Constraint Programming & Planning

Agostino Dovier

Università di Udine
Dipartimento di Matematica e Informatica

Udine, Novembre 2012

Knowledge Representation and Reasoning
Il sillogismo
La perdita della monotonia
Programmi Definiti
Richiami
Semantica
Grounding
Programmi Generali
Definizioni
La perdita della monotonia
La semantica

Sommario

Knowledge Representation and Reasoning
- Il sillogismo
- La perdita della monotonia

Programmi Definiti
- Richiami
- Semantica
- Grounding

Programmi Generali
- Definizioni
- La perdita della monotonia
- La semantica

Introduction

Knowledge representation is one of the most important subareas of artificial intelligence. If we want to design an entity (a machine or a program) capable of behaving intelligently in some environment, then we need to supply this entity with sufficient knowledge about this environment. To do that, we need an unambiguous language capable of expressing this knowledge, together with some precise and well understood way of manipulating sets of sentences of the language which will allow us to draw inferences, answer queries, and to update both the knowledge base and the desired program behavior.

Chitta Baral and Michael Gelfond. Logic Programming and Knowledge Representation Journal of Logic Programming 19/20, 73–148, 1994. (disponibile on-line il RR di UTEP dalle pagine degli autori)

Introduction

Expressing information in declarative sentences is far more modular than expressing it in segments of computer programs or in tables. Sentences can be true in a much wider context than specific programs can be used. The supplier of a fact does not have to understand much about how the receiver functions or how or whether the receiver will use it. The same fact can be used for many purposes, because the logical consequences of collections of facts can be available. [McC59]

Chitta Baral and Michael Gelfond. Logic Programming and Knowledge Representation Journal of Logic Programming 19/20, 73–148, 1994. (disponibile on-line il RR di UTEP dalle pagine degli autori)

Introduction

This idea has been further developed by many researchers with various backgrounds and interests. First, the classical logic of predicate calculus served as the main technical tool for the representation of knowledge. It has a well-defined semantics and a well-understood and powerful inference mechanism, and it proved to be sufficiently expressive for the representation of mathematical knowledge.

Chitta Baral and Michael Gelfond. Logic Programming and Knowledge Representation Journal of Logic Programming 19/20, 73–148, 1994. (disponibile on-line il RR di UTEP dalle pagine degli autori)

Introduction

It was soon realized, however, that for the representation of *commonsense knowledge*, this tool is inadequate. The difficulty is rather deep and related to the so-called "monotonicity" of theories based on predicate calculus. A logic is called *monotonic* if the addition of new axioms to a theory based on it never leads to the loss of any theorems proved in this theory. Commonsense reasoning is nonmonotonic: new information constantly forces us to withdraw previous conclusions. This observation has led to the development and investigation of new logical formalisms, *nonmonotonic logics*.

Chitta Baral and Michael Gelfond. Logic Programming and Knowledge Representation Journal of Logic Programming 19/20, 73–148, 1994. (disponibile on-line il RR di UTEP dalle pagine degli autori)

Introduzione

Il sillogismo

Socrate è un uomo

Tutti gli uomini sono mortali

Pertanto Socrate è mortale

Introduzione

Il sillogismo

Socrate è un uomo
Tutti gli uomini sono mortali
Pertanto Socrate è mortale

Knowledge Representation and Reasoning
Il sillogismo
La perdita della monotonia
Programmi Definiti
Richiami
Semantica
Grounding
Programmi Generali
Definizioni
La perdita della monotonia
La semantica

Introduzione

Il sillogismo

Socrate è un uomo
Tutti gli uomini sono mortali
Pertanto Socrate è mortale

Modelliamo la cosa con la logica del prim'ordine.

Socrate è un uomo:

$$\texttt{uomo}(\texttt{socrate}) \tag{1}$$

Tutti gli uomini sono mortali:

$$\forall X\,(\texttt{uomo}(X) \rightarrow \texttt{mortale}(X)) \tag{2}$$

Riflettiamo sul verso dell'implicazione. Ad esempio, se `rex` è un cane, avremo che `mortale(rex)` ma non `uomo(rex)` Si dovrebbe sapere che chi è cane non è uomo, a dire il vero, ne parleremo (CWA).
Inoltre, vale che

$$(1), (2) \models \texttt{mortale}(\texttt{socrate})$$

KNOWLEDGE REPRESENTATION AND REASONING
IL SILLOGISMO
LA PERDITA DELLA MONOTONIA
PROGRAMMI DEFINITI
RICHIAMI
SEMANTICA
GROUNDING
PROGRAMMI GENERALI
DEFINIZIONI
LA PERDITA DELLA MONOTONIA
LA SEMANTICA

Introduzione
Il sillogismo

Modelliamo la cosa con la logica del prim'ordine.
Socrate è un uomo:

$$\texttt{uomo}(\texttt{socrate}) \tag{1}$$

Tutti gli uomini sono mortali:

$$\forall X\,(\texttt{uomo}(X) \rightarrow \texttt{mortale}(X)) \tag{2}$$

Riflettiamo sul verso dell'implicazione. Ad esempio, se `rex` è un cane, avremo che `mortale(rex)` ma non `uomo(rex)` Si dovrebbe sapere che chi è cane non è uomo, a dire il vero, ne parleremo (CWA).
Inoltre, vale che

$$(1), (2) \models \texttt{mortale}(\texttt{socrate})$$

Introduzione
Il sillogismo

Modelliamo la cosa con la logica del prim'ordine.
Socrate è un uomo:

$$\texttt{uomo}(\texttt{socrate}) \tag{1}$$

Tutti gli uomini sono mortali:

$$\forall X\,(\texttt{uomo}(X) \rightarrow \texttt{mortale}(X)) \tag{2}$$

Riflettiamo sul verso dell'implicazione. Ad esempio, se `rex` è un cane, avremo che `mortale(rex)` ma non `uomo(rex)` Si dovrebbe sapere che chi è cane non è uomo, a dire il vero, ne parleremo (CWA).
Inoltre, vale che

