Fondamenti Logico Matematici dell'Informatica

UniShare

Davide Cozzi
@dlcgold

Indice

Capitolo 1

Introduzione

Questi appunti sono presi a lezione. Per quanto sia stata fatta una revisione è altamente probabile (praticamente certo) che possano contenere errori, sia di stampa che di vero e proprio contenuto. Per eventuali proposte di correzione effettuare una pull request. Link: `https://github.com/dlcgold/Appunti`.

Capitolo 2

Dimostrazioni = Algoritmi

Parliamo in primis del **paradigma dimostrazioni = algoritmi**.
Prendiamo come *linguaggio di specifica* un **linguaggio del prim'ordine con identità**.

Si riportano alcune definizioni utili in *logica matematica* tratte da Wikipedia:

Definizione 1. *Definiamo **linguaggio del primo ordine** come un linguaggio formale che serve per gestire meccanicamente enunciati e ragionamenti che coinvolgono i connettivi logici, le relazioni e i quantificatori $\forall$ e $\exists$.*
Si ha che "del primo ordine" ndica che c'è un insieme di riferimento e i quantificatori possano riguardare solo gli elementi di tale insieme e non i sottoinsiemi (posso dire "per tutti gli elementi" ma non "per tutti i sottoinsiemi").
Tale linguaggio è caratterizzato da:

- *un **alfabeto di simboli** per variabili, costanti, predicati, funzioni, connettivi, quantificatori o punteggiatura*
- *un **insieme di termini** per denotare gli elementi dell'insieme in analisi*
- *un **insieme di formule ben formate (FBF)** ovvero un insieme di stringhe composte di simboli dell'alfabeto che vengono considerate sintatticamente corrette*

Definizione 2. *Definiamo **sistema assiomatico** come un insieme di assiomi che possono essere usati per dimostrare teoremi. Una teoria matematica consiste quindi in una assiomatica e tutti i teoremi che ne derivano.*

Definizione 3. *Definiamo un sistema formale come una formalizzazione rigorosa e completa della nozione di sistema assiomatico costituito da:*

- *un alfabeto*
- *una grammatica che specifica quali sequenze finite dei simboli dell'alfabeto corrispondono ad una FBF. La grammatica deve essere ricorsiva, nel senso che deve esistere un algoritmo per decidere se una sequenza di simboli è o meno una formula ben formata*
- *un sottoinsieme delle FBF che sono gli assiomi. L'insieme degli assiomi è ricorsivo*
- *le regole di inferenze che associano formule ben formate ad n-uple di formule ben formate*

Definizione 4. *Definiamo gli* ***assiomi di Peano*** *come un gruppo di assiomi ideati al fine di definire assiomaticamente l'insieme dei numeri naturali:*

- *esiste un numero naturale: 0 (alternativamente 1 se si vuole escludere 0):*
$$0/1 \in \mathbb{N}$$
- *ogni naturale ha un naturale come successore. Ho quindi una funzione "successore" tale che:*
$$S : \mathbb{N} \to \mathbb{N}$$
- *numeri diversi hanno successori diversi, ovvero:*
$$x \neq y \implies S(x) \neq S(y)$$
- *0 (o alternativamente 1) non è il successore di alcun naturale, ovvero:*
$$S(x) \neq 0, \forall x \in \mathbb{N}$$
- *ogni sottoinsieme di numeri naturali che contenga lo zero e il successore di ogni proprio elemento coincide con l'intero insieme dei numeri naturali. Ovvero dato $U \subseteq \mathbb{N}$ tale che:*
  - $0 \in U$
  - $x \in U \implies S(x) \in U$

allora:

$$U = \mathbb{N}$$

Tale assioma è detto ***assioma dell'induzione*** *o* ***principio di induzione***

Definizione 5. *In una teoria del primo ordine si chiama* ***chiusura universale*** *di una formula ben formata $A(x_1, \ldots, x_n)$, con $x_1, \ldots, x_n$ variabili libere, la formula:*

$$\forall x_1 \forall x_2 \forall x_n \; A(x_1, \ldots, x_n)$$

ottenuta premettendo un quantificatore universale su ogni variabile libera.

