

DA MATITA A DEDUKTI E RITORNO

SOTTOTITOLO CHE NON SO ANCORA COSA

Mattia Girolimetto

Relazione per il corso 85001 - Metodi logici per la Filosofia
Alma Mater Studiorum, Università di Bologna

19 Luglio 2023

INDICE

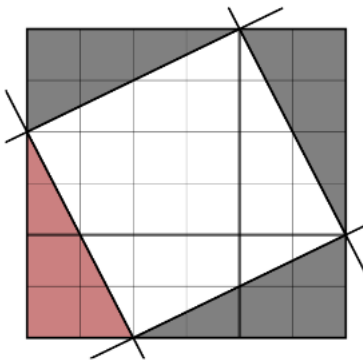
- 1 Dedukti e Matita 3**
 - 1.1 I proof assistant 4
 - 1.2 Dedukti 7
 - 1.3 Matita 9
- 2 Esportazione 11**
 - 2.1 Krajono 12
 - 2.2 La codifica 13
- 3 Importazione 14**
 - 3.1 Problemi 16
 - 3.2 Pragma 19
- 4 Conclusioni 20**

Dedukti e Matita

I PROOF ASSISTANT



matita



 Agda

ISOMORFISMO DI CURRY-HOWARD

LOGICA

Proposizioni

Dimostrazioni

Verifica di una dimostrazione

TEORIA DEI TIPI

Tipi

Programmi

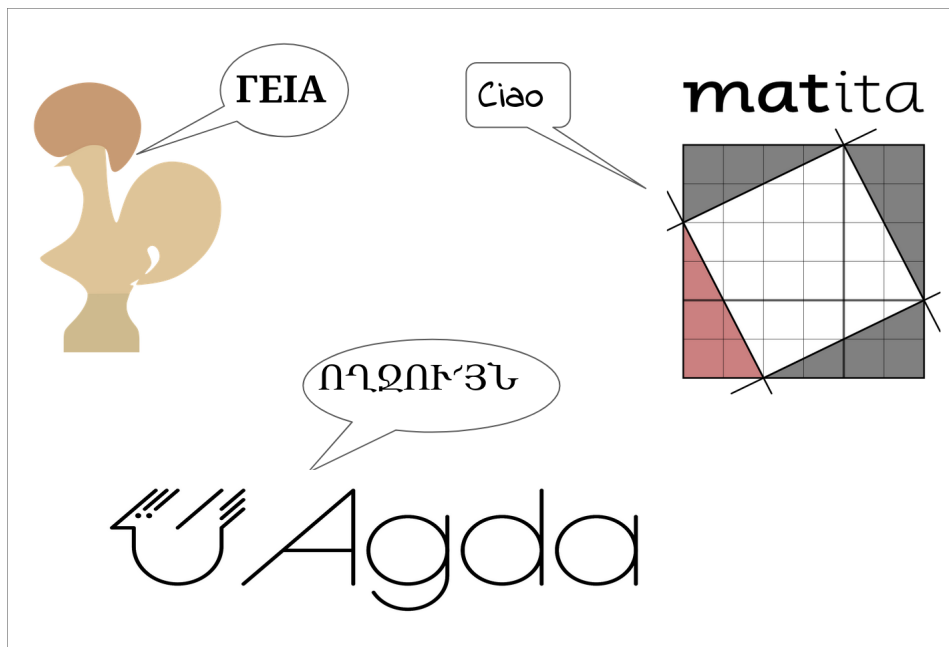
Verifica di tipo

\Leftrightarrow

\Leftrightarrow

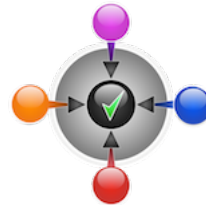
\Leftrightarrow

I PROOF ASSISTANT



DEDUKTI

- Framework logico
- Implementa logiche e teoremi
- Basato sul $\lambda\Pi$ -calcolo modulo



dedukti

$\lambda\Pi$ -CALCOLO MODULO

Estende il λ -calcolo *tipizzato* aggiungendo

- ▶ Tipi dipendenti
- ▶ Regole di riscrittura

$\lambda\Pi$ -CALCOLO MODULO

Estende il λ -calcolo *tipizzato* aggiungendo

- ▶ Tipi dipendenti
- ▶ Regole di riscrittura

Esempio (Regole di riscrittura)

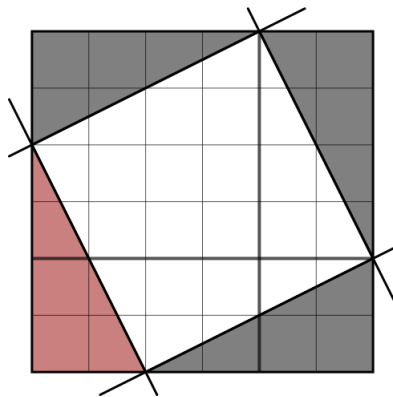
$$\text{sum } n \ 0 \hookrightarrow n$$

$$\text{prod } n \ 0 \hookrightarrow 0$$

MATITA

- ▶ Proof assistant sviluppato all'Università di Bologna
- ▶ Basato sul calcolo delle *costruzioni (co)induttive*

matita



CALCOLO DELLE COSTRUZIONI (CO)INDUTTIVE

Caratterizzato dalla possibilità di definire

- ▶ *tipi induttivi*
- ▶ *punto fissi*
- ▶ *pattern matching*

```
inductive nat: Type[0]  $\stackrel{\text{def}}{=}$   
  0 : nat  
  | S : nat → nat.
```

```
let rec plus n m on n  $\stackrel{\text{def}}{=}$   
  match n with  
  [ 0 ⇒ m  
  | S x ⇒ S (plus x m)  
  ].
```