$$(1), (2) \models \texttt{mortale}(\texttt{socrate})$$

INTRODUZIONE

IL SILLOGISMO

Modelliamo la cosa con la logica del prim'ordine.
Socrate è un uomo:

$$\texttt{uomo}(\texttt{socrate}) \tag{1}$$

Tutti gli uomini sono mortali:

$$\forall X\,(\texttt{uomo}(X) \rightarrow \texttt{mortale}(X)) \tag{2}$$

Riflettiamo sul verso dell'implicazione. Ad esempio, se `rex` è un cane, avremo che `mortale(rex)` ma non `uomo(rex)` Si dovrebbe sapere che chi è cane non è uomo, a dire il vero, ne parleremo (CWA).
Inoltre, vale che

$$(1), (2) \models \texttt{mortale}(\texttt{socrate})$$

KNOWLEDGE REPRESENTATION AND REASONING
IL SILLOGISMO
LA PERDITA DELLA MONOTONIA
PROGRAMMI DEFINITI
RICHIAMI
SEMANTICA
GROUNDING
PROGRAMMI GENERALI
DEFINIZIONI
LA PERDITA DELLA MONOTONIA
LA SEMANTICA

KNOWLEDGE REPRESENTATION AND REASONING
IL SILLOGISMO
LA PERDITA DELLA MONOTONIA
PROGRAMMI DEFINITI
RICHIAMI
SEMANTICA
GROUNDING
PROGRAMMI GENERALI
DEFINIZIONI
LA PERDITA DELLA MONOTONIA
LA SEMANTICA

INTRODUZIONE

IL SILLOGISMO

Modelliamo la cosa con la logica del prim'ordine.
Socrate è un uomo:

$$\texttt{uomo}(\texttt{socrate}) \tag{1}$$

Tutti gli uomini sono mortali:

$$\forall X\,(\texttt{uomo}(X) \rightarrow \texttt{mortale}(X)) \tag{2}$$

Riflettiamo sul verso dell'implicazione. Ad esempio, se `rex` è un cane, avremo che `mortale(rex)` ma non `uomo(rex)` Si dovrebbe sapere che chi è cane non è uomo, a dire il vero, ne parleremo (CWA).
Inoltre, vale che

$$(1), (2) \models \texttt{mortale}(\texttt{socrate})$$

Introduzione
Il sillogismo

Modelliamo la cosa con la logica del prim'ordine.
Socrate è un uomo:

$$\texttt{uomo}(\texttt{socrate}) \tag{1}$$

Tutti gli uomini sono mortali:

$$\forall X\,(\texttt{uomo}(X) \rightarrow \texttt{mortale}(X)) \tag{2}$$

Riflettiamo sul verso dell'implicazione. Ad esempio, se `rex` è un cane, avremo che `mortale(rex)` ma non `uomo(rex)` Si dovrebbe sapere che chi è cane non è uomo, a dire il vero, ne parleremo (CWA).
Inoltre, vale che

$$(1), (2) \models \texttt{mortale}(\texttt{socrate})$$

Introduzione
Il sillogismo

Modelliamo la cosa con la logica del prim'ordine.
Socrate è un uomo:

$$\texttt{uomo}(\texttt{socrate}) \tag{1}$$

Tutti gli uomini sono mortali:

$$\forall X\,(\texttt{uomo}(X) \rightarrow \texttt{mortale}(X)) \tag{2}$$

Riflettiamo sul verso dell'implicazione. Ad esempio, se `rex` è un cane, avremo che `mortale(rex)` ma non `uomo(rex)` Si dovrebbe sapere che chi è cane non è uomo, a dire il vero, ne parleremo (CWA).
Inoltre, vale che

$$(1), (2) \models \texttt{mortale}(\texttt{socrate})$$

Introduzione
Il sillogismo

Modelliamo la cosa con la logica del prim'ordine.
Socrate è un uomo:

$$\texttt{uomo}(\texttt{socrate}) \tag{1}$$

Tutti gli uomini sono mortali:

$$\forall X\,(\texttt{uomo}(X) \rightarrow \texttt{mortale}(X)) \tag{2}$$

Riflettiamo sul verso dell'implicazione. Ad esempio, se `rex` è un cane, avremo che `mortale(rex)` ma non `uomo(rex)` Si dovrebbe sapere che chi è cane non è uomo, a dire il vero, ne parleremo (CWA).
Inoltre, vale che

$$(1), (2) \models \texttt{mortale}(\texttt{socrate})$$

Inoltre, se ora scopriamo che

$$\texttt{uomo}(\texttt{agostino}) \tag{3}$$

allora vale che

$$(1), (2), (3) \models \texttt{mortale}(\texttt{socrate})$$

e

$$(1), (2), (3) \models \texttt{mortale}(\texttt{agostino})$$

All'aumentare delle premesse, aumentano i teoremi (e quelli vecchi rimangono validi). Siamo in presenza di monotonia.

KNOWLEDGE REPRESENTATION AND REASONING
IL SILLOGISMO
LA PERDITA DELLA MONOTONIA
PROGRAMMI DEFINITI
RICHIAMI
SEMANTICA
GROUNDING
PROGRAMMI GENERALI
DEFINIZIONI
LA PERDITA DELLA MONOTONIA
LA SEMANTICA

Inoltre, se ora scopriamo che

$$\texttt{uomo}(\texttt{agostino}) \tag{3}$$

allora vale che

$$(1), (2), (3) \models \texttt{mortale}(\texttt{socrate})$$

e

$$(1), (2), (3) \models \texttt{mortale}(\texttt{agostino})$$

All'aumentare delle premesse, aumentano i teoremi (e quelli vecchi rimangono validi). Siamo in presenza di monotonia.

KNOWLEDGE REPRESENTATION AND REASONING
IL SILLOGISMO
LA PERDITA DELLA MONOTONIA
PROGRAMMI DEFINITI
RICHIAMI
SEMANTICA
GROUNDING
PROGRAMMI GENERALI
DEFINIZIONI
LA PERDITA DELLA MONOTONIA
LA SEMANTICA

Introduzione

La perdita della monotonia

Se si attraversano i binari quando passa il treno non si raggiunge l'altra parte.

