propositional Linear Temporal Logics. *J. ACM* , 32(3):733–749, 1985.
 29. W. M. P. van der Aalst, M. Pesic, and H. Schonenberg. Declarative Workflows: Balancing Between Flexibility and Support. *Computer Science - Research and Development* , 23(2):99–113, 2009.

Descrizione delle Attività

LTL. LTL è una logica modale le cui modalità sono operatori temporali relativi a eventi che avvengono in istanti temporali disposti su una linea temporale ordinata. LTL è stata introdotta negli anni 70 come strumento formale per verificare la correttezza dei programmi e dei sistemi reattivi. Da allora ha trovato applicazione in diverse aree dell'intelligenza artificiale e dell'informatica. Le formule LTL sono interpretate su tracce infinite, cioè sequenze di stati che descrivono gli eventi, modellati come variabili proposizionali, che avvengono nei diversi istanti temporali. Data una formula LTL il problema più rilevante è quello di stabilire se esiste una traccia infinita che la soddisfa. Tuttavia, in alcune applicazioni, come la specifica e verifica dei processi di business, è stato osservato che una scelta più naturale è quella di interpretare LTL su tracce finite (LTLf). Verificare la soddisfacibilità di formule LTL è in generale un problema PSPACE-completo e molti sforzi sono stati fatti per identificare classi di complessità più favorevoli concentrandosi su frammenti di LTL. Questi risultati di trattabilità non si applicano alle formule LTLf per le quali la soddisfacibilità resta PSPACE-completa e non erano stati fatti studi sistematici sulla complessità dei frammenti definiti rispetto all'insieme di operatori temporali usati. Quindi un primo contributo è stato quello di colmare questa mancanza con uno studio

sistematico della complessità del problema di verificare la soddisfacibilità di formule LTLf. In particolare, per identificare le isole di trattabilità, circumnavigando la NP-hardness dalla logica proposizionale, sono state considerate classi di formule ottenute definendo restrizioni sintattiche sia rispetto agli operatori temporali che ai connettivi logici. Formalmente, per ogni insieme $B \subseteq \{\text{AND}, \text{OR}, \text{NOT}\}$ e per ogni insieme $T \subseteq \{X, G, F\}$, abbiamo definito $\langle B, T \rangle$ -LTLf come la classe di tutte le formule LTLf in cui solo connettivi in B e operatori temporali in T possono essere usati. Per ogni $B \subseteq \{\text{AND}, \text{OR}, \text{NOT}\}$ e $T \subseteq \{X, G, F\}$, la classe $\langle B, T \rangle$ -LTLfs è stata definita analogamente alla classe $\langle B, T \rangle$ -LTLf la cui sintassi coincide con la sintassi di $\langle B, T \rangle$ -LTLf, e la cui semantica differisce solo nel fatto che la soddisfacibilità è definita rispetto ai modelli 1-limitati, cioè in cui in ogni istante di tempo una e una sola variabile proposizionale è vera. LTLf è stata una scelta naturale nel contesto del ragionamento su problemi legati alla specifica e verifica dei processi aziendali e, in particolare, lo studio di LTLfs è stato motivato dal fatto che in tali applicazioni il numero di attività (trasparentemente modellate come variabili proposizionali) che può essere eseguito in parallelo, cioè, nello stesso istante di tempo, è limitato a uno.

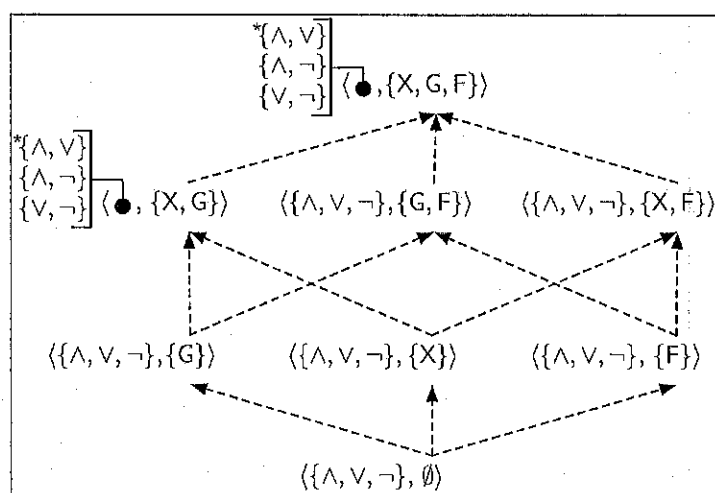
LTL e Process Mining. In un secondo momento, è stata studiata l'applicabilità di LTLf al process mining come strumento per definire vincoli sulla struttura dei processi ottenuti analizzando i log. In particolare, LTLf è una scelta naturale nel contesto del ragionamento su problemi legati alla specifica e verifica dei processi aziendali e, in particolare, lo studio di una particolare restrizione di LTLf la cui semantica differisce nel fatto che la soddisfacibilità è definita solo rispetto ai modelli k -limitati, cioè in ogni istante di tempo al più k attività possono essere eseguite, è motivato dal fatto che in tali applicazioni il numero di attività (trasparentemente modellate come variabili proposizionali) che può essere eseguito in parallelo, cioè, nello stesso istante di tempo, è molto spesso una piccola costante (rispetto alla lunghezza delle tracce). Per esempio assumendo che al massimo una variabile possa essere vera in qualsiasi istante di tempo sono considerati soltanto modelli 1-limitati. Questa ipotesi restrittiva è piuttosto standard nel contesto del process mining, il cui obiettivo è quello di derivare automaticamente un modello di processo in grado di spiegare tutti gli episodi registrati in un log di eventi legati all'esecuzione delle attività di un processo sottostante. Infatti, gli algoritmi di process mining si basano molto spesso su viste astratte di un log, semplicemente concentrandosi sull'ordine in cui le varie attività sono eseguite (in particolare, completate).