Definizione 6. *Definiamo, in logica matematica,* ***aritmetica di Peano (PA)*** *come una teoria del primo ordine che ha come assiomi propri una versione degli* ***assiomi di Peano*** *espressi nel linguaggio del primo ordine. Si ha quindi che il linguaggio di PA è il linguaggio dell'aritmetica del primo ordine con i seguenti simboli:*

- *vari simboli per le variabili: x, y, z, x_1 etc...*
- *costanti individuali: 0 etc...*
- *simboli per funzioni unarie: S*
- *simboli per funzioni binarie $+$, $\times$ ($+(x, y)$ si indica anche con $x + y$ e analogamente si fa per $\times$)*
- *simboli per relazioni unarie: $=$*
- *simboli per connettivi logici, quantificatori e parentesi*

Gli assiomi di PA sono costituiti da:

- *gli assiomi logici*
- *gli assiomi per l'uguaglianza*
- *i seguenti assiomi propri (che "traducono" nella logica di Peano gli assiomi di Peano):*
  - $\forall x \neg(S(x) = 0)$
  - $\forall x \forall y (S(x) = S(y) \implies x = y)$
  - $\forall x (x + 0 = x)$

- $\forall x \forall y(x + S(y) = S(x + y))$
- $\forall x(x \times 0 = 0)$
- $\forall x \forall y(x \times S(y) = (x \times y) + x)$

Agli assiomi propri si aggiunge anche il seguente assioma proprio:

$$(\phi(0, x_1, \ldots, x_n) \wedge (\forall x(\phi(x, x_1, \ldots, x_n) \implies \phi(S(x), x_1, \ldots, x_n)) \implies \forall x \phi(x, x_1, \ldots, x_n)$$

per ogni FBF $\phi(x, x_1, \ldots, x_n)$ di cui $x, x_1, \ldots, x_n$ sono variabili libere. Questo è uno schema di assiomi detto ***schema di induzione*** *e si ha un assioma per ogni FBF ϕ*

Definizione 7. *Definiamo, in logica classica, il* ***principio del terzo escluso*** *che stabilisce che una proposizione e la sua negazione hanno valore opposto, non avendo una "terza opzione". In logica classica è una* ***tautologia****.*

Definizione 8. *Un termine è un* ***termine chiuso*** *sse non contiene delle variabili individuali.*

Definizione 9. *Una* ***formula chiusa*** *è una formula costruita nel linguaggio dei predicati in cui o non compaiono variabili o tutte le variabili presenti sono vincolate a un quantificatore e sono dunque variabili legate.*

Esempio 1. *Vediamo qualche esempio:*

$$\forall x, y \in \mathbb{N},\ \exists z \in \mathbb{N}\ t.c.\ mcd(x, y, z)$$

ovvero z è l'mcd di x e y.
Un altro esempio:

$$\forall x \in \mathbb{N},\ \exists y \in \mathbb{N}\ t.c.\ fatt(x, y)$$

ovvero y è il fattoriale di z.

Formule come quelle dell'esempio possono essere lette come **specifiche del problema di trovare un algoritmo totalmente corretto** che calcoli il risultato di tale problema per ogni input valido. Questa lettura non è implicita nella logica classica, dove non è richiesto di stabilire come viene prodotto il risultato. Si ha quindi a che fare con una lettura di un problema algoritmico di interesse per un informatico.
Le **dimostrazioni** di questa tipologia di formule, nell'ambito dell'**aritmetica di Peano (*PA*)**, sono quindi interpretabili come gli algoritmi che calcolano le funzioni specificate.
Come *vantaggi* di questa "atteggiamento" si ha che:

- l'attenzione si concentra su costruire la dimostrazione, sui passi dimostrativi, e non sulla stesura del codice
- i passi elementari della dimostrazione sono automatici
- la correttezza della dimostrazione è verificabile in modo automatico
- l'estrazione/sintesi dell'algoritmo dalla dimostrazione è diretta. Una volta che si ha la dimostrazione corretta si può estrarre direttamente l'algoritmo. Tale algoritmo è totalmente corretto rispetto alla specifica

La difficoltà si trasferisce dall'ambito convenzionale della programmazione e codifica dell'algoritmo in se alla costruzione dimostrazione e dei passi dimostrativi.
Si hanno quindi anche degli *svantaggi*, abbastanza problematici:

- l'algoritmo ottenuto non è ottimale rispetto al problema. Rispetto a questo bisognerebbe capire come incorporare "più semantica" del problema da risolvere nella dimostrazione stessa
- il formalismo e il linguaggio delle dimostrazioni sono "lontani" da quelli usati usualmente nella pratica informatica

Non tutte le dimostrazioni sono direttamente interpretabili come algoritmi. Per vedere questa cosa prendiamo un esempio famoso di formula da dimostrare in analisi.

Esempio 2 (esempio di Troelstra)**.** *Esistono due numeri irrazionali n e m tali che n^m è razionale. In termini di formula del primo ordine si ha quindi:*

$$\exists n, m \in \{\mathbb{R}/\mathbb{Q}\} \ t.c. \ n^m \in \mathbb{Q}$$

Cerchiamo di capire se:

$$\sqrt{2}^{\sqrt{2}} \in \mathbb{Q} \ o \ \sqrt{2}^{\sqrt{2}} \notin \mathbb{Q}$$

Vediamo quindi i due casi (sono solo due per il principio del terzo escluso):

1. *assumo $\sqrt{2}^{\sqrt{2}} \in \mathbb{Q}$ e pongo $n = m = \sqrt{2}$ avendo trovato due numeri irrazionali n e m tali per cui $n^m \in \mathbb{Q}$*

2. assumo $\sqrt{2}^{\sqrt{2}} \notin \mathbb{Q}$ *e pongo* $n = \sqrt{2}^{\sqrt{2}}$ *e* $m = \sqrt{2}$*. Ne segue che:*

$$n^m = \left(\sqrt{2}^{\sqrt{2}}\right)^{\sqrt{2}} = (\sqrt{2})^2 = 2 \in \mathbb{Q}$$

e quindi ho due numeri irrazionali n *e* m *tali per cui* $n^m \in \mathbb{Q}$

Non possiamo essere soddisfatti di questa dimostrazione. Non veniamo a conoscenza, tramite la dimostrazione, che $\sqrt{2}^{\sqrt{2}}$ *sia o meno razionale. Non possiamo capirlo in quanto assumo il terzo escluso e quindi non so quale dei due casi sia valido, non abbiamo un "esiste" costruttivo (*$\exists n, m \in \{\mathbb{R}/\mathbb{Q}\}$*) in quanto non sappiamo se* $\sqrt{2}^{\sqrt{2}}$ *è razionale o meno. Nonostante ciò al dimostrazione sta perfettamente "in piedi" ma non esibisce* n *e* m *in quanto non determina se* $\sqrt{2}^{\sqrt{2}} \in \mathbb{Q}$*.*

Quanto successo nell'esempio di Troelstra non può succedere in un **sistema costruttivo**.

Definizione 10. *Definiamo* ***sistema costruttivo*** *un sistema dove si hanno come requisiti minimali:*

- $S \vdash A \vee B \implies S \vdash A$ *oppure* $S \vdash B$ *quindi se nel sistema dimostro* $A \vee B$ *allora nel sistema dimostro* A *o dimostro* B*, con* A *e* B *formule chiuse. Questa è la* ***disjunction property (DP)***
- $S \vdash \exists x A(x) \implies S \vdash A(t)$ *quindi se ho dimostrato un esistenziale allora deve esistere un termine chiuso* t *per cui dimostro* $A(t)$ *nel sistema. Questa è la* ***explicitly definibility property (EDP)****, detta anche* ***existence/witness property***