Se si attraversano i binari quando non c'è il treno si raggiunge l'altra parte.

Se le sbarre sono abbassate c'è il treno.

Vogliamo andare dall'altra parte.

Le sbarre non sono abbassate.

Alla luce di questa conoscenza sembrerebbe plausibile attraversare

Se c'è un guasto elettrico, allora le sbarre sono sempre alzate o sempre abbassate.

E con questa nuova informazione, attraverseste?

E se sapessimo anche che c'è un guasto elettrico?

Introduzione

La perdita della monotonia

Se si attraversano i binari quando passa il treno non si raggiunge l'altra parte.
Se si attraversano i binari quando non c'è il treno si raggiunge l'altra parte.
Se le sbarre sono abbassate c'è il treno.
Vogliamo andare dall'altra parte.
Le sbarre non sono abbassate.
Alla luce di questa conoscenza sembrerebbe plausibile attraversare

Se c'è un guasto elettrico, allora le sbarre sono sempre alzate o sempre abbassate.

E con questa nuova informazione, attraverseste?

E se sapessimo anche che c'è un guasto elettrico?

Introduzione

La perdita della monotonia

Se si attraversano i binari quando passa il treno non si raggiunge l'altra parte.
Se si attraversano i binari quando non c'è il treno si raggiunge l'altra parte.
Se le sbarre sono abbassate c'è il treno.
Vogliamo andare dall'altra parte.
Le sbarre non sono abbassate.
Alla luce di questa conoscenza sembrerebbe plausibile attraversare

Se c'è un guasto elettrico, allora le sbarre sono sempre alzate o sempre abbassate.

E con questa nuova informazione, attraverseste?

E se sapessimo anche che c'è un guasto elettrico?

Introduzione

La perdita della monotonia

Se si attraversano i binari quando passa il treno non si raggiunge l'altra parte.
Se si attraversano i binari quando non c'è il treno si raggiunge l'altra parte.
Se le sbarre sono abbassate c'è il treno.
Vogliamo andare dall'altra parte.
Le sbarre non sono abbassate.
Alla luce di questa conoscenza sembrerebbe plausibile attraversare

Se c'è un guasto elettrico, allora le sbarre sono sempre alzate o sempre abbassate.

E con questa nuova informazione, attraverseste?

E se sapessimo anche che c'è un guasto elettrico?

Introduzione

La perdita della monotonia

Se si attraversano i binari quando passa il treno non si raggiunge l'altra parte.
Se si attraversano i binari quando non c'è il treno si raggiunge l'altra parte.
Se le sbarre sono abbassate c'è il treno.
Vogliamo andare dall'altra parte.
Le sbarre non sono abbassate.
Alla luce di questa conoscenza sembrerebbe plausibile attraversare

Se c'è un guasto elettrico, allora le sbarre sono sempre alzate o sempre abbassate.

E con questa nuova informazione, attraverseste?

E se sapessimo anche che c'è un guasto elettrico?

Introduzione

La perdita della monotonia

Se si attraversano i binari quando passa il treno non si raggiunge l'altra parte.
Se si attraversano i binari quando non c'è il treno si raggiunge l'altra parte.
Se le sbarre sono abbassate c'è il treno.
Vogliamo andare dall'altra parte.
Le sbarre non sono abbassate.
Alla luce di questa conoscenza sembrerebbe plausibile attraversare

Se c'è un guasto elettrico, allora le sbarre sono sempre alzate o sempre abbassate.

E con questa nuova informazione, attraverseste?

E se sapessimo anche che c'è un guasto elettrico?

Introduzione

La perdita della monotonia

Se si attraversano i binari quando passa il treno non si raggiunge l'altra parte.
Se si attraversano i binari quando non c'è il treno si raggiunge l'altra parte.
Se le sbarre sono abbassate c'è il treno.
Vogliamo andare dall'altra parte.
Le sbarre non sono abbassate.
Alla luce di questa conoscenza sembrerebbe plausibile attraversare

Se c'è un guasto elettrico, allora le sbarre sono sempre alzate o sempre abbassate.

E con questa nuova informazione, attraverseste?

E se sapessimo anche che c'è un guasto elettrico?

Introduzione

La perdita della monotonia

Se si attraversano i binari quando passa il treno non si raggiunge l'altra parte.
Se si attraversano i binari quando non c'è il treno si raggiunge l'altra parte.
Se le sbarre sono abbassate c'è il treno.
Vogliamo andare dall'altra parte.
Le sbarre non sono abbassate.
Alla luce di questa conoscenza sembrerebbe plausibile attraversare

Se c'è un guasto elettrico, allora le sbarre sono sempre alzate o sempre abbassate.

E con questa nuova informazione, attraverseste?

E se sapessimo anche che c'è un guasto elettrico?

Introduzione

La perdita della monotonia

Se si attraversano i binari quando passa il treno non si raggiunge l'altra parte.
Se si attraversano i binari quando non c'è il treno si raggiunge l'altra parte.
Se le sbarre sono abbassate c'è il treno.
Vogliamo andare dall'altra parte.
Le sbarre non sono abbassate.
Alla luce di questa conoscenza sembrerebbe plausibile attraversare

Se c'è un guasto elettrico, allora le sbarre sono sempre alzate o sempre abbassate.

E con questa nuova informazione, attraverseste?

E se sapessimo anche che c'è un guasto elettrico?

Introduzione

La perdita della monotonia

Nel commonsense reasoning si fa uso continuo di asserzioni note come normative statements, ovvero del tipo:

$$A\text{'s are normally } B\text{'s}$$

che ammettono però delle eccezioni.

