Valutazione efficiente di aggregati ricorsivi in programmazione logica

Mario Alviano

RELATORI: Nicola Leone e Wolfgang Faber

Università della Calabria

27 Luglio 2007

Outline

1. Motivazioni
2. Programmazione logica con aggregati
   - Sintassi
   - Semantica
   - Monotonicità e antimonotonicità
   - Rappresentazione della conoscenza
3. Contributi

Motivazioni

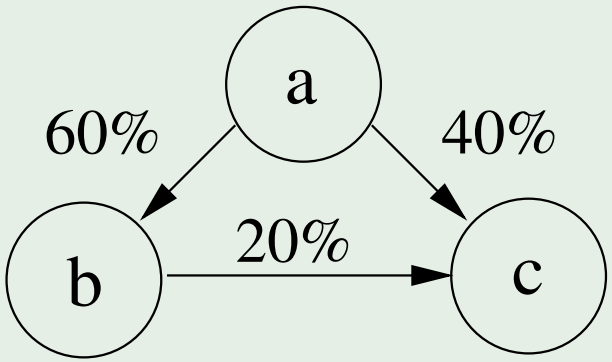
- La Programmazione Logica Disgiuntiva (DLP) consente la modellazione di problemi difficili
- Gli aggregati facilitano la rappresentazione dei problemi e migliorano l'efficienza
- Possiamo usare efficientemente la ricorsione sugli aggregati?
- I precedenti lavori mostrano che questo può essere fatto...
- ... ma non come farlo!

Motivazioni

- La Programmazione Logica Disgiuntiva (DLP) consente la modellazione di problemi difficili
- Gli aggregati facilitano la rappresentazione dei problemi e migliorano l'efficienza
- Possiamo usare efficientemente la ricorsione sugli aggregati?
- I precedenti lavori mostrano che questo può essere fatto...
- ... ma non come farlo!

Esempio: Company Controls

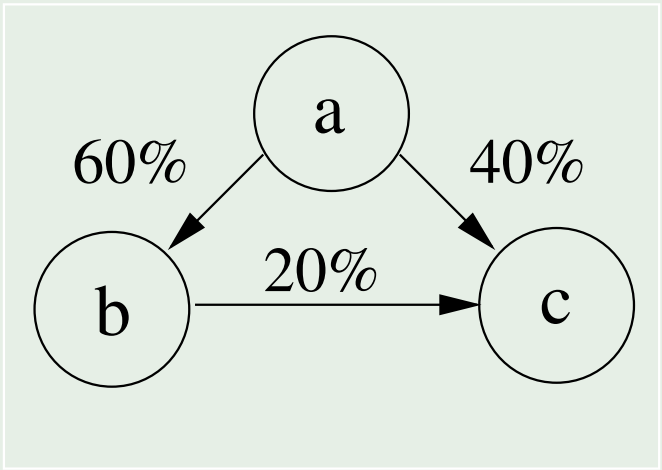
Esempio (Company Controls)



- Compagnie
- Stock azionari
- Controllo con più del 50% di azioni
- “Possesso indiretto” degli stock delle compagnie controllate

Esempio: Company Controls

Esempio (Company Controls)



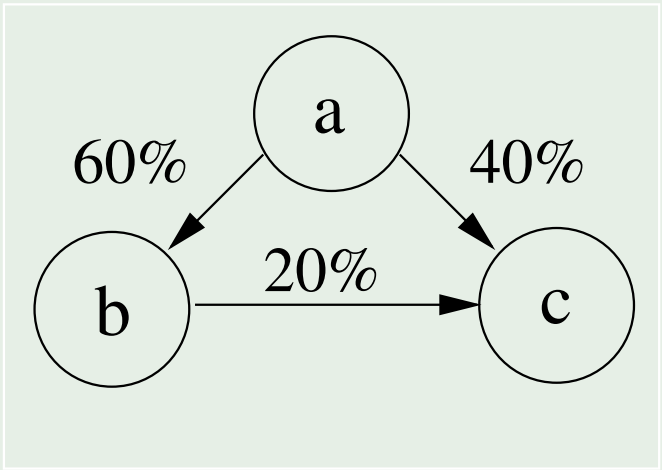

```
% companies      % stocks
company(a).      owns(a,b,60).
company(b).      owns(a,c,40).
company(c).      owns(b,c,20).

cv(X,X,Y,S) :- owns(X,Y,S).
cv(X,Z,Y,S) :- controls(X,Z), owns(Z,Y,S).

controls(X,Y) :- company(X), company(Y),
                 #sum {S, Z : cv(X,Z,Y,S)} > 50.
```

