Logica

Sintassi di Matita

Claudio Sacerdoti Coen

<sacerdot@cs.unibo.it>

Universitá di Bologna

16,18/10/2017

Outline

Sintassi

Sintassi dei termini di Matita

Termini: $t ::= x \mid c \mid t\ t \mid \lambda x : T.t \mid \ldots$

- x sono variabili (potete usare qualunque identificatore)
- c sono costanti (potete usare qualunque identificatore)
- $t\ t$ è l'applicazione di funzione (non servono le parentesi!)
- $\lambda x : T.t$ è una funzione anonima di input x (di tipo T) e output t
- l'utente può estendere la sintassi introducendo operatori infissi, prefissi, postfissi, n-ari a piacimento con precedenze/associatività a scelta
- altri operatori verranno introdotti in seguito

Sintassi

Esempi:

- "$f(x, g(y))$" (in notazione matematica) si scrive $f\ x\ (g\ y)$
- $f\ x + g\ y$ si legge $(f\ x) + (g\ y)$
  l'applicazione ha una precedenza alta
- $\lambda x : \mathbb{N}.x + x$ è la funzione che raddoppia x
  input: x; output: $x + x$
- $(\lambda x : \mathbb{N}.x + x)\ 3$ ha come risultato 6

Sintassi

Sintassi dei tipi di Matita

Tipi: $T ::= c \mid T \to T \mid \ldots$

- c sono tipi atomici (potete usare qualunque identificatore)
- $T_1 \to T_2$ è il tipo di una funzione con input T_1 e output T_2
- $\to$ associa a destra
- $T_1 \to T_2 \to T_3$ è una funzione che prende in input un T_1 e (restituisce una funzione che) prende in input un T_2 e restituisce un T_3
  potete pensarla come una funzione con due argomenti in input (ma è veramente una funzione che restituisce una funzione)
- altri costruttori di tipo verranno introdotti in seguito

Sintassi

Sintassi delle formule di Matita

Formule:

$$\begin{array}{rcl} F & ::= & p\ t_1\ \ldots\ t_n \mid \mathit{True} \mid \mathit{False} \mid F \wedge F \mid F \vee F \mid \neg F \\ & \mid & F \to F \mid F \iff F \mid \forall x : T.F \mid \exists x : T.F \end{array}$$

- i predicati $p\ t_1\ \ldots\ t_n$ sono definiti dall'utente
- gli operatori $\wedge, \vee, \to, \iff$ sono tutti associativi a sinistra
- precedenza: $\neg > \wedge > \vee > \to = \iff$
- i quantificatori $\forall, \exists$ si estendono fino alla fine della formula o fino alla prossima parentesi chiusa
- $\wedge$ è la "e" o congiunzione; $\vee$ la "o" o disgiunzione; $\to$ il "se ...allora" o implicazione; $\iff$ il "se e solo se" o coimplicazione; $\neg$ il "non" o negazione

Sintassi

Definizioni o dichiarazioni di tipo, predicato, etc.

Ci sono comandi per definire o dichiarare tipi, predicati, termini, funzioni, etc.
Vi verranno spiegati in seguito.
Ci sono comandi per definire nuove notazioni e simboli. Non vi verranno spiegati.
Matita ha un manuale in linea per chi fosse curioso.

Sintassi

Dichiarazioni di tipi, predicati e termini primitivi

- **axiom** *set* : *Type*. dichiara che *set* da ora è un nuovo tipo primitivo (= non definito)
- **axiom** *mem* : *set* $\rightarrow$ *set* $\rightarrow$ *Prop*. dichiara che *mem*, poi indicata con $\in$, da ora è un nuovo predicato binario primitivo (= non definito) su insiemi
- **axiom** *intersect* : *set* $\rightarrow$ *set* $\rightarrow$ *set*. dichiara che *intersect*, poi indicato con $\cap$, da ora è una nuova operazione binaria primitiva (= non definita) fra due insiemi
- ovvero dati due insiemi A e B sto dicendo che esiste un terzo insieme indicato con $A \cap B$: è la prima parte dell'assioma dell'intersezione (binaria)