Introduzione

La perdita della monotonia: i pinguini

Suppose that a reasoning agent has the following knowledge about birds: birds typically fly and penguins are non flying birds. He also knows that Tweety is a bird. Suppose now that the agent is hired to build a cage for Tweety, and he leaves off the roof on the grounds that he does not whether or nor Tweety can fly. It would be reasonable for us to view this argument as invalid and to refuse the agent's product. This would be not the case if Tweety could not fly for some reason (unknown to the agent), and we refused to pay for the bird cage because had unnecessarily put a roof on it. [McCarthy, 1959]

Introduzione

La perdita della monotonia: i pinguini

Ogni uccello, che non sia anormale, vola.
Ogni pinguino è un uccello.
Ogni canarino è un uccello.
Ogni pinguino è anormale.
Metti il tetto alla gabbietta per un uccello che vola.
tweety è un uccello.
Costruisci la gabbietta per tweety
tweety è un canarino
Pensate invece a pingu, che è un pinguino.

Introduzione

La perdita della monotonia: i pinguini

Ogni uccello, che non sia anormale, vola.
Ogni pinguino è un uccello.
Ogni canarino è un uccello.
Ogni pinguino è anormale.
Metti il tetto alla gabbietta per un uccello che vola.
tweety è un uccello.
Costruisci la gabbietta per tweety
tweety è un canarino
Pensate invece a pingu, che è un pinguino.

Knowledge Representation and Reasoning
Il sillogismo
La perdita della monotonia
Programmi Definiti
Richiami
Semantica
Grounding
Programmi Generali
Definizioni
La perdita della monotonia
La semantica

Introduzione

La perdita della monotonia: i pinguini

Ogni uccello, che non sia anormale, vola.
Ogni pinguino è un uccello.
Ogni canarino è un uccello.
Ogni pinguino è anormale.
Metti il tetto alla gabbietta per un uccello che vola.
tweety è un uccello.
Costruisci la gabbietta per tweety
tweety è un canarino
Pensate invece a pingu, che è un pinguino.

Introduzione

La perdita della monotonia: i pinguini

Ogni uccello, che non sia anormale, vola.
Ogni pinguino è un uccello.
Ogni canarino è un uccello.
Ogni pinguino è anormale.
Metti il tetto alla gabbietta per un uccello che vola.
tweety è un uccello.
Costruisci la gabbietta per tweety
tweety è un canarino
Pensate invece a pingu, che è un pinguino.

Introduzione

La perdita della monotonia: i pinguini

Ogni uccello, che non sia anormale, vola.
Ogni pinguino è un uccello.
Ogni canarino è un uccello.
Ogni pinguino è anormale.
Metti il tetto alla gabbietta per un uccello che vola.
tweety è un uccello.
Costruisci la gabbietta per tweety
tweety è un canarino
Pensate invece a pingu, che è un pinguino.

Introduzione

La perdita della monotonia: i pinguini

Ogni uccello, che non sia anormale, vola.
Ogni pinguino è un uccello.
Ogni canarino è un uccello.
Ogni pinguino è anormale.
Metti il tetto alla gabbietta per un uccello che vola.
tweety è un uccello.
Costruisci la gabbietta per tweety
tweety è un canarino
Pensate invece a pingu, che è un pinguino.

Introduzione

La perdita della monotonia: i pinguini

Ogni uccello, che non sia anormale, vola.
Ogni pinguino è un uccello.
Ogni canarino è un uccello.
Ogni pinguino è anormale.
Metti il tetto alla gabbietta per un uccello che vola.
tweety è un uccello.
Costruisci la gabbietta per tweety
tweety è un canarino
Pensate invece a pingu, che è un pinguino.

Introduzione

La perdita della monotonia: i pinguini

Ogni uccello, che non sia anormale, vola.
Ogni pinguino è un uccello.
Ogni canarino è un uccello.
Ogni pinguino è anormale.
Metti il tetto alla gabbietta per un uccello che vola.
tweety è un uccello.
Costruisci la gabbietta per tweety
tweety è un canarino
Pensate invece a pingu, che è un pinguino.

Introduzione

La perdita della monotonia: i pinguini

Ogni uccello, che non sia anormale, vola.
Ogni pinguino è un uccello.
Ogni canarino è un uccello.
Ogni pinguino è anormale.
Metti il tetto alla gabbietta per un uccello che vola.
tweety è un uccello.
Costruisci la gabbietta per tweety
tweety è un canarino
Pensate invece a pingu, che è un pinguino.

Introduzione

La perdita della monotonia: i pinguini

Ogni uccello, che non sia anormale, vola (per $\leftarrow$, pensate alle api)

$$\forall X\,(\texttt{uccello}(X) \wedge \neg\texttt{ab}(X) \rightarrow \texttt{vola}(X)) \tag{4}$$

Ogni pinguino è un uccello. Ogni canarino è un uccello.

$$\forall X\,(\texttt{pinguino}(X) \rightarrow \texttt{uccello}(X)) \tag{5}$$

$$\forall X\,(\texttt{canarino}(X) \rightarrow \texttt{uccello}(X)) \tag{6}$$

Ogni pinguino è anormale, metti il tetto alla gabbietta per un uccello che vola, tweety è un uccello.

$$\forall X\,(\texttt{pinguino}(X) \rightarrow \texttt{ab}(X)) \tag{7}$$

$$\forall X\,(\texttt{uccello}(X) \wedge \texttt{vola}(X) \rightarrow \texttt{metti_tetto_gabbia}(X)) \tag{8}$$

$$\texttt{uccello}(\texttt{tweety}) \tag{9}$$

$$(4), (5), (6), (7), (8), (9), \texttt{CWA} \models \texttt{metti_tetto_gabbia}(\texttt{tweety})$$

Introduzione

La perdita della monotonia: i pinguini

Ogni uccello, che non sia anormale, vola (per $\leftarrow$, pensate alle api)

$$\forall X\,(\texttt{uccello}(X) \wedge \neg\texttt{ab}(X) \rightarrow \texttt{vola}(X)) \tag{4}$$

Ogni pinguino è un uccello. Ogni canarino è un uccello.