Esempio: Company Controls

Esempio (Company Controls)




```
% companies      % stocks
company(a).      owns(a,b,60).
company(b).      owns(a,c,40).
company(c).      owns(b,c,20).

cv(X,X,Y,S) :- owns(X,Y,S).
cv(X,Z,Y,S) :- controls(X,Z), owns(Z,Y,S).

controls(X,Y) :- company(X), company(Y),
                 #sum {S, Z : cv(X,Z,Y,S)} > 50.
```

Atomi aggregati

$$\#aggr\{\overline{X} : Conj\} \prec T$$

- $aggr \in \{count, sum, times, min, max\}$
- $\overline{X}$: lista di termini
- $Conj$: congiunzione di atomi standard
- $\prec \in \{<, \leq, >, \geq\}$
- T: termine (guardia)

Esempio (Atomo aggregato)

$$\#count\{X : employee(X, Y), Y > 30\} < 10$$

Vero se meno di 10 impiegati hanno più di 30 anni

Atomi aggregati

$$\#aggr\{\overline{X} : Conj\} \prec T$$

- $aggr \in \{count, sum, times, min, max\}$
- $\overline{X}$: lista di termini
- $Conj$: congiunzione di atomi standard
- $\prec \in \{<, \leq, >, \geq\}$
- T: termine (guardia)

Esempio (Atomo aggregato)

$$\#count\{X : employee(X, Y), Y > 30\} < 10$$

Vero se meno di 10 impiegati hanno più di 30 anni

Regole con aggregati

$$a_1 \vee \cdots \vee a_n \text{ :- } b_1, \ldots, b_k, \ not \ b_{k+1}, \ldots, \ not \ b_m.$$

- $a_1, \ldots, a_n$
  - Atomi standard
- $b_1, \ldots, b_m$
  - Atomi standard oppure atomi aggregati
- $b_1, \ldots, b_k$
  - Letterali positivi
- $not\ b_{k+1}, \ldots, not\ b_m$
  - Letterali negativi

Semantica

$$a_1 \vee \cdots \vee a_n \coloneq b_1, \ldots, b_k, \ not \ b_{k+1}, \ldots, \ not \ b_m.$$

Definizione

Un insieme di letterali standard M (un'interpretazione)
è un modello di una regola se è totale
e almeno uno fra $a_1, \ldots, a_n$ è in M
se $b_1, \ldots, b_k, not\ b_{k+1}, \ldots, not\ b_m$ sono tutti veri rispetto a M.

Definizione

M è un modello di un programma (un insieme di regole)
se è un modello di tutte le sue regole.

Semantica

$$a_1 \vee \cdots \vee a_n \coloneq b_1, \ldots, b_k, \ not \ b_{k+1}, \ldots, \ not \ b_m.$$

Definizione

Un insieme di letterali standard M (un'interpretazione)
è un modello di una regola se è totale
e almeno uno fra $a_1, \ldots, a_n$ è in M
se $b_1, \ldots, b_k, not\ b_{k+1}, \ldots, not\ b_m$ sono tutti veri rispetto a M.

Definizione

M è un modello di un programma (un insieme di regole)
se è un modello di tutte le sue regole.

Definizione di “riduzione”

“Riduzione” di un programma rispetto ad un’interpretazione in accordo con [Faber,Leone,Pfeifer 2004]:

- Elimina le regole per cui un letterale del corpo è falso

Modelli stabili: modelli minimali del programma ridotto

Teorema

Per programmi privi di aggregati questi modelli stabili coincidono con quelli definiti in [Gelfond, Lifschitz 1991].

Definizione di “riduzione”

“Riduzione” di un programma rispetto ad un’interpretazione in accordo con [Faber,Leone,Pfeifer 2004]:

- Elimina le regole per cui un letterale del corpo è falso

Modelli stabili: modelli minimali del programma ridotto

Teorema

Per programmi privi di aggregati questi modelli stabili coincidono con quelli definiti in [Gelfond, Lifschitz 1991].

