Paweł Marczewski

Systemy wspomagania dowodzenia

jak wytłumaczyć matematykę komputerowi

Formalne dowody

P1. $\phi \rightarrow \phi$

P2. $\phi \rightarrow (\psi \rightarrow \phi)$

P3. $(\phi \rightarrow (\psi \rightarrow \xi)) \rightarrow ((\phi \rightarrow \psi) \rightarrow (\phi \rightarrow \xi))$

P4. $(\neg\phi \rightarrow \neg\psi) \rightarrow (\psi \rightarrow \phi)$

Zasady są bardzo proste...

aksjomaty systemu hilbertowskiego, Wikipedia

http://us.metamath.org,
Logicomix: An Epic Search for Truth

...ale dowody mogą być długie

Assertion

Ref	Expression
2p2e4	$\vdash (2 + 2) = 4$

Proof of Theorem 2p2e4

Step	Hyp	Ref	Expression
1		df-2 7230	$\ldots 3 \vdash 2 = (1 + 1)$
2	1	opreq2i 4953	$\ldots 2 \vdash (2 + 2) = (2 + (1 + 1))$
3		df-4 7232	$\ldots 3 \vdash 4 = (3 + 1)$
4		df-3 7231	$\ldots 4 \vdash 3 = (2 + 1)$
5	4	opreq1i 4952	$\ldots 3 \vdash (3 + 1) = ((2 + 1) + 1)$
6		2cn 7240	$\ldots 4 \vdash 2 \in \mathbb{C}$
7		ax1cn 6498	$\ldots 4 \vdash 1 \in \mathbb{C}$
8	6, 7, 7	addassi 6553	$\ldots 3 \vdash ((2 + 1) + 1) = (2 + (1 + 1))$
9	3, 5, 8	3eqtri 1947	$\ldots 2 \vdash 4 = (2 + (1 + 1))$
10	2, 9	eqtr4i 1946	$1 \vdash (2 + 2) = 4$

s)co 4944 1c1 6463 + caddc 6465 2c2 7221 3c3 7222 4c4 7223

2e4 7282 sqr2gt1lt2 8045 i4 8060 sin01bndlem1 8809 cos01bndlem2 8812 pilem1 10385 pcoass 16584

1 4 ax-2 5 ax-3 6 ax-mp 7 ax-7 1339 ax-gen 1340 ax-8 1341 ax-9 1342 ax-10 1343 ax-11 1344 ax-12 1345 ax-13 1346
x-6o 1359 ax-9o 1516 ax-10o 1535 ax-16 1615 ax-11o 1623 ax-ext 1900 ax-rep 3486 ax-sep 3496 ax-nul 3503 ax-pow 3539

185 df-or 262 df-an 263 df-3or 887 df-3an 888 df-ex 1362 df-sb 1571 df-eu 1810 df-mo 1811 df-clab 1907 df-cleq 1912
df-reu 2163 df-rab 2164 df-v 2348 df-sbc 2509 df-csb 2596 df-dif 2652 df-un 2655 df-in 2658 df-ss 2660 df-pss 2662 df-nul 2931
-tp 3108 df-op 3109 df-uni 3234 df-int 3271 df-iun 3313 df-br 3395 df-opab 3454 df-tr 3470 df-eprel 3641 df-id 3644 df-po 3649
df-on 3719 df-lim 3720 df-suc 3721 df-om 4008 df-xp 4058 df-rel 4059 df-cnv 4060 df-co 4061 df-dm 4062 df-rn 4063 df-res 4064

Idealne zadanie dla komputera

(demonstracja)

Potem...