$$\forall X\,(\texttt{pinguino}(X) \rightarrow \texttt{uccello}(X)) \tag{5}$$

$$\forall X\,(\texttt{canarino}(X) \rightarrow \texttt{uccello}(X)) \tag{6}$$

Ogni pinguino è anormale, metti il tetto alla gabbietta per un uccello che vola, tweety è un uccello.

$$\forall X\,(\texttt{pinguino}(X) \rightarrow \texttt{ab}(X)) \tag{7}$$

$$\forall X\,(\texttt{uccello}(X) \wedge \texttt{vola}(X) \rightarrow \texttt{metti_tetto_gabbia}(X)) \tag{8}$$

$$\texttt{uccello}(\texttt{tweety}) \tag{9}$$

$$(4), (5), (6), (7), (8), (9), \texttt{CWA} \models \texttt{metti_tetto_gabbia}(\texttt{tweety})$$

Introduzione

La perdita della monotonia: i pinguini

Ogni uccello, che non sia anormale, vola (per $\leftarrow$, pensate alle api)

$$\forall X\,(\texttt{uccello}(X) \wedge \neg\texttt{ab}(X) \rightarrow \texttt{vola}(X)) \tag{4}$$

Ogni pinguino è un uccello. Ogni canarino è un uccello.

$$\forall X\,(\texttt{pinguino}(X) \rightarrow \texttt{uccello}(X)) \tag{5}$$

$$\forall X\,(\texttt{canarino}(X) \rightarrow \texttt{uccello}(X)) \tag{6}$$

Ogni pinguino è anormale, metti il tetto alla gabbietta per un uccello che vola, tweety è un uccello.

$$\forall X\,(\texttt{pinguino}(X) \rightarrow \texttt{ab}(X)) \tag{7}$$

$$\forall X\,(\texttt{uccello}(X) \wedge \texttt{vola}(X) \rightarrow \texttt{metti_tetto_gabbia}(X)) \tag{8}$$

$$\texttt{uccello}(\texttt{tweety}) \tag{9}$$

$$(4), (5), (6), (7), (8), (9), \texttt{CWA} \models \texttt{metti_tetto_gabbia}(\texttt{tweety})$$

Introduzione

La perdita della monotonia: i pinguini

Se ora so che tweety è un canarino.

$$\texttt{canarino(tweety)} \tag{10}$$

vale ancora che

$$(4)\text{–}(10), \texttt{CWA} \models \texttt{metti_tetto_gabbia(tweety)}$$

Ragioniamo con pingu. Se so che pingu è un uccello:

$$\texttt{uccello(pingu)} \tag{11}$$

vale che

$$(4)\text{–}(11), \texttt{CWA} \models \texttt{metti_tetto_gabbia(pingu)}$$

Ma se scoprissi ora che

$$\texttt{pinguino(pingu)} \tag{12}$$

$$(4)\text{–}(12), \texttt{CWA} \not\models \texttt{metti_tetto_gabbia(pingu)}$$

Introduzione

La perdita della monotonia: i pinguini

Se ora so che tweety è un canarino.

$$\texttt{canarino(tweety)} \tag{10}$$

vale ancora che

$$(4)\text{–}(10), \texttt{CWA} \models \texttt{metti_tetto_gabbia(tweety)}$$

Ragioniamo con pingu. Se so che pingu è un uccello:

$$\texttt{uccello(pingu)} \tag{11}$$

vale che

$$(4)\text{–}(11), \texttt{CWA} \models \texttt{metti_tetto_gabbia(pingu)}$$

Ma se scoprissi ora che

$$\texttt{pinguino(pingu)} \tag{12}$$

$$(4)\text{–}(12), \texttt{CWA} \not\models \texttt{metti_tetto_gabbia(pingu)}$$

KNOWLEDGE REPRESENTATION AND REASONING
IL SILLOGISMO
LA PERDITA DELLA MONOTONIA
PROGRAMMI DEFINITI
RICHIAMI
SEMANTICA
GROUNDING
PROGRAMMI GENERALI
DEFINIZIONI
LA PERDITA DELLA MONOTONIA
LA SEMANTICA

Abbiamo perso la monotonia.

Dove?

Il punto critico è questo:

$$\forall X\,(\texttt{uccello}(X) \wedge \neg\texttt{ab}(X) \rightarrow \texttt{vola}(X)) \qquad (13)$$

Non è una clausola di Horn

Vedremo anche come lo studio dei frammenti di logica predicativa adatti alla KR e NMR ha portato dei risultati molto eleganti della teoria della complessità computazionale.

Tali risultati sono anche UTILI, nel senso che avremo dei linguaggi per diverse classi di complessità (tra cui NP) corredati di risolutori con efficienza al massimo delle conoscenze attuali.

KNOWLEDGE REPRESENTATION AND REASONING
IL SILLOGISMO
LA PERDITA DELLA MONOTONIA
PROGRAMMI DEFINITI
RICHIAMI
SEMANTICA
GROUNDING
PROGRAMMI GENERALI
DEFINIZIONI
LA PERDITA DELLA MONOTONIA
LA SEMANTICA

Abbiamo perso la monotonia.
Dove?

Il punto critico è questo:

$$\forall X\,(\texttt{uccello}(X) \wedge \neg\texttt{ab}(X) \rightarrow \texttt{vola}(X)) \qquad (13)$$

Non è una clausola di Horn

Vedremo anche come lo studio dei frammenti di logica predicativa adatti alla KR e NMR ha portato dei risultati molto eleganti della teoria della complessità computazionale.

Tali risultati sono anche UTILI, nel senso che avremo dei linguaggi per diverse classi di complessità (tra cui NP) corredati di risolutori con efficienza al massimo delle conoscenze attuali.