Monotonicità e antimonotonicità

- I e J due interpretazioni. $I \leq J$ se e solo se
  - $I^+ \subseteq J^+$
  - $I^- \supseteq J^-$
- Letterali monotoni
  - verità per I implica verità per ogni J t.c. $I \leq J$
- Letterali antimonotoni
  - verità per J implica verità per ogni I t.c. $I \leq J$
- Letterali nonmonotoni
  - né monotoni né antimonotoni

Monotonicità e antimonotonicità

- I e J due interpretazioni. $\mathrm{I} \leq \mathrm{J}$ se e solo se
  - $\mathrm{I}^+ \subseteq \mathrm{J}^+$
  - $\mathrm{I}^- \supseteq \mathrm{J}^-$
- Letterali monotoni
  - verità per I implica verità per ogni J t.c. $\mathrm{I} \leq \mathrm{J}$
- Letterali antimonotoni
  - verità per J implica verità per ogni I t.c. $\mathrm{I} \leq \mathrm{J}$
- Letterali nonmonotoni
  - né monotoni né antimonotoni

Monotonicità e antimonotonicità

- I e J due interpretazioni. $\mathrm{I} \leq \mathrm{J}$ se e solo se
  - $\mathrm{I}^+ \subseteq \mathrm{J}^+$
  - $\mathrm{I}^- \supseteq \mathrm{J}^-$
- Letterali monotoni
  - verità per I implica verità per ogni J t.c. $\mathrm{I} \leq \mathrm{J}$
- Letterali antimonotoni
  - verità per J implica verità per ogni I t.c. $\mathrm{I} \leq \mathrm{J}$
- Letterali nonmonotoni
  - né monotoni né antimonotoni

Monotonicità e antimonotonicità

- I e J due interpretazioni. $\mathrm{I} \leq \mathrm{J}$ se e solo se
  - $\mathrm{I}^+ \subseteq \mathrm{J}^+$
  - $\mathrm{I}^- \supseteq \mathrm{J}^-$
- Letterali monotoni
  - verità per I implica verità per ogni J t.c. $\mathrm{I} \leq \mathrm{J}$
- Letterali antimonotoni
  - verità per J implica verità per ogni I t.c. $\mathrm{I} \leq \mathrm{J}$
- Letterali nonmonotoni
  - né monotoni né antimonotoni

Monotonicità: esempi

- $\#count\{\ldots\} \geq 1$ è monotono
- $\#count\{\ldots\} < 1$ è antimonotono
- $\#avg\{\ldots\} < 3$ è nonmonotono
- I letterali standard positivi sono monotoni
- I letterali standard negativi sono antimonotoni

Definizione ($DLP^{\mathcal{A}}_{m,a}$)

DLP con aggregati ricorsivi monotoni e antimonotoni

Monotonicità: esempi

- $\#count\{\ldots\} \geq 1$ è monotono
- $\#count\{\ldots\} < 1$ è antimonotono
- $\#avg\{\ldots\} < 3$ è nonmonotono
- I letterali standard positivi sono monotoni
- I letterali standard negativi sono antimonotoni

Definizione ($DLP^{\mathcal{A}}_{m,a}$)

DLP con aggregati ricorsivi monotoni e antimonotoni

Monotonicità: esempi

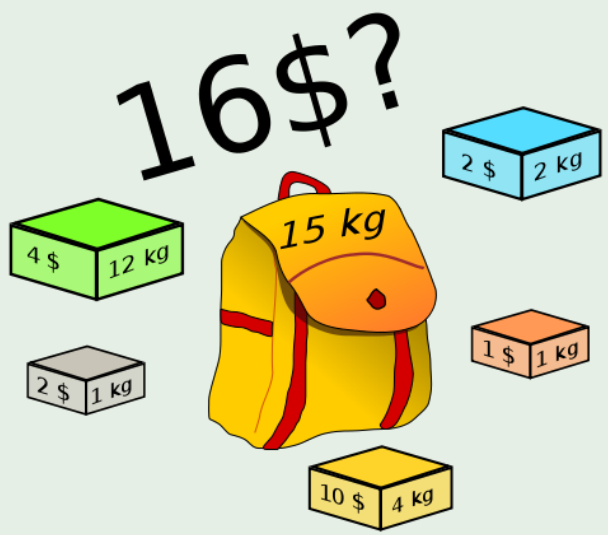
- $\#count\{\ldots\} \geq 1$ è monotono
- $\#count\{\ldots\} < 1$ è antimonotono
- $\#avg\{\ldots\} < 3$ è nonmonotono
- I letterali standard positivi sono monotoni
- I letterali standard negativi sono antimonotoni

