Forum "Math-for-Industry" 2015 -The Role and Importance of Mathematics in Innovation-

A Coq Library for Interactive Formal Theorem Proving in the Theory of Relational Calculus

Toshiaki Matsushima
Graduate School of Mathematics, Kyushu University
ma214037@math.kyushu-u.ac.jp

Abstract

Relations between objects are basic mathematical concepts, so calculus and equations of relations are important. One of our final goal is to reconstruct proofs of mathematics using relational calculus and to provide formal proofs using our library. As the first step, we made a library for relational calculus using "**Coq**", which is one of the proof assistants. We defined operators and notations of relational equations, and proved various lemmas of relations. Then we implemented automatic proving procedures (*tactics*) for relational calculus, which is especially useful for lemmas about functions.

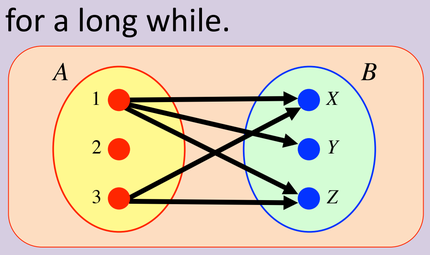
Introduction

We use many **equations of numbers** and we can calculate.

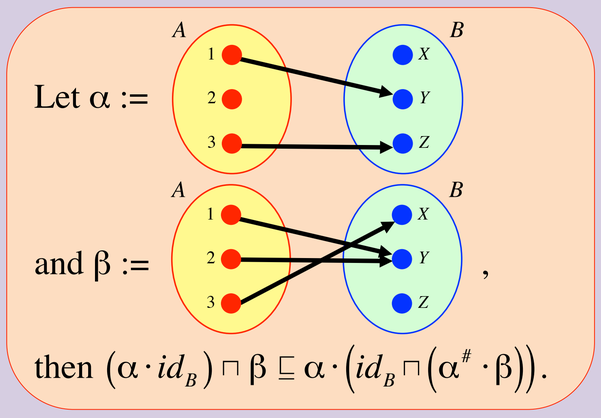
$4 \times 2 + (15 \div 3) - 7 = 6$

$x = 3 \Rightarrow x^2 + 5x + 4 = 28$

A sufficiently developed **theory of relations** has been existing for a long while.



But...
We seldom use **calculations or equations of relations**!

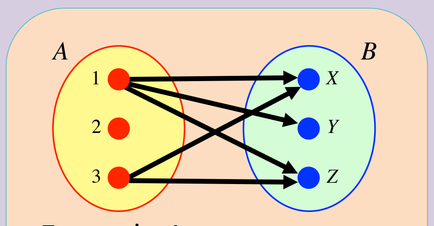


Therefore, we tried to implement a "**Relational Calculus Library**".

What is relation?

Let A and B be two sets.

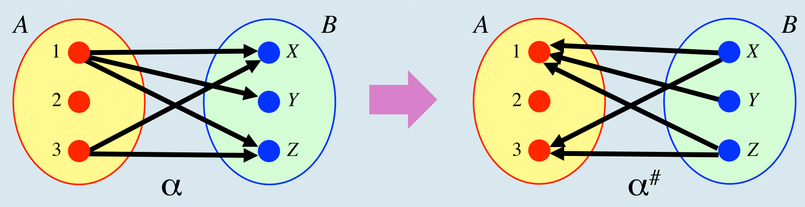
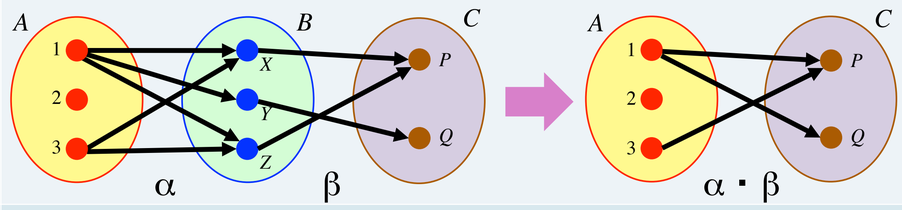
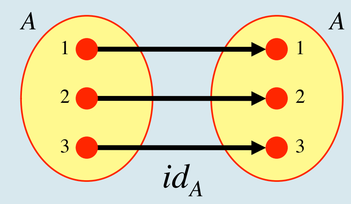
- We define "**relation from A to B**" as $A \times B \to \{0,1\}$.
- We use the notation "$A \rightharpoondown B$" as a relation.
- If there is a "connection" in $\alpha : A \rightharpoondown B$ between the element of A and that of B, then the return value is **1**. If not, it is **0**.
- We can regard $\alpha : A \rightharpoondown B$ as **a subset of $A \times B$**; $\alpha := \{(a,b) \in A \times B \mid \alpha(a,b) = 1\}$.