La logica classica quindi **non è un sistema costruttivo** perché in logica classica riesco sempre a dimostrare $A \vee \neg A$ mentre nell'esempio di Troelstra si nota come non si possa dimostrare né A né $\neg A$. Quindi da una dimostrazione classica di $A \vee \neg A$ io non tiro fuori una dimostrazione classica di A oppure una dimostrazione classica di $\neg A$ quindi non vale la DP. Inoltre non vale nemmeno la EDP, ho dimostrato l'esistenza di n e m (che sono termini chiusi) ma non ho trovato se vale la proprietà che siano irrazionali. La logica classica quindi non è una logica costruttiva.

2.1 Interpretazione BHK

Passiamo quindi ad una semantica informale che per ogni costante logica associa una condizione per la sua *costruibilità*. Questa è **l'interpretazione Brouwer-Heyting-Kreisel (*BHK*)**.

Definizione 11. *Presa una costruzione π per questa semantica proposizionale si ha che:*

- $\pi(A \wedge B) = \pi'(A)$ *e* $\pi''(B)$ *ovvero una costruzione di* $A \wedge B$ *e uguale ad un'altra costruzione di* A ***e*** *un'altra ancora di* B
- $\pi(A \vee B) = \pi'(A)$ *o* $\pi''(B)$ *ovvero una costruzione di* $A \vee B$ *e uguale ad un'altra costruzione di* A ***o*** *un'altra ancora di* B
- $\pi(A \implies B)$ *è una funzione (o un funzionale, ovvero un insieme di funzioni)* f *che associa ad ogni costruzione* $\pi'(A)$ *una costruzione* $\pi''(B)$ *tale che* $\pi'' = f(\pi')$*. Quindi* f *associa costruzioni di* A *a costruzioni di* B
- $\pi(\neg A)$ *è una costruzione* π' *di* $A \implies \bot$

Lato semantica predicativa si ha che, dato un dominio D per la variabile x:

- $\pi(\exists x A(x)) = \langle c, \pi' \rangle |\, c \in D$ *e* $\pi'(A(c))$ *quindi è uguale ad una coppia* $= \langle c, \pi' \rangle$ *tale che c appartiene al dominio e* π' *è una costruzione effettiva di* $A(c)$
- $\pi(\forall x A(x)) =$ *è una funzione* f *che associa ad ogni elemento* $c \in D$ *una costruzione* $\pi'(A(c))$ *tale che* $\pi' = f(c)$

Questa semantica "naive" ha alcune problematiche/aporie:

- la BHK non specifica la costruzione di una formula atomica
- la BHK non dimostra il falso infatti nella costruzione di $\neg$ associamo la costruzione del $\neg A$ a quella dell'implicazione, che è una costruzione che associa costruzioni di A e costruzioni di B e quindi nessuna costruzione potrebbe avere il falso (???)

Bisognerà quindi chiarire alcune restrizioni dell'interpretazione BHK.
Si hanno varie semantiche per il costruttivismo che hanno "precisato" la BHK:

- la semantica della**realizzabilità ricorsiva** di Kleene
- la semantica dell'**interpretazione dialettica** di Gödel
- la semantica delle **prove possibili** di Prawitz
- la semantica dei **problemi finiti** di Medvedev

Queste 4 semantiche sono coerenti con la BHK e quindi sono **semantiche del costruttivismo**.
Passiamo ora ad una definizione formale.

Definizione 12. *Definiamo come* ***sistema costruttivo*** *un sistema S:*

$$S = T + L$$

dove:

- *T è una teoria con assiomi di forma particolare*
- *L è una logica intuizionistica, che prendiamo come punto di partenza per il costruttivismo, con le sue estensioni*

Non sempre comunque date T e L si ha che S è un sistema costruttivo.