Definizione ($DLP^{\mathcal{A}}_{m,a}$)

DLP con aggregati ricorsivi monotoni e antimonotoni

Esempio: Knap-sack

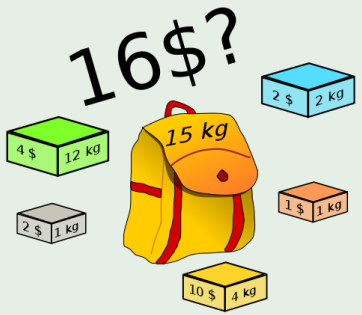
Esempio (Knap-sack)



- Oggetti con peso e valore
- Peso massimo sostenibile
- Valore minimo voluto

Esempio: Knap-sack

Esempio (Knap-sack)

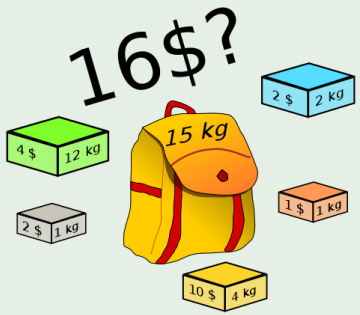



```
cost(green,12).    value(green,4).
cost(gray,1).      value(gray,2).
...                ...
maxCost(15).       minValue(16).
```

```
select(X) :- cost(X,C), maxCost( MC ),
             #sum {C1, Y : select( Y ), cost( Y, C1 ), X != Y} <= MC-C.
:- minValue( MV ), not #sum {C, X : select( X ), value( X, C )} >= MV.
```

Esempio: Knap-sack

Esempio (Knap-sack)



cost(green,12).	value(green,4).
cost(gray,1).	value(gray,2).
...	...
maxCost(15).	minValue(16).

```
select(X) :- cost(X,C), maxCost( MC ),
             #sum {C1, Y : select( Y ), cost( Y, C1 ), X != Y} <= MC-C.
:- minValue( MV ), not #sum {C, X : select( X ), value( X, C )} >= MV.
```

Contributi

- Analisi delle proprietà del linguaggio $\text{DLP}^{\mathcal{A}}_{m,a}$
  - Sottoclassi di problemi con modello unico
  - Definizione di insieme infondato
  - Caratterizzazione dei modelli stabili
- Progettazione di algoritmi per la valutazione efficiente
  - Strategie di istanziazione
  - Operatore per il GUS (e computazione modulare)
  - Operatori per il taglio dello spazio di ricerca: $\mathcal{T}_{\mathcal{P}}, \Phi_{\mathcal{P}}, \mathcal{W}_{\mathcal{P}}$
- Implementazione e sperimentazione del prototipo

Contributi

- Analisi delle proprietà del linguaggio $\text{DLP}^{\mathcal{A}}_{m,a}$
  - Sottoclassi di problemi con modello unico
  - Definizione di insieme infondato
  - Caratterizzazione dei modelli stabili
- Progettazione di algoritmi per la valutazione efficiente
  - Strategie di istanziazione
  - Operatore per il GUS (e computazione modulare)
  - Operatori per il taglio dello spazio di ricerca: $\mathcal{T}_{\mathcal{P}}, \Phi_{\mathcal{P}}, \mathcal{W}_{\mathcal{P}}$
- Implementazione e sperimentazione del prototipo

Insiemi infondati per programmi standard

[Leone, Rullo, Scarcello 1997] definiscono gli insiemi infondati per programmi standard (senza aggregati)

Definizione

Un insieme X di atomi standard
è un insieme infondato rispetto ad I
se per tutte le regole con qualche atomo della testa in X

1. il corpo è falso rispetto ad I oppure
2. qualche letterale positivo del corpo appartiene ad X oppure
3. qualche atomo della testa non appartenente ad X è vero rispetto ad I

Insiemi infondati per programmi con aggregati

Estendiamo questa definizione ai programmi con aggregati

Definizione (Insieme infondato)