Example 1 :
$\alpha = \{(1,X),(1,Y),(1,Z),(3,X),(3,Z)\}$.

- As a relation $A \rightharpoondown B$ is a subset of $A \times B$, the **inclusion of relation**, **union** and **intersection** of them are available as usual and denoted by $\sqsubseteq$, $\sqcup$, and $\sqcap$, respectively.

Operators and constants

Functions and Notations in Library	Definition
inverse_relation $\alpha^{\#}$ (**alpha #**)	$\alpha^{\#} := \{(b,a) \in B \times A \mid (a,b) \in \alpha\}$ 
composite $\alpha \cdot \beta$ (**alpha · beta**)	$\alpha \cdot \beta := \{(a,c) \in A \times C \mid \exists b \in B, (a,b) \in \alpha \wedge (b,c) \in \beta\}$ 
identity_relation id_A (**Id A**)	$id_A := \{(a,a) \in A \times A\}$  Id is the identity element of composite operation.

etc...

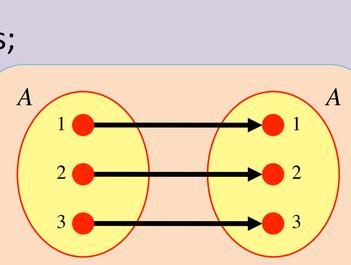
Function

- A "**function**" $f : X \to Y$ is defined in set theory like "$\forall x, \exists! y \ s.t. \ (x,y) \in f$". ("$(x,y) \in f$" means "$f(x) = y$")
- We can define this by using relational equation as follows;

"f is a function" $\Leftrightarrow id_X \sqsubseteq f \cdot f^{\#} \wedge f^{\#} \cdot f \sqsubseteq id_Y$.

- We can also define **surjection** and **injection**.

"f is a surjection" $\Leftrightarrow id_Y \sqsubseteq f^{\#} \cdot f \wedge$ "f is a function"
"f is an injection" $\Leftrightarrow f \cdot f^{\#} \sqsubseteq id_X \wedge$ "f is a function"



id is always a function.

Proof using relational calculus

Theorem 1 : If $f : X \rightharpoondown Y$ and $g : Y \rightharpoondown Z$ are functions, then $f \cdot g$ is also a function.

If we formulate this by using relational equation, it can be written as follows.

$id_X \sqsubseteq f \cdot f^{\#} \wedge f^{\#} \cdot f \sqsubseteq id_Y \wedge id_Y \sqsubseteq g \cdot g^{\#} \wedge g^{\#} \cdot g \sqsubseteq id_Z$
$\Rightarrow id_X \sqsubseteq (f \cdot g) \cdot (f \cdot g)^{\#} \wedge (f \cdot g)^{\#} \cdot (f \cdot g) \sqsubseteq id_Z$

We can prove it as follows.

$$\begin{aligned} id_X &\sqsubseteq f \cdot f^{\#} \\ &= f \cdot id_Y \cdot f^{\#} \\ &\sqsubseteq f \cdot (g \cdot g^{\#}) \cdot f^{\#} \\ &= (f \cdot g) \cdot (g^{\#} \cdot f^{\#}) \\ &= (f \cdot g) \cdot (f \cdot g)^{\#} \end{aligned}$$

$$\begin{aligned} (f \cdot g)^{\#} \cdot (f \cdot g) &= (g^{\#} \cdot f^{\#}) \cdot (f \cdot g) \\ &= g^{\#} \cdot (f^{\#} \cdot f) \cdot g \\ &\sqsubseteq g^{\#} \cdot id_Y \cdot g \\ &= g^{\#} \cdot g \\ &\sqsubseteq id_Z \end{aligned}$$

Lemmas

- $\alpha \cdot id = \alpha$
- $(\alpha \cdot \beta)^{\#} = \beta^{\#} \cdot \alpha^{\#}$
- $(\alpha \cdot \beta) \cdot \gamma = \alpha \cdot (\beta \cdot \gamma)$
- $\beta \sqsubseteq \beta' \Rightarrow \alpha \cdot \beta \cdot \gamma \sqsubseteq \alpha \cdot \beta' \cdot \gamma$

This proof using some basic lemmas is so simple that we can implement automatic proving procedures.