KNOWLEDGE REPRESENTATION AND REASONING
IL SILLOGISMO
LA PERDITA DELLA MONOTONIA
PROGRAMMI DEFINITI
RICHIAMI
SEMANTICA
GROUNDING
PROGRAMMI GENERALI
DEFINIZIONI
LA PERDITA DELLA MONOTONIA
LA SEMANTICA

Abbiamo perso la monotonia.
Dove?
Il punto critico è questo:

$$\forall X\,(\texttt{uccello}(X) \wedge \neg\texttt{ab}(X) \rightarrow \texttt{vola}(X)) \qquad (13)$$

Non è una clausola di Horn
Vedremo anche come lo studio dei frammenti di logica predicativa adatti alla KR e NMR ha portato dei risultati molto eleganti della teoria della complessità computazionale.
Tali risultati sono anche UTILI, nel senso che avremo dei linguaggi per diverse classi di complessità (tra cui NP) corredati di risolutori con efficienza al massimo delle conoscenze attuali.

KNOWLEDGE REPRESENTATION AND REASONING
IL SILLOGISMO
LA PERDITA DELLA MONOTONIA
PROGRAMMI DEFINITI
RICHIAMI
SEMANTICA
GROUNDING
PROGRAMMI GENERALI
DEFINIZIONI
LA PERDITA DELLA MONOTONIA
LA SEMANTICA

Abbiamo perso la monotonia.
Dove?
Il punto critico è questo:

$$\forall X\,(\texttt{uccello}(X) \wedge \neg\texttt{ab}(X) \rightarrow \texttt{vola}(X)) \qquad (13)$$

Non è una clausola di Horn

Vedremo anche come lo studio dei frammenti di logica predicativa adatti alla KR e NMR ha portato dei risultati molto eleganti della teoria della complessità computazionale.

Tali risultati sono anche UTILI, nel senso che avremo dei linguaggi per diverse classi di complessità (tra cui NP) corredati di risolutori con efficienza al massimo delle conoscenze attuali.

KNOWLEDGE REPRESENTATION AND REASONING
IL SILLOGISMO
LA PERDITA DELLA MONOTONIA
PROGRAMMI DEFINITI
RICHIAMI
SEMANTICA
GROUNDING
PROGRAMMI GENERALI
DEFINIZIONI
LA PERDITA DELLA MONOTONIA
LA SEMANTICA

Abbiamo perso la monotonia.
Dove?
Il punto critico è questo:

$$\forall X\,(\texttt{uccello}(X) \wedge \neg\texttt{ab}(X) \rightarrow \texttt{vola}(X)) \qquad (13)$$

Non è una clausola di Horn
Vedremo anche come lo studio dei frammenti di logica predicativa adatti alla KR e NMR ha portato dei risultati molto eleganti della teoria della complessità computazionale.
Tali risultati sono anche UTILI, nel senso che avremo dei linguaggi per diverse classi di complessità (tra cui NP) corredati di risolutori con efficienza al massimo delle conoscenze attuali.

KNOWLEDGE REPRESENTATION AND REASONING
IL SILLOGISMO
LA PERDITA DELLA MONOTONIA
PROGRAMMI DEFINITI
RICHIAMI
SEMANTICA
GROUNDING
PROGRAMMI GENERALI
DEFINIZIONI
LA PERDITA DELLA MONOTONIA
LA SEMANTICA

INTRODUZIONE

LA RICERCA DELLA PERDITA DELLA MONOTONIA

Abbiamo perso la monotonia.
Dove?
Il punto critico è questo:

$$\forall X\,(\texttt{uccello}(X) \wedge \neg\texttt{ab}(X) \rightarrow \texttt{vola}(X)) \qquad (13)$$

Non è una clausola di Horn
Vedremo anche come lo studio dei frammenti di logica predicativa adatti alla KR e NMR ha portato dei risultati molto eleganti della teoria della complessità computazionale.
Tali risultati sono anche UTILI, nel senso che avremo dei linguaggi per diverse classi di complessità (tra cui NP) corredati di risolutori con efficienza al massimo delle conoscenze attuali.

Programmi Definiti

Un programma definito P è un insieme di clausole (di Horn) con esattamente un letterale positivo

$$A :- \ B_1, \ldots, B_n$$

$n \geq 0$ e $A, B_1, \ldots, B_n$ sono formule atomiche (atomi).

Programmi Definiti

- Dato P possiamo definire l'universo di Herbrand H_P (insieme dei termini ground) e la base di Herbrand B_P (insieme degli atomi ground).

$$P = \left\{ \begin{array}{ccc} q(a). & & \\ q(b). & & \\ r(b). & & \\ r(c). & & \\ p(X,Y) & :- & q(X), r(Y). \end{array} \right\}$$

- $H_P = \{a, b, c\}$
- $B_P = \{q(a), q(b), q(c), r(a), r(b), r(c),$
  $p(a,a), p(a,b), p(a,c), p(b,a), p(b,b), p(b,c),$
  $p(c,a), p(c,b), p(c,c)\}$.

Programmi Definiti

- Dato P possiamo definire l'universo di Herbrand H_P (insieme dei termini ground) e la base di Herbrand B_P (insieme degli atomi ground).

$$P = \left\{ \begin{array}{ccc} q(a). & & \\ q(b). & & \\ r(b). & & \\ r(c). & & \\ p(X,Y) & :- & q(X), r(Y). \end{array} \right\}$$

- $H_P = \{a, b, c\}$
- $B_P = \{q(a), q(b), q(c), r(a), r(b), r(c), p(a,a), p(a,b), p(a,c), p(b,a), p(b,b), p(b,c), p(c,a), p(c,b), p(c,c)\}$.

Programmi Definiti

- Dato P possiamo definire l'universo di Herbrand H_P (insieme dei termini ground) e la base di Herbrand B_P (insieme degli atomi ground).

$$P = \left\{ \begin{array}{ccc} q(a). & & \\ q(b). & & \\ r(b). & & \\ r(c). & & \\ p(X,Y) & :- & q(X), r(Y). \end{array} \right\}$$

- $H_P = \{a, b, c\}$
- $B_P = \{q(a), q(b), q(c), r(a), r(b), r(c), p(a,a), p(a,b), p(a,c), p(b,a), p(b,b), p(b,c), p(c,a), p(c,b), p(c,c)\}$.