Esportazione

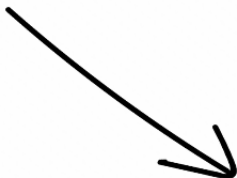
KRAJONO

- ▶ *Fork* di Matita
- ▶ Aggiunge l'esportazione verso Dedukti
- ▶ Non più supportato



LA CODIFICA

```
let rec plus n m on n   
match n with  
[ 0 => m  
| S x => S (plus x m) ].
```

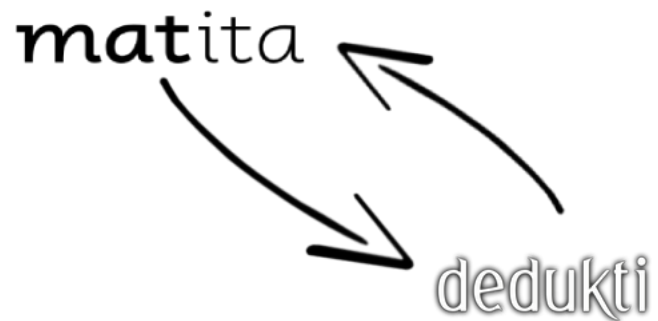


```
def plus :  
  cic.Term (cic.type cic.z)  
  (cic.prod (cic.type cic.z) (cic.type cic.z)  
   matita_esperimento_other.nat  
   (___ : cic.Term (cic.type cic.z) matita_esperimento_other.nat =>  
    cic.prod (cic.type cic.z) (cic.type cic.z)  
    matita_esperimento_other.nat  
    (___1 :  
     cic.Term (cic.type cic.z) matita_esperimento_other.nat =>  
      matita_esperimento_other.nat)))
```

```
def plus_body :  
  cic.Term (cic.type cic.z)  
  (cic.prod (cic.type cic.z) (cic.type cic.z)  
   matita_esperimento_other.nat  
   (___ : cic.Term (cic.type cic.z) matita_esperimento_other.nat =>  
    cic.prod (cic.type cic.z) (cic.type cic.z)  
    matita_esperimento_other.nat  
    (___1 :  
     cic.Term (cic.type cic.z) matita_esperimento_other.nat =>  
      matita_esperimento_other.nat)))
```

Importazione

IMPORTAZIONE



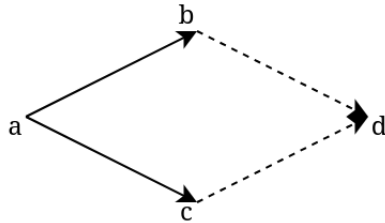
PROBLEMI

- **Problema** In Dedukti le proprietà di *confluenza* e *normalizzazione* non sono garantite.

CONFLUENZA E NORMALIZZAZIONE

Definizione (Confluenza)

Dato un termine a , se esistono due regole di riscrittura $a \hookrightarrow^* b$ e $a \hookrightarrow^* c$, allora esiste un termine d tale che $b \hookrightarrow^* d$ e $c \hookrightarrow^* d$.



Definizione (Normalizzazione)

Dato un termine, questo può essere ridotto al più un numero finito di volte.

PROBLEMI

- ▶ **Problema** In Dedukti le proprietà di *confluenza* e *normalizzazione* non sono garantite.
- ▶ **Problema** Durante la codifica vengono perse informazioni necessarie alla ricostruzione dei termini originali.

PRAGMA

Per preservare tali informazioni vengono usate le *pragma*:

```
1 #PRAMGA BEGIN INDUCTIVE NAME=nat LEFTNO=0 CONS:nat=0 CONS:nat=S.  
2  
3 nat : cic.Univ (cic.type cic.z).  
4  
5 O : cic.Term (cic.type cic.z) matita_test_nat.nat.  
6  
7 S : cic.Term (cic.type cic.z) (cic.prod (cic.type cic.z) (cic.type cic.z)  
8     matita_test_nat.nat(__ : cic.Term (cic.type cic.z)  
9     matita_esperimento_nat.nat => matita_test_nat.nat)).  
10  
11 #PRAMGA END INDUCTIVE.  
12
```

Conclusioni

CONCLUSIONI

- ▶ È possibile esportare i termini Matita nel linguaggio di Dedukti
- ▶ È possibile importare in Matita termini Dedukti
- ▶ È possibile reimportare in Matita termini Dedukti precedentemente esportati da Matita, ricostruendo (in parte) l'oggetto Matita originale

LAVORI FUTURI

- ▶ Aggiungere ricostruzione del costrutto di *pattern matching*
- ▶ Integrare le funzionalità di importazione/esportazione con la UI di Matita

LINK UTILI

- ▶ Matita: <https://github.com/sacredot/matita>
- ▶ Dedukti: <https://deducteam.github.io/>
- ▶ Krajono: <https://github.com/Deducteam/Krajono>
- ▶ Coq: <https://coq.inria.fr/>
- ▶ Agda: <https://wiki.portal.chalmers.se/agda/pmwiki.php>