Progettazione, implementazione e valutazione di un ragionatore. Nell'ultima fase del progetto ha riguardato l'implementazione di un ragionatore per formule LTLf basato sui risultati teorici nel progetto. Per le classi sulle quali la soddisfacibilità è risultata essere trattabile, nel ragionatore sono stati implementati gli algoritmi definiti nello studio teorico. Per le altre classi, l'obiettivo è stato quello di sviluppare un algoritmo di riscrittura che traducesse le formule LTLf in formule booleane equivalenti e utilizzasse un SAT solver per calcolare un modello. Riguardo alle classi per cui la soddisfacibilità è risultata essere NP-completa, sfruttando ove possibile la proprietà sulla lunghezza lineare dei modelli, si è potuto garantire che la dimensione della formula booleana risultante sia polinomiale nella lunghezza della formula LTLf. In generale, invece, la codifica potrebbe essere esponenziale rispetto alla lunghezza della formula di ingresso. In questi casi il ragionatore è stato implementato in modo da attuare una strategia che vincoli la dimensione della riscrittura fissando una

lunghezza massima per i modelli, nel qual caso viene garantita la correttezza del ragionatore ma non la sua completezza.

Risultati Ottenuti

La figura seguente mostra le classi di formule per le quali esiste sempre almeno un modello la cui lunghezza è lineare nella dimensione della formula.



Tutte le classi di formule LTLf e LTLfs, per cui vale la proprietà di lunghezza lineare sono summarize in termini del diagramma di Hasse costruito sul sottoinsieme $\{X, G, F\}$. In particolare, per ogni sottoinsieme $T \subseteq \{X, G, F\}$ la figura riporta il sottoinsieme massimale di operatori booleani $B \subseteq \{\text{AND}, \text{OR}, \text{NOT}\}$ per cui la proprietà vale (lo * indica che la proprietà vale per LTLf ma non per LTLfs).

La tabella seguente riporta i risultati di complessità ottenuti per il problema di verificare la soddisfacibilità di formule LTLf. In particolare, sulle righe della tabella sono riportati i sottoinsiemi di operatori temporali e sulle colonne i sottoinsiemi di connettivi booleani.

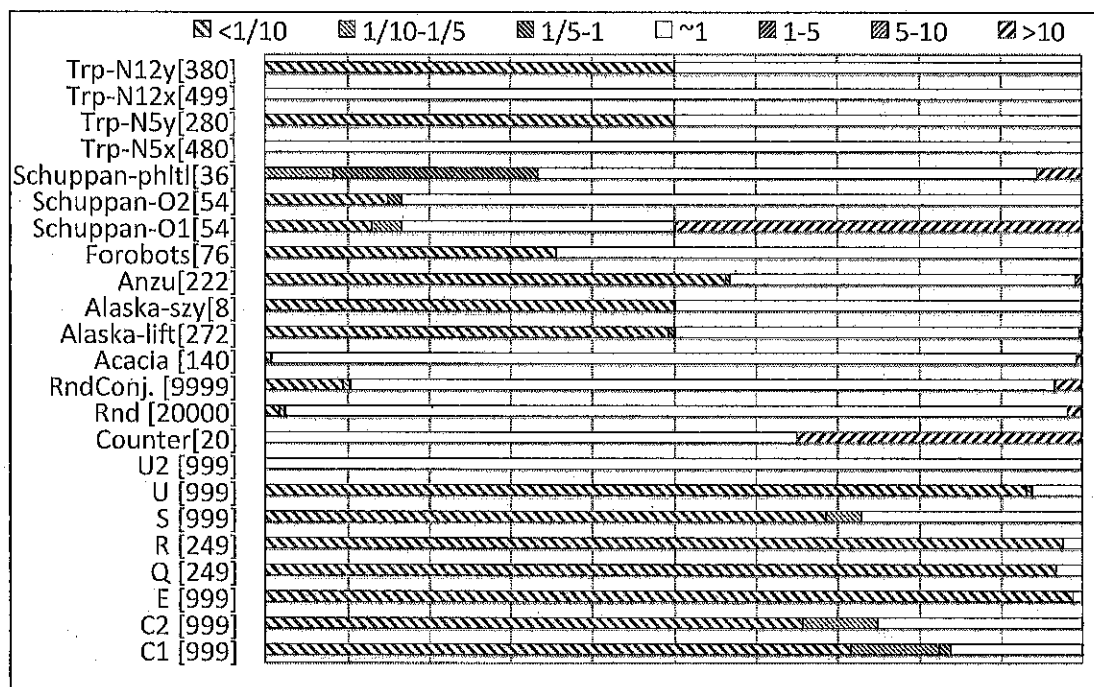
| LTL _f | | | | | | | |
|------------------|----------------------|----------------|----------------|--------------|----------|--------|--------|
| | \wedge, \vee, \neg | \wedge, \vee | \wedge, \neg | \vee, \neg | \wedge | \vee | \neg |
| G | NP-c | in P | in P | in P | in P | in P | in P |
| F | NP-c | in P | in P | in P | in P | in P | in P |
| X | NP-c | in P | in P | in P | in P | in P | in P |
| XF | NP-c | in P | in P | in P | in P | in P | in P |
| GF | NP-c | in P | in P | in P | in P | in P | in P |
| XG | PSPACE-c | in P | in P | in P | in P | in P | in P |
| XGF | PSPACE-c | in P | NP-c | in P | in P | in P | in P |