Programmi Definiti

- Hanno una semantica *logica* di unico modello minimo: Dato P esiste M_P minimo modello di Herbrand.
- $P = \{q(a).\, q(b).\, r(b).\, r(c).\, p(X, Y) :- \; q(X), r(Y).\}$, $M_P = \{q(a), q(b), r(b), r(c), p(a, b), p(a, c), p(b, b), p(b, c)\}$.
- Ogni atomo $A \in B_P$ in M_P è conseguenza logica di P, ($P \models A$, ovvero A vale in tutti i modelli di P ovvero $P \cup \{\neg A\}$ è insoddisfacibile).
- Hanno una semantica *denotazionale* di minimo punto fisso mediante l'operatore $T_P : \wp(B_P) \longrightarrow \wp(B_P)$:

$$T_P(I) \;=\; \{a : \; a \leftarrow b_1, \ldots, b_m \in ground(P) \quad b_1 \in I, \ldots, b_m \in I\}$$

- Hanno una semantica *operazionale* basata sulla SLD-risoluzione.

Programmi Definiti

- Hanno una semantica *logica* di unico modello minimo: Dato P esiste M_P minimo modello di Herbrand.
- $P = \{q(a).\, q(b).\, r(b).\, r(c).\, p(X, Y) :- \; q(X), r(Y).\}$, $M_P = \{q(a), q(b), r(b), r(c), p(a, b), p(a, c), p(b, b), p(b, c)\}$.
- Ogni atomo $A \in B_P$ in M_P è conseguenza logica di P, ($P \models A$, ovvero A vale in tutti i modelli di P ovvero $P \cup \{\neg A\}$ è insoddisfacibile).
- Hanno una semantica *denotazionale* di minimo punto fisso mediante l'operatore $T_P : \wp(B_P) \longrightarrow \wp(B_P)$:

$$T_P(I) \;=\; \{a : \; a \leftarrow b_1, \ldots, b_m \in ground(P) \quad b_1 \in I, \ldots, b_m \in I\}$$

- Hanno una semantica *operazionale* basata sulla SLD-risoluzione.

Programmi Definiti

- Hanno una semantica *logica* di unico modello minimo: Dato P esiste M_P minimo modello di Herbrand.
- $P = \{q(a).\, q(b).\, r(b).\, r(c).\, p(X, Y) :- \; q(X), r(Y).\}$, $M_P = \{q(a), q(b), r(b), r(c), p(a, b), p(a, c), p(b, b), p(b, c)\}$.
- Ogni atomo $A \in B_P$ in M_P è conseguenza logica di P, ($P \models A$, ovvero A vale in tutti i modelli di P ovvero $P \cup \{\neg A\}$ è insoddisfacibile).
- Hanno una semantica *denotazionale* di minimo punto fisso mediante l'operatore $T_P : \wp(B_P) \longrightarrow \wp(B_P)$:

$$T_P(I) \;=\; \{a : \; a \leftarrow b_1, \ldots, b_m \in ground(P) \quad b_1 \in I, \ldots, b_m \in I\}$$

- Hanno una semantica *operazionale* basata sulla SLD-risoluzione.

Programmi definiti

- Vale che: esiste SLD derivazione di successo per A se e solo se $A \in M_P = T_P \uparrow \omega(\emptyset)$ (equivalenza tra le semantiche)
- In altri termini, tutto ciò che sta in M_P è quello che intendiamo essere vero, il resto lo intendiamo falso.
- Sono Turing Completi (se si usa almeno un simbolo di funzione e uno di costante — e.g. 0 e $s/1$)
- Altrimenti (solo simboli di costante) diventano decidibili e sono un formalismo "povero" per il calcolo (DB deduttivi).
- Sono poco indicati per rappresentare la conoscenza (anche nel caso di presenza di simboli di funzione).

Grounding

- Le varie semantiche sono fondate sulla nozione di programma ground
- Dato un programma generale P, il corrispondente programma ground $ground(P)$ è l'insieme di tutte le istanze ground su $\mathcal{H}_P$ delle clausole di P.

Knowledge Representation and Reasoning
Il sillogismo
La perdita della monotonia
Programmi Definiti
Richiami
Semantica
Grounding
Programmi Generali
Definizioni
La perdita della monotonia
La semantica

Grounding

$P =$

```
q(a).  q(b).  r(b).  r(c).
p(X,Y) :- q(X),r(Y).
```

$ground(P) =$

```
q(a).  q(b).  r(b).  r(c).
p(a,a) :- q(a),r(a).
p(a,b) :- q(a),r(b).
p(a,c) :- q(a),r(c).
p(b,a) :- q(b),r(a).
p(b,b) :- q(b),r(b).
p(b,c) :- q(b),r(c).
p(c,a) :- q(c),r(a).
p(c,b) :- q(c),r(b).
p(c,c) :- q(c),r(c).
```

Grounding

- Quant'è costoso il grounding?
- Assumiamo che H_P sia finito (e sia $|H_P| = c$).
- Siano $c_1, \ldots, c_n$ le clausole di P ($|P| = n$), e siano, rispettivamente, $\alpha_1, \ldots, \alpha_n$ i numeri delle variabili occorrenti in esse.
- Allora

$$|ground(P)| = \sum_{i=1}^{n} c^{\alpha_i}$$

- Se $k = \max_i\{\alpha_i\}$, abbiamo che

$$|ground(P)| \leq nc^k$$

- c può essere definito implicitamente (p.es. $p(1..100)$). In questi casi i numeri si sentono.

Grounding

- Quant'è costoso il grounding?
- Assumiamo che H_P sia finito (e sia $|H_P| = c$).
- Siano $c_1, \ldots, c_n$ le clausole di P ($|P| = n$), e siano, rispettivamente, $\alpha_1, \ldots, \alpha_n$ i numeri delle variabili occorrenti in esse.
- Allora

$$|ground(P)| = \sum_{i=1}^{n} c^{\alpha_i}$$

- Se $k = \max_i\{\alpha_i\}$, abbiamo che

$$|ground(P)| \leq nc^k$$

- c può essere definito implicitamente (p.es. $p(1..100)$). In questi casi i numeri si sentono.