Un insieme X di atomi standard
è un insieme infondato rispetto ad I
se per ogni regola con qualche atomo della testa in X

1. qualche letterale antimonotono del corpo è falso rispetto ad I oppure
2. qualche letterale monotono del corpo è falso rispetto ad $I \dot{\cup} \neg.X$ oppure
3. qualche atomo della testa è vero rispetto ad $I \dot{\cup} \neg.X$

Greatest Unfounded Set

Definizione (Greatest Unfouded Set)

Denotiamo l'unione di tutti gli insiemi infondati per $\mathcal{P}$ rispetto ad I con $GUS_{\mathcal{P}}(\mathrm{I})$
(il "più grande insieme infondato" per $\mathcal{P}$ rispetto ad I)

- GUS non è sempre un insieme infondato
- L'unione di più insiemi infondati è un insieme infondato se questi sono disgiunti dall'interpretazione
- Un'interpretazione disgiunta da tutti i suoi insiemi infondati viene detta unfounded-free

Greatest Unfounded Set

Definizione (Greatest Unfouded Set)

Denotiamo l'unione di tutti gli insiemi infondati per $\mathcal{P}$ rispetto ad I con $GUS_{\mathcal{P}}(\mathrm{I})$
(il "più grande insieme infondato" per $\mathcal{P}$ rispetto ad I)

- GUS non è sempre un insieme infondato
- L'unione di più insiemi infondati è un insieme infondato se questi sono disgiunti dall'interpretazione
- Un'interpretazione disgiunta da tutti i suoi insiemi infondati viene detta unfounded-free

Greatest Unfounded Set

Definizione (Greatest Unfouded Set)

Denotiamo l'unione di tutti gli insiemi infondati per $\mathcal{P}$ rispetto ad I con $GUS_{\mathcal{P}}(\mathrm{I})$
(il "più grande insieme infondato" per $\mathcal{P}$ rispetto ad I)

- GUS non è sempre un insieme infondato
- L'unione di più insiemi infondati è un insieme infondato se questi sono disgiunti dall'interpretazione
- Un'interpretazione disgiunta da tutti i suoi insiemi infondati viene detta unfounded-free

Greatest Unfounded Set

Definizione (Greatest Unfouded Set)

Denotiamo l'unione di tutti gli insiemi infondati per $\mathcal{P}$ rispetto ad I con $GUS_{\mathcal{P}}(\mathrm{I})$
(il "più grande insieme infondato" per $\mathcal{P}$ rispetto ad I)

- GUS non è sempre un insieme infondato
- L'unione di più insiemi infondati è un insieme infondato se questi sono disgiunti dall'interpretazione
- Un'interpretazione disgiunta da tutti i suoi insiemi infondati viene detta unfounded-free

Operatore $\mathcal{R}_{\mathcal{P},I}$

L'operatore $\mathcal{R}_{\mathcal{P},\mathrm{I}}$

- A partire da un insieme di atomi X
- Scarta gli atomi per cui non è nota l'infondatezza
- Decresce monotonicamente e converge a $\mathcal{R}^{\omega}_{\mathcal{P},\mathrm{I}}(\mathrm{X})$

Teorema

$\mathcal{R}^{\omega}_{\mathcal{P},\mathrm{I}}(B_{\mathcal{P}} \setminus \mathrm{I}) = GUS_{\mathcal{P}}(\mathrm{I})$ *se* I *è unfounded-free*

Operatore $\mathcal{R}_{\mathcal{P},I}$

L'operatore $\mathcal{R}_{\mathcal{P},\mathrm{I}}$

- A partire da un insieme di atomi X
- Scarta gli atomi per cui non è nota l'infondatezza
- Decresce monotonicamente e converge a $\mathcal{R}^{\omega}_{\mathcal{P},\mathrm{I}}(\mathrm{X})$

Teorema

$\mathcal{R}^{\omega}_{\mathcal{P},\mathrm{I}}(B_{\mathcal{P}} \setminus \mathrm{I}) = \mathit{GUS}_{\mathcal{P}}(\mathrm{I})$ *se* I *è unfounded-free*

Valutazione modulare del GUS

$C_1, \ldots, C_n$ un ordine totale per le componenti

$I_0 := I$
FOR i $:= 1$ TO n DO
$X_i := \mathcal{R}^{\omega}_{\mathcal{P}_{C_i}, I_{i-1}}(C_i \setminus I)$
$I_i := I_{i-1} \cup \neg.X_i$