Un esempio di sistema costruttivo è dato dall'**aritmetica intuizionistica**, ovvero la PA interpretata all'interno della logica intuizionistica. Altri esempi sono le **teorie con assiomi di Harrop**, teorie con assiomi $\forall\exists$ con matrice positiva priva di quantificatori e con minimo modello di Herbrandt etc...
Le clausole di Horn usate i Prolog hanno un modello minimo di Herbrandt e hanno assiomi $\forall\exists$ con matrice positiva priva di quantificatori dove $\exists$ viene eliminato attraverso skolemizzazione e il $\forall$ è implicito nelle regole del programma in quanto tutte le X, Y etc... si intendono quantificati universalmente.
Quindi la parte assiomatica di una teoria non basta a rendere costruttivo il sistema anche se la logica è costruttiva. Tuttavia se si restringono le assiomatizzazioni con formule del primo ordine di tipo particolare si possono ottenere sistemi costruttivi in cui vale come minimo la *DP* e la *EDP*.

2.2 Deduzione naturale

Vediamo un accenno della **deduzione naturale** ovvero di un **calcolo diretto** (quindi differente dal calcolo indiretto dei tableaux). Nella deduzione naturale si ha per ogni connettivo una **regola di introduzione** i e una **regola di eliminazione** e. Ad esempio se ho A e B come premesse posso introdurre l'and con la regola d:i introduzione dell'and:

$$\frac{A \quad B}{A \wedge B} i\wedge$$

Se invece ho $A \wedge B$ come premessa posso usare la regola di eliminazione dell'and, producendo:

$$\frac{A \wedge B}{A} e\wedge \text{ oppure } \frac{A \wedge B}{B} e\wedge$$

Avendo quindi due regole di eliminazione per l'and.
Passiamo all'or. Ho due regole di introduzione:

$$\frac{A}{A \vee B} i\vee \text{ oppure } \frac{B}{A \vee B} i\vee$$

L'eliminazione della or è complessa e verrà trattata più avanti ma è della forma:

$$\frac{A \vee B \quad C \quad C}{C} e\vee$$

Dove si ha che se se ho $A \vee B$ come premessa e assumendo A ho C ma anche assumendo B ho C posso eliminare l'or e ottenere C.
Passiamo all'implicazione. Se ho come assunzione A e da B riesco a dimostrare B allora posso introdurre l'implicazione:

$$\frac{\overset{[A]}{\overset{\cdots}{B}}}{A \implies B} i \implies$$

Per l'eliminazione ho che, tramite **modus ponens**:

$$\frac{A \implies B \quad A}{B} e \implies$$

Abbiamo poi la **regola del falso** che dice che dal falso segue qualsiasi cosa:

$$\frac{\bot}{B} \bot$$

e la regola dell'eliminazione della negazione che dice che se assumo $\neg A$ e ottengo il falso significa che si è ottenuta una contraddizione e quindi elimino il $\neg$:

$$\frac{\overset{[\neg A]}{\bot}}{A} e\neg$$

Passiamo al $\forall$. Se ho dedotto $A(p)$ per p generico posso introdurre il $\forall$:

$$\frac{A(p)}{\forall x A(x)} i\forall$$

Se ho come assunzione $\forall x A(x)$ posso dedurre un qualsiasi $A(t)$:

$$\frac{\forall x A(x)}{A(t)} e\forall$$

Possiamo fare l'equivalente per l'$\exists$, dove se esiste $A(t)$ posso dedurre l'esistenza di un certo x per cui vale $A(x)$:

$$\frac{A(t)}{\exists x A(x)} i\exists$$

L'eliminazione dell'esiste è complessa e verrà trattata più avanti ma è della forma:

$$\frac{\exists x A(x) \qquad \overset{[A(p)]}{C}}{C} e\exists$$

Dove assumendo $\exists x A(x)$, assumendo $A(p)$ con p generico e riuscendo ad ottenere C da quest'ultima assunzione con una serie di restrizioni posso ottenere C eliminando $\exists$.
Una dimostrazione in deduzione naturale è modulare alle logiche si vogliono usare:

- nelle dimostrazioni in logica classica utilizzo tutte le regole della deduzione naturale appena introdotte
- nelle dimostrazioni in logica intuizionistica non devo usare la regola di eliminazione della negazione
- nelle dimostrazioni in logica minimale non devo usare la regola di eliminazione della negazione e nemmeno la regola che dal falso segue qualsiasi cosa

Possiamo quindi caratterizzare queste tre logiche e nelle ultime due, quella intuizionistica e quella minimale, si può dimostrare che valgono DP e EDP mentre non posso dire lo stesso per la logica classica. Si ha inoltre che:

$$\text{logica minimale} \subseteq \text{logica intuizionistica} \subseteq \text{logica classica}$$

Discorso diverso vale per le teorie, ovvero per i sistemi, dove l'assiomatizzazione può fare la differenza portando anche fuori dalla costruttività (se ad esempio assumo come assioma che $\forall x A(x) \vee \neg A(x)$ e gli aggiungo la logica intuizionistica ottengo la logica classica che non è costruttiva).
Una teoria specifica è l'aritmetica di Peano dove si hanno i seguenti assiomi:

- $\forall x \neg(S(x) = 0)$

- $\forall x \forall y(S(x) = S(y) \implies x = y)$
- $\forall x(x + 0 = x)$
- $\forall x \forall y(x + S(y) = S(x + y))$
- $\forall x(x \times 0 = 0)$
- $\forall x \forall y(x \times S(y) = (x \times y) + x)$

Dove si ha la regola d'identità che in realtà sono due, ovvero $id1$ e $id2$:

$$\frac{}{x = x} id1 \text{ e } \frac{x = y \quad A(x)}{A(y)} id2$$

Dove si ha anche il principio/regola d'induzione:

$$\frac{A(0) \quad \overset{[A(j)]}{\overset{\cdots}{A(S(j))}}}{A(t)} ind$$

Dove se dimostro $A(0)$ e assunto $A(j)$ dimostro il successore di $A(j)$ allora posso dedurre $A(t), \forall t$.

Esempio 3. *Vediamo ora un esempio con degli assiomi specifici, costruttivi:*

- $pari(0)$
- $\forall x(pari(x) \implies \neg pari(S(x)))$
- $\forall x(\neg pari(x) \implies pari(S(x)))$

Quindi si ha che se x è apri non lo è il successore e se x non è pari lo è il successore. Assumiamo inoltre che 0 sia pari.
Scritta così potrebbe essere riscritta "1:1" in Prolog.
Cerchiamo quindi din dimostrare che che:

$$\forall x(pari(x) \vee \neg pari(x))$$

che si può pensare si un terzo escluso e quindi valga sempre. Questa formula, a livello di specifica, va letta come una funzione "per ogni numero naturale costruisce $pari(x)$ o $\neg pari(x)$" quindi costruisce o la parte sinistra o la parte destra. Quindi la formula i può leggere come la specifica di algoritmo che per ogni naturale mi dice se vale la parte sinistra o la parte destra dell'or e quindi è un algoritmo di decisione effettivo che per ogni naturale ti dice se è pari o non è pari. Questa è un'interpretazione diversa da quella classica.

Posso quindi fare una dimostrazione per induzione.
Il ***caso base*** *è, chiamando pari p, assunto per assioma* $p(0)$ *e usando l'introduzione dell'or:*

$$\frac{p(0)}{p(0) \vee \neg p(0)} i\vee$$

Passo al ***caso passo****.*
Assumo per ipotesi induttiva $p(j) \vee \neg p(j)$ *e assumiamo* $\forall x(p(x) \vee \neg p(S(x)))$ *che è un altro degli assiomi. Procedo eliminando il* $\forall$*:*

$$\frac{\forall x(p(x) \vee \neg p(S(x)))}{p(j), p(j) \implies \neg p(S(j))} e\forall$$

procedo quindi eliminando l'implicazione:

$$\frac{p(j), p(j) \implies \neg p(S(j))}{\neg p(S(j))} e \implies$$

e continuo inserendo l'or:

$$\frac{\neg p(S(j))}{p(S(j)) \vee \neg p(S(j))} i\vee$$

Analogamente faccio per l'altro assioma $\forall x(\neg p(x) \vee p(S(x)))$*:*

$$\frac{\forall x(\neg p(S(x)) \vee p(S(x)))}{\neg p(j), \neg p(j) \implies p(S(j))} e\forall$$

$$\frac{\neg p(j), \neg p(j) \implies p(S(j))}{p(S(j))} e \implies$$

$$\frac{p(S(j))}{p(S(j)) \vee \neg p(S(j))} i\vee$$

Partendo quindi da $p(j) \vee \neg p(j)$ *posso fare l'eliminazione dell'or ottenendo il* ***caso passo****:*

$$\frac{p(j) \vee \neg p(j)}{p(S(j)) \vee \neg p(S(j))} e\vee$$

e quindi posso concludere il passo induttivo:

$$\frac{p(0) \vee \neg p(0) \quad p(S(j)) \vee \neg p(S(j))}{\forall x(p(x) \vee \neg p(x))} ind$$

concludendo al dimostrazione costruttiva.
Posso quindi dire che, essendo 0 pari, 1 è dispari e quindi 2 è pari, 3 dispari etc... in pratica è un ciclo che parte dal caso base e poi decide per qualsiasi numero naturale. Possiamo quindi estrarre un algoritmo iterativo (potrei anche estrarne uno ricorsivo) da questa dimostrazione. Tale algoritmo è visualizzabile nell'implementazione C nel listing 2.2.

Listing 1 Codice C dell'algoritmo di calcolo pari creato dalla dimostrazione

```c
#include <stdio.h>

void A1(int* fdv){
  *fdv = 0;
}
void A2(int* j, int*fdv){
  *fdv = 1;
  *j = *j + 1;
}
void A3(int* j, int*fdv){
  *fdv = 0;
  *j = *j + 1;
}
void base(int* f){
  A1(&*f);
}
void passo(int* j, int ff1, int* ff2){
  if (ff1 == 0){
    A2(&*j, &*ff2);
  }else{
    A3(&*j, &*ff2);
  }
}

int main(){
  int x, j, ff;
  scanf("%d", &x);
  j = 0;
  base(&ff);
  while(j <= x - 1){
    passo(&j, ff, &ff);
  }
  if (ff == 0){
    printf("%d è pari\n", x);
  }else{
    printf("%d è dispari\n", x);
  }
  return 0;
}
```

Come detto algoritmi così sintetizzati non sono algoritmi ottimali e l'esempio del pari o dispari è evidente. Normalmente si avrebbe infatti:

Listing 2 Codice C dell'algoritmo di calcolo pari ottimale

```c
#include <stdio.h>
int main(){
  int x;
  scanf("%d", &x);
  if ((x % 2) == 0){
    printf("%d è pari\n", x);
  }else{
    printf("%d è dispari\n", x);
  }
  return 0;
}
```

E quindi si ha un limite nella costruzione dell'algoritmo anche se il primo sappiamo essere corretto (avendo applicato la deduzione naturale e l'induzione, ho la garanzia che in quella assiomatizzazione il programma sia totalmente corretto) mentre di quest'ultimo dovremmo dimostrare la correttezza. Nel primo programma la componente funzionale della dimostrazione è data dalle varie funzioni che si richiamano. L'algoritmo per ogni istanza calcola se vale la parte sinistra o destra della disgiunzione in modo uniforme. Le funzioni si deducono automaticamente dalla prova costruttiva. Niente di tutto ciò è asseribile sull'algoritmo ottimo.
Parlando invece di Prolog avrei una situazione diversa. In Prolog ogni computazione è un'*istanza di dimostrazione* che varia di caso in caso e quindi non si ha una dimostrazione generale. Possiamo dire che un programma Prolog non rappresenta l'algoritmo che risolve il problema specificato per ogni dato di input. Si ha quindi un diverso modello di calcolo.