MACLANE PENTAGON IS SOME RECURSIVE SQUARE , NOW WHAT ?

1337777.NET
HTTPS://GITHUB.COM/1337777

Abstract

This Coq text shows that Dosen semiassociative coherence covers Maclane associative coherence by some recursive adjunction, embedding : SemiAssoc ⇆ Assoc : flattening reflection. Now what ?

Categories and Subject Descriptors F.4.1 [*Mathematical Logic And Formal Languages*]: Mechanical Theorem Proving

Keywords Coq, Coherence, Comonadic descent, Automation

1. Contents

This recursive square *normalize_map_assoc* is the "functorial" parallel to the normalization/flattening of binary trees; and is simply the unit of the reflection for the adjunction.

`Inductive` *same_assoc* : ∀ *A B* : *objects*, *arrows_assoc A B* → *arrows_assoc A B* → `Set`
:= *same_assoc_refl* : ∀ (*A B* : *objects*) (*f* : *arrows_assoc A B*), *f* ∼*a f*
| ...
| *same_assoc_bracket_left_5* : ∀ *A B C D* : *objects*,
bracket_left_assoc (*A* /\0 *B*) *C D* <*oa bracket_left_assoc A B* (*C* /\0 *D*) ∼*a*
bracket_left_assoc A B C /\1*a unitt_assoc D* <*oa bracket_left_assoc A* (*B* /\0 *C*) *D* <*oa*
unitt_assoc A /\1*a bracket_left_assoc B C D*.

`Inductive` *normal* : *objects* → `Set` :=
normal_cons1 : ∀ *l* : *letters*, *normal* (*letter l*)
| *normal_cons2* : ∀ (*A* : *objects*) (*l* : *letters*), *normal A* → *normal* (*A* /\0 *letter l*).

`Fixpoint` *normalize_aux* (*Z A* : *objects*) {`struct` *A*} : *objects* :=
`match` *A* `with`
| *letter l* ⇒ *Z* /\0 *letter l*
| *A1* /\0 *A2* ⇒ (*Z* </\0 *A1*) </\0 *A2*
`end`
`where` "Z </\0 A" := (*normalize_aux Z A*).

`Fixpoint` *normalize* (*A* : *objects*) : *objects* :=
`match` *A* `with`
| *letter l* ⇒ *letter l*
| *A1* /\0 *A2* ⇒ (*normalize A1*) </\0 *A2*
`end`.

`Fixpoint` *normalize_aux_unitrefl_assoc Y Z* (*y* : *arrows_assoc Y Z*) *A*
: *arrows_assoc* (*Y* /\0 *A*) (*Z* </\0 *A*) :=
`match` *A* `with`
| *letter l* ⇒ *y* /\1*a unitt_assoc* (*letter l*)
| *A1* /\0 *A2* ⇒ ((*y* </\1*a A1*) </\1*a A2*) <*oa bracket_left_assoc Y A1 A2*
`end`
`where` "y </\1a A" := (*normalize_aux_unitrefl_assoc y A*).

`Fixpoint` *normalize_unitrefl_assoc* (*A* : *objects*) : *arrows_assoc A* (*normalize A*) :=
`match` *A* `with`
| *letter l* ⇒ *unitt_assoc* (*letter l*)
| *A1* /\0 *A2* ⇒ (*normalize_unitrefl_assoc A1*) </\1*a A2*
`end`.

`Check` th151 : ∀ *A* : objects, normal *A* → normalize *A* = *A*.

Aborted th270 : for local variable *A* with *normal A*, although there is the propositional equality *th151*: *normalize A* = *A*, one gets that *normalize A* and *A* are not convertible (definitionally/meta equal); therefore one shall not regard *normalize_unitrefl_assoc A* and *unitt A* as sharing the same domain-codomain indices of *arrows_assoc*.

`Check` th260 : ∀ *N P* : objects, arrows_assoc *N P* → normalize *N* = normalize *P*.

Aborted lemma_coherence_assoc0 : for local variables *N*, *P* with *arrows_assoc N P*, although there is the propositional equality *th260* : *normalize N* = *normalize P*, one gets that *normalize N* and *normalize P* are not convertible (definitionally/meta equal); therefore some transport other than *eq_rect*, some coherent transport is lacked.

Below *directed y* signify that *y* is in the image of the embedding of *arrows* into *arrows_assoc*.

`Check` normalize_aux_map_assoc : ∀ (*X Y* : objects) (*x* : arrows_assoc *X Y*)
(*Z* : objects) (*y* : arrows_assoc *Y Z*), directed *y* →
∀ (*A B* : objects) (*f* : arrows_assoc *A B*),
{ *y_map* : arrows_assoc (*Y* </\0 *A*) (*Z* </\0 *B*) &
(*y_map* <oa *x* </\1a *A* ∼a *y* </\1a *B* <oa *x* /\1a *f*) × directed *y_map* }.

`Check` normalize_map_assoc : ∀ (*A B* : objects) (*f* : arrows_assoc *A B*),
{ *y_map* : arrows_assoc (normalize *A*) (normalize *B*) &
(*y_map* <oa normalize_unitrefl_assoc *A* ∼a
normalize_unitrefl_assoc *B* <oa *f*) × directed *y_map*
}.

```
Print Assumptions normalize_map_assoc.

Closed under the global context
```

References

[1] Jason Gross, Adam Chlipala, David I. Spivak. "Experience Implementing a Performant Category-Theory Library in Coq" In: Interactive Theorem Proving. Springer, 2014.

[2] Gregory Malecha, Adam Chlipala, Thomas Braibant. "Compositional Computational Reflection" In: Interactive Theorem Proving. Springer, 2014.

[3] Benjamin Delaware, Clement Pit–Claudel, Jason Gross, Adam Chlipala. "Fiat: Deductive Synthesis of Abstract Data Types in a Proof Assistant" In: Principles of Programming Languages. ACM, 2015.

[4] Adam Chlipala. "Ur/Web: A Simple Model for Programming the Web" In: Principles of Programming Languages. ACM, 2015.

[5] 1337777.org. "1337777.org"
https://github.com/1337777/upo/

[6] Robert Harper. "Programming in Standard ML"
http://www.cs.cmu.edu/~rwh/smlbook/

[7] Adam Chlipala. "Certified Programming with Dependent Types"
http://adam.chlipala.net/cpdt/

[8] Kosta Dosen, Zoran Petric. "Proof-Theoretical Coherence"
http://www.mi.sanu.ac.rs/~kosta/coh.pdf , 2007.

[9] Martin Aigner. "Combinatorial Theory" Springer, 1997

[10] Francis Borceux, George Janelidze. "Galois Theories" Cambridge University Press, 2001.

[11] Francis Borceux. "Handbook of categorical algebra. Volumes 1 2 3" Cambridge University Press, 1994.