Teorema

Otteniamo $I_n = I \cup \neg.GUS_{\mathcal{P}}(I)$

Valutazione modulare del GUS

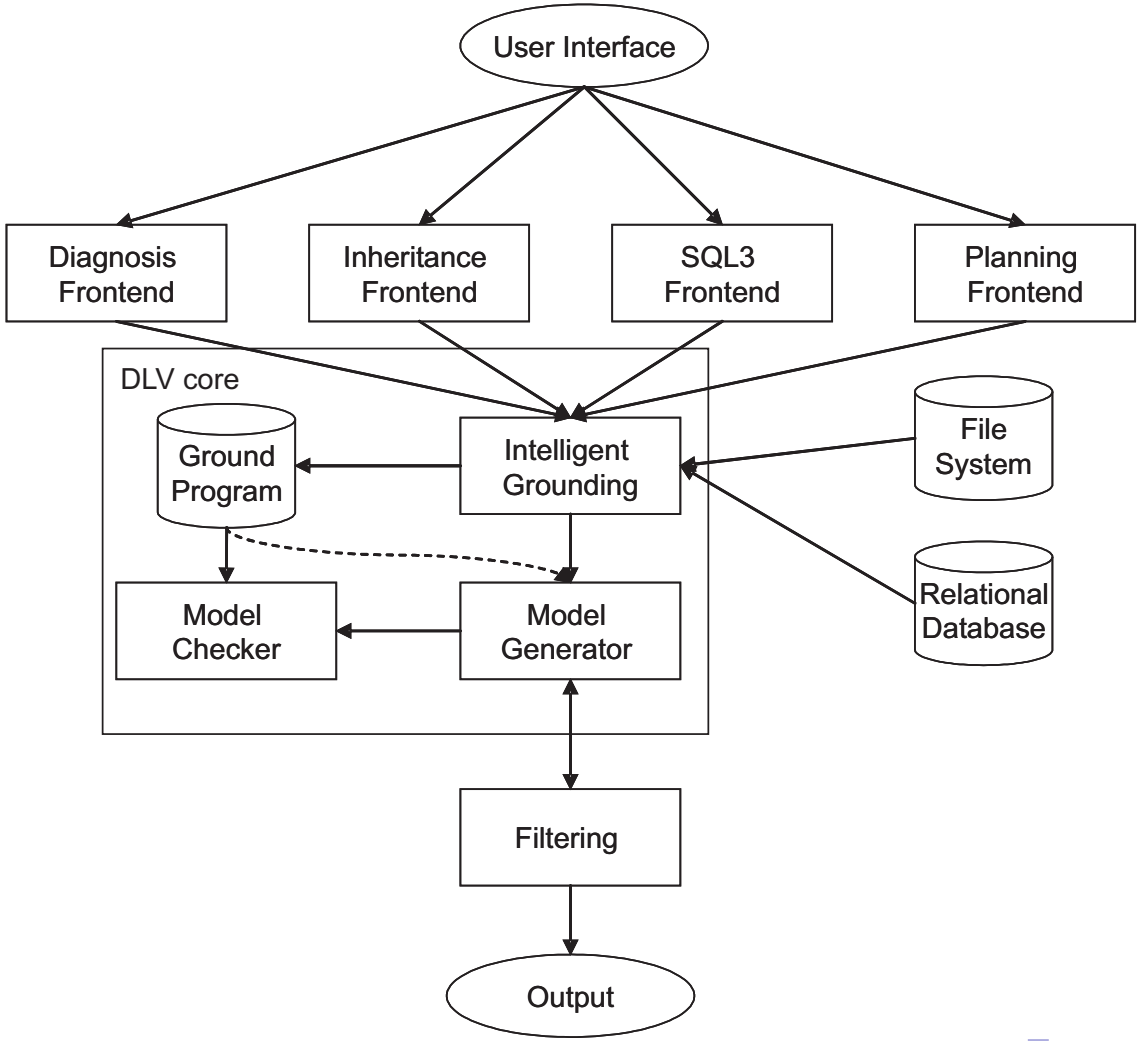
$C_1, \ldots, C_n$ un ordine totale per le componenti

