sketching Lagrange

Freek Wiedijk

Brouwer seminar

Radboud University Nijmegen

2009 06 09, 15 : 45

a claim about formal mathematics

is formalization research?

claim:

writing a formalization is laborious, but trivial

formalization =

full formal proof needing just the axioms of the system

part 1 of the claim

turning informal text into formal proof sketch is trivial

formal proof sketch =
rendering of existing proof text in a formal language =
incomplete formalization

part 2 of the claim

turning formal proof sketch into full formalization is trivial

‘coloring in’ the formal proof sketch

‘coloring in’ $=$
adding intermediate steps $+$
adding references between steps $+$
adding references to lemmas from the library

additional claim

there is not a 'best way' to formalize a theorem

any informal presentation is formalizable

the story of Martijn, Dan, John, Henk and Tonny

an exercise for Dan

Martijn's PhD thesis (among many other things):
Coq formalization of Fermat's little theorem

$$a^{p-1} \equiv 1 \pmod{p}$$

my proposal to Dan:
more 'mathematical' formalization?

- formalize Lagrange's theorem in C-CoRN
- prove Fermat's little theorem from Lagrange
- add everything to C-CoRN

Dan's struggle

- constructive finiteness is subtle
  e.g., subsets of a finite set not necessarily finite

  various equivalent definitions of finiteness
  'the right definition'?

- Lagrange proof needs representants of cosets
  where to get the choice operator?

- also: setoids were a pain (in those days...)

John's version

249 lines of HOL Light code

statement:

```
!g h (**) i e.
    group (g,(**),i,e:A) /\ subgroup h (g,(**),i,e) /\ FINITE g
    ==> ?q. CARD(g) = CARD(q) * CARD(h) /\
            !b. b IN g ==> ?a x. a IN q /\ x IN h /\ b = a**x
```

Henk's vision

$$
\begin{aligned}
\text{computer mathematics} &\supseteq \text{formalization} \\[1em]
\text{computer mathematics} &= \text{formalization} \\
&+ \text{computer algebra} \\
&+ \text{visualization} \\
&+ \text{presentation} \\
&+ \text{exploration} \\
&+ \ldots
\end{aligned}
$$

'everything a computer can do for a mathematician, in a unified system'

Lagrange as a case study!

Tonny's version

'the best way' to formalize Lagrange?

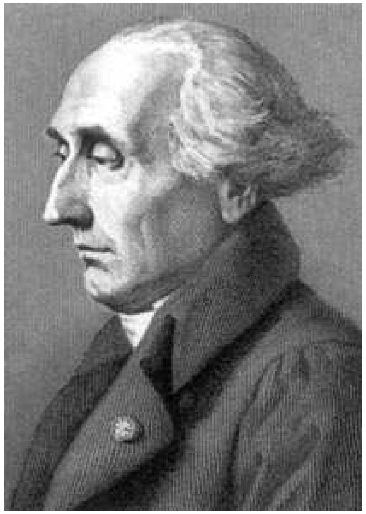
where to go in the space of possible formalizations?

- Tonny's game: nicest way
- my game: follow any textbook source

(my game covers a bigger part of the space...)

Lagrange's theorem

Giuseppe Lodovico Lagrangia



order of subgroup divides order of the group

subgroup =
subset closed under group operations

order =
number of elements

trivial:
group is partitioned by cosets, which all have the size of the subgroup

71. **Order of a Subgroup**
   HOL Light, John Harrison
   Mizar, Wojciech Trybulec
   Isabelle, Florian Kammüller
   Coq, almost C-CoRN, Dan Synek & contrib, Laurent Théry
   ProofPower, Rob Arthan
   PVS, NASA library, David Lester

van der Waerden's version

page 26 of Algebra I

the formal proof sketch game:

- select a textbook (trivial)
- translate into a formal proof sketch (trivial)
- color in the formalization (trivial, but laborious)

what is a good textbook for Lagrange?

Zwei Nebenklassen $a\mathfrak{g}$, $b\mathfrak{g}$ können sehr wohl gleich sein, ohne daß $a = b$ ist. Immer dann nämlich, wenn $a^{-1}b$ in $\mathfrak{g}$ liegt, gilt

$$b\mathfrak{g} = aa^{-1}b\mathfrak{g} = a(a^{-1}b\mathfrak{g}) = a\mathfrak{g}.$$

Zwei *verschiedene* Nebenklassen haben kein Element gemeinsam. Denn wenn die Nebenklassen $a\mathfrak{g}$ und $b\mathfrak{g}$ ein Element gemein haben, etwa

$$ag_1 = bg_2\,,$$

so folgt

$$g_1 g_2^{-1} = a^{-1}b,$$

so daß $a^{-1}b$ in $\mathfrak{g}$ liegt; nach dem Vorigen sind also $a\mathfrak{g}$ und $b\mathfrak{g}$ identisch.

Jedes Element a gehört einer Nebenklasse an, nämlich der Nebenklasse $a\mathfrak{g}$. Diese enthält ja sicher das Element $ae = a$. Nach dem eben Bewiesenen gehört das Element a auch *nur* einer Nebenklasse an. Wir können demnach jedes Element a als *Repräsentanten* der a enthaltenden Nebenklass $a\mathfrak{g}$ ansehen.

Nach dem vorhergehenden bilden die Nebenklassen eine *Klasseneinteilung* der Gruppe $\mathfrak{G}$. Jedes Element gehört einer und nur einer Klasse an.