- takie proste rzeczy potrafi "auto"
- ciągle trzeba komputerowi podpowiadać
- czytelniejsze dowody trudniej pisać

```
lemma prop_2_14_1_part: "⟦ c1 c2 MeetAt p; c isSumOf c1 c2; r isEndPoint c ⟧
  ⟹ ¬ (r incident c1 ∧ r incident c2)"
apply (subgoal_tac "c1 isPartOf c ∧ c2 isPartOf c")
 apply (rule notI)
 apply (subgoal_tac "c1 isPartOf c2 ∨ c2 isPartOf c1")
  apply (erule disjE)
   apply (drule corollary_2_9, simp)
  apply (drule MeetAt_sym, drule corollary_2_9, simp)
 apply (simp add: isEndPoint_def)
apply (simp add: isSumOf_def add: isPartOf_def)
done

lemma prop_2_14_1: "⟦ c1 c2 MeetAt p; c isSumOf c1 c2; isOpenCurve c ⟧
  ⟹ (∀ q. q ≠ p ⟶ ¬ (q incident c1 ∧ q incident c2))"
apply (rule allI, rule impI, rule notI)
apply (unfold isOpenCurve_def, erule exE)
apply (frule prop_2_14_1_part, assumption+)
apply (subgoal_tac "q ≠ pa ∧ pa ≠ p ∧ (q isEndPoint c1 ∧ q isEndPoint c2) ∧
  (p isEndPoint c1 ∧ p isEndPoint c2)")
 apply (erule conjE)+
 apply (subgoal_tac "pa isEndPoint c1 ∧ pa isEndPoint c2")
  apply (erule disjE)
  (* p incident c1 *)
   apply (drule_tac p="p" and q="q" and r="pa" in axiom_c5, assumption+, simp)
   apply (rule_tac c="c" in theorem_2_10)
    apply (simp add: isSumOf_def add: isPartOf_def)+
   apply (subgoal_tac "pa isEndPoint c2")
    apply (drule_tac p="p" and q="q" and r="pa" in axiom_c5, assumption+, simp)
   apply (rule_tac c="c" in theorem_2_10)
    apply (simp add: isSumOf_def add: isPartOf_def)+
 (* pa incident c1 ∨ pa incident c2 *)
 apply (simp add: isSumOf_def add: isEndPoint_def)
(* now for all the subgoals... *)
apply (rule conjI)
 (* q ≠ pa *)
 apply (rule notI)
 apply (subgoal_tac "pa incident c1 ∧ pa incident c2", simp, simp)
apply (rule conjI)
 (* pa ≠ p *)
 apply (rule notI)
 apply (subgoal_tac "p incident c1 ∧ p incident c2")
  apply (simp, simp add: MeetAt_def)
(* (q isEndPoint c1 ∧ q isEndPoint c2) ∧ (p isEndPoint c1 ∧ p isEndPoint c2) *)
apply (auto intro: theorem_2_10 simp add: isSumOf_def simp add: isPartOf_def simp add: MeetAt
```

Moje wrażenia

- niezła zabawa!
- trzeba dobrze rozumieć, co chcemy napisać
- trochę jak programowanie, ale nie do końca

Programming in Lisp is like playing with the primordial forces of the universe. It feels like lightning between your fingertips.

- tu można pod palcami czuć Prawdę :)

Glenn Ehrlich, http://cooking-with-lisp.blogspot.com/2004/05/my-road-to-lisp-answer.html

Trzeba dobrze rozumieć

I have this sort of statement: the difference between science and art is that science is something that we understand well enough to explain it to a computer, and art is everything else.

- Donald Knuth

- formalizując matematykę, można zrozumieć ją lepiej
- z drugiej strony: co z dowodami, które rozumie tylko komputer? (tw. o czterech kolorach)

Ciekawy pomysł - ProofPeer

- systemów i formalizmów jest wiele
- jak ułatwić innym korzystanie z naszej pracy?
- pomysł: wspólna platforma, "GitHub dla matematyków"

Inne zastosowanie: dowodzenie poprawności programów

- statyczne typy (`int f(int a, char b)`) usuwają całą klasę błędów
- dowodzenie pozwala to uogólnić
- jest nawet formalnie zweryfikowany kompilator C
- proof-carrying code: alternatywa dla podpisywania kodu

Dziękuję za uwagę

pytania?