Grounding

- Quant'è costoso il grounding?
- Assumiamo che H_P sia finito (e sia $|H_P| = c$).
- Siano $c_1, \ldots, c_n$ le clausole di P ($|P| = n$), e siano, rispettivamente, $\alpha_1, \ldots, \alpha_n$ i numeri delle variabili occorrenti in esse.
- Allora

$$|ground(P)| = \sum_{i=1}^{n} c^{\alpha_i}$$

- Se $k = \max_i\{\alpha_i\}$, abbiamo che

$$|ground(P)| \leq nc^k$$

- c può essere definito implicitamente (p.es. $p(1..100)$). In questi casi i numeri si sentono.

Grounding

- Quant'è costoso il grounding?
- Assumiamo che H_P sia finito (e sia $|H_P| = c$).
- Siano $c_1, \ldots, c_n$ le clausole di P ($|P| = n$), e siano, rispettivamente, $\alpha_1, \ldots, \alpha_n$ i numeri delle variabili occorrenti in esse.
- Allora

$$|ground(P)| = \sum_{i=1}^{n} c^{\alpha_i}$$

- Se $k = \max_i\{\alpha_i\}$, abbiamo che

$$|ground(P)| \leq nc^k$$

- c può essere definito implicitamente (p.es. $p(1..100)$). In questi casi i numeri si sentono.

Programmi Generali

Introduzione

```
studente(mark). studente(bill).
sposato(joe). sposato(mark). sposato(bob).
```

Definiamo il predicato che caratterizza le persone che sono studenti non sposati.

```
studente_single(X) :-
      studente(X),
      not sposato(X).
```

Programmi Generali

Introduzione

- La presenza del `not` nel corpo di una clausola ci fa uscire dall'insieme dei programmi definiti/clausole di Horn.
- Ad esempio,

```
studente_single(X) :-
      studente(X),
      not sposato(X).
```

equivale a:

$$\texttt{studente_single(X)} \vee \neg\texttt{studente(X)} \vee \texttt{sposato(X)}$$

che non è una clausola di Horn.

Programmi Generali

Definizione

- Un *programma generale* è un programma in cui sono ammessi letterali negativi nel corpo delle clausole.
- Ovvero le regole sono del tipo

$$H \leftarrow B_1, \ldots, B_m, \texttt{not}\ \ C_1, \ldots, \texttt{not}\ \ C_n$$

  con H, B_i, C_i atomi.
- Un *goal generale* è un goal in cui sono ammessi letterali negativi.
- Le clausole non sono più clausole di Horn:

$$p(a) \leftarrow \neg q(a)$$

  è equivalente a $p(a) \vee q(a)$.
- Un programma comunque ammette sempre un modello: l'insieme di tutti gli atomi ground $\mathcal{B}_P$, soddisfacendo tutte le teste, è un modello.

Programmi Generali

Non Monotonia

- L'intersezione di modelli non è necessariamente un modello. Ad esempio, $p(a) \leftarrow \texttt{not}\ \ q(a)$ ammette (anche) i due modelli $\{p(a)\}$ e $\{q(a)\}$; tuttavia la loro intersezione non è un modello.
- Può non esistere un unico modello minimo; in generale possono esistere più modelli minimali (si veda sopra)

Programmi Generali

Non Monotonia

- Si può definire anche per i programmi generali un operatore T_P. Il modo naturale per fare ciò è:

$$
\begin{array}{rcl}
T_P(I) & = & \{a : \; a \leftarrow b_1, \ldots, b_m, \texttt{not}\ c_1, \ldots, \texttt{not}\ c_n \in ground(P) \\
 & & \quad b_1 \in I, \ldots, b_m \in I, \\
 & & \quad c_1 \notin I, \ldots, c_n \notin I\}
\end{array}
$$

Tuttavia, $T_P(\emptyset) = \{p(a)\}$ e $T_P(\{q(a)\}) = \emptyset$:
T_P non è monotono (né continuo).
Non possiamo sfruttare il Teorema di punto fisso (Tarski) come per i programmi definiti.

Programmi Generali

Negazione classica o altro?

- La SLD-risoluzione non è adeguata.
- Si consideri il programma:

```
p(a) :- p(b). %%%% p(a) \/ not p(b)
p(a) :- not p(b). %%%% p(a) \/ p(b)
p(b) :- p(a). %%%% not p(a) \/ p(b)
```

- ed il goal

```
:- p(a), p(b) %%%% not p(a) \/ not p(b)
```

- L'insieme di queste quattro clausole è insoddisfacibile
- $P \cup \{\neg\varphi\}$ è insodd. sse $P \models \varphi$ ($\varphi = p(a) \wedge p(b)$)
- SLD derivazione però non è in grado di dedurre.
- Prolog entra in loop (capiremo).
- Potremmo usare risoluzione, ma è quello che vogliamo?

Programmi Generali

Negazione classica o altro?

- Quando scriviamo $p(a)$: $-$`not` $p(b)$, intendiamo davvero $p(a) \vee p(b)$?
- E' dunque lo stesso che scrivere $p(b)$: $-$`not` $p(a)$?
- Dobbiamo chiarire quali sono le conseguenze che vogliamo veramente ottenere o quantomeno ci aspettiamo da una regola generale.
- In realtà in KR sono due concetti diversi. Si parla di negazione classica ($\neg$) e di default negation (`not`), anche detta negation as failure.

Programmi Generali

Negazione classica o altro?

- Quando scriviamo $p(a) : -\texttt{not}\ \ p(b)$, intendiamo davvero $p(a) \vee p(b)$?
- E' dunque lo stesso che scrivere $p(b) : -\texttt{not}\ \ p(a)$?
- Dobbiamo chiarire quali sono le conseguenze che vogliamo veramente ottenere o quantomeno ci aspettiamo da una regola generale.
- In realtà in KR sono due concetti diversi. Si parla di negazione classica ($\neg$) e di default negation (`not`), anche detta negation as failure.