Sintassi

Dichiarazioni di assiomi

- con **axiom** *nome* : F. posso dichiarare che una formula F vale come assioma
- a volte è meglio spezzare assiomi in assiomi più semplici
  1. **axiom** *ax_intersect1* :
     $\forall A, B. \forall Z. (Z \in A \cap B \rightarrow Z \in A \wedge Z \in B)$.
  2. **axiom** *ax_intersect2* :
     $\forall A, B. \forall Z. (Z \in A \wedge Z \in B \rightarrow Z \in A \cap B)$.

Sintassi

Teoremi

theorem *nome* : *F*.
. . . dimostrazione . . .
qed.

lemma e corollary possono essere usati al posto di theorem con la stessa sintassi

Esempio

theorem *intersect_empty* : $\forall A.A \cap \emptyset = \emptyset$.
. . . dimostrazione . . .
qed.

Linguaggio delle prove

Alcuni comandi di prova: regole di introduzione

Per dimostrare	si usa	dove
$\forall x : T.F$	**assume** $x : T$.	
$F_1 \to F_2$	**suppose** $F_1(H)$.	H è un nome per l'ipotesi
F	**by** $H_1, \ldots, H_n$ **done**	$H_1, \ldots, H_n$ sono ipotesi o teoremi da combinare

Teoremi di introduzione per $\wedge, \vee, \iff$

- $conj : \forall A, B.A \to B \to A \wedge B$
- $conj : \forall A, B.(A \to B) \to (B \to A) \to A \iff B$
- $or\_introl : \forall A, B.A \to A \vee B$
- $or\_intror : \forall A, B.B \to A \vee B$

Linguaggio delle prove

Alcuni comandi di prova: regole di eliminazione

Per usare	si procede con
$H : F_1 \wedge F_2$	**by** H **we have** F_1 (H_1) **and** F_2 (H_2).
$H : F_1 \vee F_2$	**we proceed by cases on** H **to prove** F. **case Left**. . . . prova di F sotto l'ipotesi $H' : F_1$. . . **case Right**. . . . prova di F sotto l'ipotesi $H' : F_2$. . .
$H : False$	**by** (**ABSURDUM** H) **done**.

Linguaggio delle prove

Altri comandi di prova

Il comando	serve a
we need to prove F.	ribadire ciò che stiamo provando
we need to prove F (H). . . . prova di F. . . . . . prova dove uso H. . .	provare F chiamando H il risultato per usarlo in seguito
by $H_1, \ldots, H_n$ **we proved** F (H).	concludere F combinando $H_1, \ldots, H_n$ per usarlo in seguito col nome H

Ulteriori comandi di prova

I rimanenti quattro comandi vi verranno descritti in seguito.

Linguaggio delle prove

Esempio:

theorem *union_empty* : $\forall A.\ A \cup \emptyset = A$.
assume A : *set*.
we need to prove $(\forall Z.\ Z \in A \cup \emptyset \iff Z \in A)\ (H)$.
assume Z : *set*.
we need to prove $(Z \in A \cup \emptyset \to Z \in A)\ (H1)$.
suppose $(Z \in A \cup \emptyset)\ (K)$.
by *ax_union*1, K **we proved** $(Z \in A \vee Z \in \emptyset)\ (K1)$.
we proceed by cases on $K1$ **to prove** $(Z \in A)$.
case *Left*.
by H
done.
case *Right*.
by *ax_empty*, H **we proved** *False* $(H2)$.
by $(ABSURDUM\ H2)$
done.
we need to prove $(Z \in A \to Z \in A \cup \emptyset)\ (H2)$.
suppose $(Z \in A)\ (K)$.
by K, *or_introl* **we proved** $(Z \in A \vee Z \in \emptyset)\ (K1)$.
by $K1$, *ax_union*2
done.
by $H1$, $H2$, *conj*
done.
by H, *ax_extensionality*
done.
qed.