Je zwei Nebenklassen sind gleichmächtig. Denn durch $a\mathfrak{g} \to b\mathfrak{g}$ ist eine eineindeutige Abbildung von $a\mathfrak{g}$ auf $b\mathfrak{g}$ definiert.

Die Nebenklassen sind, mit Ausnahme von $\mathfrak{g}$ selbst, *keine* Gruppen; denn eine Gruppe müßte das Einselelement enthalten.

Die Anzahl der verschiedenen Nebenklassen einer Untergruppe $\mathfrak{g}$ in $\mathfrak{G}$ heißt der *Index* von $\mathfrak{g}$ in $\mathfrak{G}$. Der Index kann endlich oder unendlich sein.

Ist N die als (endlich angenommene) Ordnung von $\mathfrak{G}$, n die von $\mathfrak{g}$, j der Index, so gilt die Relation

$$N = jn; \tag{2}$$

denn $\mathfrak{G}$ ist ja in j Klassen eingeteilt, deren jede n Elemente enthält.

Man kann für endliche Gruppen aus (2) den Index j berechnen:

$$j = \frac{N}{n}.$$

Folge. *Die Ordnung einer Untergruppe einer endlichen Gruppe ist ein Teiler der Ordnung der Gesamtgruppe.*

sketching the proof

the formal proof sketch

```
now let H,G;
 now let a,b;
  assume a^-1*b in G;
  thus b*G = a*a^-1*b*G .= a*(a^-1*b*G) .= a*G;
 end;
 for a,b st a*G <> b*G holds (a*G) /\ (b*G) = {}
 proof let a,b;
  now assume (a*G) /\ (b*G) <> {};
   consider g1,g2 such that a*g1 = b*g2;
   g1*g2^-1 = a^-1*b;
   a^-1*b in G;
   thus a*G = b*G;
  end;
  thus thesis;
 end;
```

the formal proof sketch (continued)

```
for a holds a in a*G
proof let a;
 a*e(G) = a;
 thus thesis;
end;
{a*G : a in H} is a_partition of H;
for a,b holds card(a*G) = card(b*G)
proof let a,b;
 consider f being Function of a*G,b*G such that
  for g holds f.(a*g) = b*g;
 f is bijective;
 thus thesis;
end;
```

the formal proof sketch (continued)

```
 set 'Index' = card {a*G : a in H};
 now
  let N such that N = card H;
  let n such that n = card G;
  let j such that j = 'Index';
  thus '2': N = j*n;
 end;
 thus card G divides card H;
end;
```

the formalization

filling in a fragment of the full proof

```
A3: for a,b st a*G <> b*G holds (a*G) /\ (b*G) = {}
 proof let a,b;
  now assume (a*G) /\ (b*G) <> {};
   then consider x such that
A4: x in (a*G) /\ (b*G) by XBOOLE_0:7;
A5: x in a*G & x in b*G by A4,XBOOLE_0:def 4;
   consider g1 such that
A6: x = a*g1 by A5,Th5;
   consider g2 such that
A7: x = b*g2 by A5,Th5;
   set g1G = g1;
   set g2G = g2;
   reconsider g1 as Element of H by GROUP_2:51;
   reconsider g2 as Element of H by GROUP_2:51;
A8: a*g1 = a*g1G by Th2
    .= b*g2 by A6,A7,Th2;
```

the collection

for every formal proof sketch:

- source of the informal proof
- text of the informal proof
- formal proof sketch: informal layout
- formal proof sketch: formal layout
  with errors marked in the margin
- full formal proof
  with formal proof sketch part underlined
- Mizar version used

the Lagrange formalization **live**

444 lines of Mizar code

only $\frac{1}{3}$rd of the file corresponds to the sketch

2 definitions + 7 lemmas, including:

```
for X being finite non empty set, P being a_partition of X,
    n being natural number st
  for A being set st A in P holds card A = n
holds card X = (card P)*n;
```

proof of this one lemma takes about $\frac{1}{4}$th of the file

de Bruijn factors

de Bruijn factor in time

$$\frac{\text{1 full week of work}}{\text{1 textbook page}}$$

in this case: a bit smaller

de Bruijn factor in space

$$\frac{\text{size of formalization}}{\text{size of textbook}} \approx 4$$

in this case: factor $\approx$ 1.3

specifics of the de Bruijn factor game:

- *gzip both source and translation*
  otherwise: factor $\approx$ 1.7

- *only count the part of the formalization that parallels the source*
  otherwise: factor $\approx$ 3.3

  otherwise both: factor $\approx$ 5.2

the good news and the bad news

the good news: definitions do not matter

it seems:

one can sketch any textbook proof

(importance of choice of definitions is an artefact of type theory)

the good news: a good library really helps

why did Dan have such a hard time?

- constructive complications
- not a good library about counting finite sets

the bad news: I faked it a bit

elements of the subgroup are also elements of the group

Mizar is not flexible enough to handle this transparantly (nor is Coq)

maybe using non-`struct` types for groups helps?

two approaches:

- explicit operation `[g]` that embeds the subgroup
- define new operation `h*g` for '`h*[g]`'

I used the second approach (for cosmetic reasons)

one more thing to try

porting the sketch

Mizar Light III =
Mizar proof language on top of HOL Light

with Mizar Light III it will become possible to
generate Mizar-style proofs by executing tactics

James' proposal:

play the same game in Mizar Light III!

but: algebra in HOL not very nice . . .
but: I expect that will not really matter much for my game