Nella tabella seguente sono invece summarize i risultati per le formule LTLfs.

| LTL _{f,e} | | | | | | | |
|--------------------|----------------------|----------------|----------------|--------------|----------|--------|--------|
| | \wedge, \vee, \neg | \wedge, \vee | \wedge, \neg | \vee, \neg | \wedge | \vee | \neg |
| G | in P | in P | in P | in P | in P | in P | in P |
| F | in P | in P | in P | in P | in P | in P | in P |
| X | NP-c | NP-c | in P | in P | in P | in P | in P |
| XF | NP-c | NP-c | in P | in P | in P | in P | in P |
| GF | NP-c | NP-c | in P | in P | in P | in P | in P |
| XG | PSPACE-c | PSPACE-c | in P | in P | in P | in P | in P |
| XGF | PSPACE-c | PSPACE-c | NP-c | in P | NP-c | in P | in P |

Tutti i risultati sono di trattabilità (P) o risultati di completezza per le classi NP o PSPACE.

Infine, è stata effettuata una valutazione sperimentale del ragionatore e un confronto con i ragionatori presentati in letteratura sia per LTL che per LTL_f e in molti casi il ragionatore proposto è più efficiente di quelli presenti in letteratura. Gli esperimenti sono stati eseguiti su un PC Intel Core i5 2,4 GHz, 8GB RAM. Per ogni formula è stato calcolato il rapporto tra il tempo impiegato dal ragionatore proposto per decidere sulla soddisfacibilità e il tempo impiegato da Aalta, il solo ragionatore disponibile in letteratura specificatamente proposto per LTL_f. La figura sottostante riporta i risultati come grafici a barre percentuali per ognuno dei dataset utilizzati. Le barre bianche riportano la percentuale di formule in cui i due sistemi ottengono le stesse performance (con una tolleranza di 30 ms) mentre le barre sulla sinistra (rispettivamente sulla destra) sono associate con le formule in cui il rapporto è minore (rispettivamente maggiore) di 1, tale che il nostro ragionatore (rispettivamente Aalta) è più veloce.



Pubblicazioni

- Fionda V., Greco G. *The Complexity of LTL on Finite Traces: Hard and Easy Fragments*. In the proceedings of the "Thirtieth AAAI Conference on Artificial Intelligence (AAAI2016)". Phoenix, Arizona, 12-17 February 2016.
- Consens M.P., Fionda V., Khatchadourian S. and Pirrò G. S+EPP: Construct and Explore Bisimulation Summaries, plus Optimize Navigational Queries; all on Existing SPARQL Systems. Proceedings of the 41st International Conference on Very Large Databases (VLDB). Demo Paper, VLDB Endowment, September 2015.

Sottomessi e in fase di revisione:

- Fionda V., Pirrò G. and Gutierrez C. *Building Knowledge Maps of Web Graphs*. Sottomesso ad Artificial Intelligence – Elsevier
- Fionda V., Pirrò G. and Consens M.P. *Extended Property Paths: Expressive Navigation and Query-Based RDFS Reasoning*. Sottomesso a IEEE Transactions on Knowledge and Data Engineering.


Conclusioni

Le attività svolte durante i 12 mesi di assegno sono terminate con successo e gli obiettivi previsti sono stati raggiunti. I risultati ottenuti sono stati più che soddisfacenti e aprono le porte ad ulteriori investigazioni e miglioramenti.

Firma del Titolare dell'Assegno di ricerca



Firma del Responsabile Scientifico



Firma del Direttore del Dipartimento