$I_0 := I$
FOR i $:= 1$ TO n DO
$X_i := \mathcal{R}^{\omega}_{\mathcal{P}_{C_i}, I_{i-1}}(C_i \setminus I)$
$I_i := I_{i-1} \cup \neg.X_i$

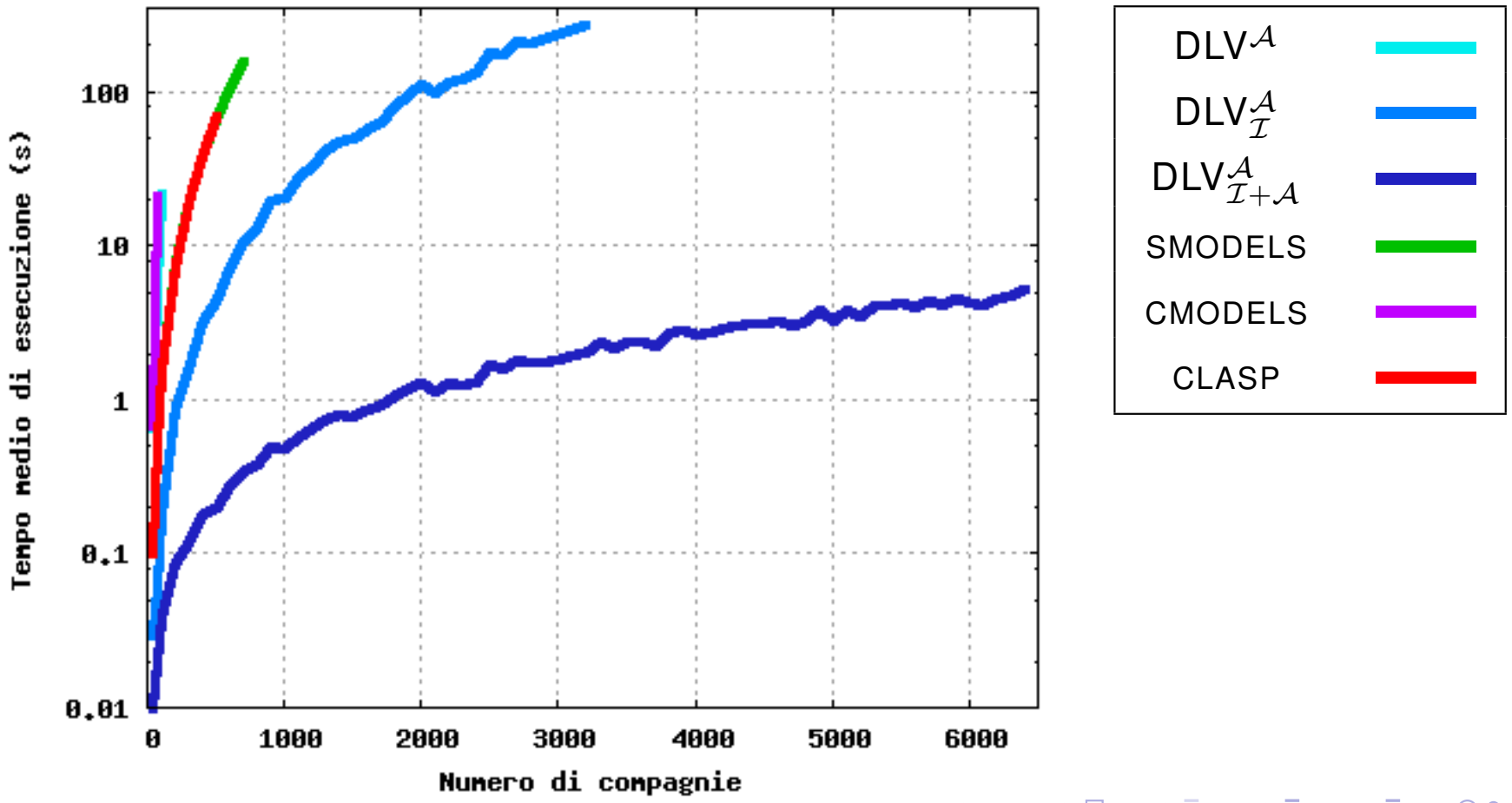
Teorema

Otteniamo $I_n = I \cup \neg.GUS_{\mathcal{P}}(I)$

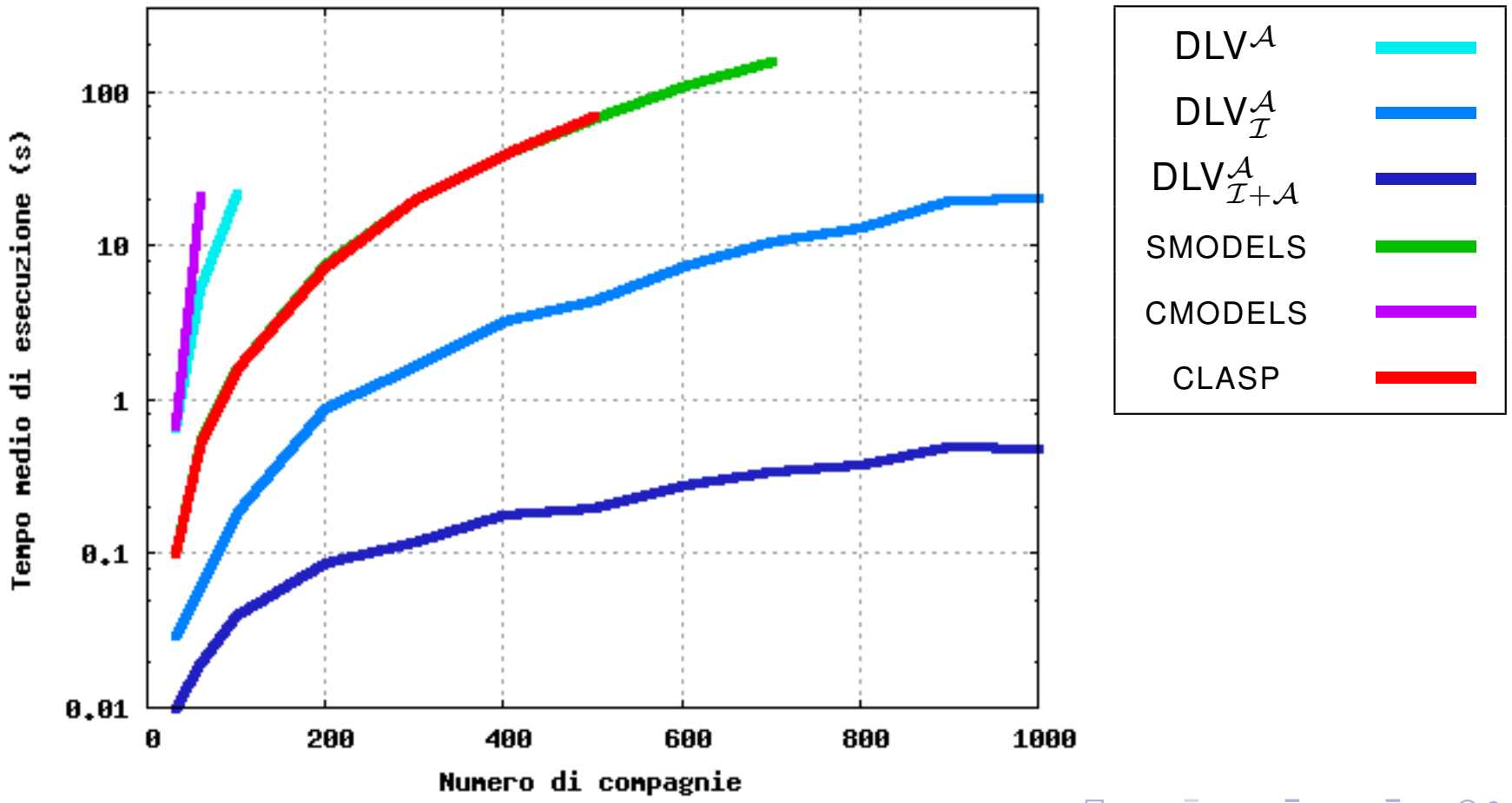
Architettura del prototipo



Risultati sperimentali (Company Controls)



Risultati sperimentali (Company Controls)



Conclusioni

- Alta espressività del linguaggio
- Risultati sperimentali positivi
- Il prototipo è disponibile in rete

  `http://www.dlvsystem.com/dlvRecAggr`
- Presentazione al CILC07:
  Using Unfounded Sets for Computing Answer Sets of Programs with Recursive Aggregates