Tactics (procedures to assist automated proving)

- We also implemented "**tactics**" for relational calculus.
- *Tactics* are **automatic proving procedures** of Coq and its source code is shown on (Figure 1).

```
Ltac Rel_simpl1 :=
  Rel_simpl_intro;
  repeat match goal with
           | [_ : _ |- _ ⊆ _ ] => apply f_include
           | [ H : _ |- _ ⊆ _ ] => apply H
           | [_ : _ |- (_ · _) ⊆ (_ · _) ] => apply composite_include
           | [_ : _ |- (_ · _) ⊆ _ ] => apply composite_include_left_a_id
           | [_ : _ |- _ ⊆ (_ · _) ] => apply composite_include_left_id_a
           | [_ : _ |- (_ · _) ⊆ _ ] => apply composite_include_right_b_id
           | [_ : _ |- _ ⊆ (_ · _) ] => apply composite_include_right_id_b
           | [ H : _ ⊆ _ , H0 : _ ⊆ _ |- _ ⊆ _ ] => apply (include_include H H0)
           | [ H : (Id _) ⊆ _ ,H0 : _ ⊆ (Id _) |- _ ] => rewrite (include_equal H H0)
           | [_ : _ |- (_ #) ⊆ (_ #) ] => apply include_inverse
           | [_ : _ |- _ ] => rewrite composite_inverse
           | [_ : _ |- _ ] => rewrite composite_composite4
           end.
```

Figure 1 : A sample code of a *tactic*.

- Some easy theorems can be proved **just using tactics** (Figure 3).

Looks troublesome...

```
Theorem function_composite_rel
 {X Y Z : eqType}{f : Rel X Y}{g : Rel Y Z}:
 Id X ⊆ (f · (f #)) /\ ((f #) · f) ⊆ Id Y ->
 Id Y ⊆ (g · (g #)) /\ ((g #) · g) ⊆ Id Z ->
 Id X ⊆ ((f · g) · ((f · g) #)) /\ (((f · g) #) · (f · g)) ⊆ Id Z.
Proof.
elim => H H0.
elim => H1 H2.
split.
rewrite composite_inverse composite_composite4.
apply (include_include H).
apply (composite_include_left_right_id_b H1).
rewrite composite_inverse composite_composite4.
apply (fun H' => include_include H' H2).
apply (composite_include_left_right_b_id H0).
Qed.
```

← Figure 2 : The proof of Theorem 1 without using *tactics*.

↓ Figure 3 : The same proof with a *tactic*.

Very easy!

```
Theorem function_composite_rel_tactic
 {X Y Z : eqType}{f : Rel X Y}{g : Rel Y Z}:
 Id X ⊆ (f · (f #)) /\ ((f #) · f) ⊆ Id Y ->
 Id Y ⊆ (g · (g #)) /\ ((g #) · g) ⊆ Id Z ->
 Id X ⊆ ((f · g) · ((f · g) #)) /\ (((f · g) #) · (f · g)) ⊆ Id Z.
Proof.
Rel_simpl2.
Qed.
```

Future works

- To organize the levels of axioms and propositions.
- Improvement of the *tactics* for relational calculus.
- Applications to automaton theory.
- To prove the properties of the problems formulated in relational equations.
- To reform other mathematical theories using relational equations.

Figure 4 : Automaton.

References

[1] R. Affeldt and M. Hagiwara,
Formalization of Shannon's Theorems in SSReflect-Coq,
In 3rd Conference on Interactive Theorem Proving, LNCS 7406, 233–249, 2012.

[2] H. Furusawa and Y. Kawahara,
Point axioms and related conditions in dedekind categories,
Journal of Logical and Algebraic Methods in Programming, 84:359–376, 2015.

[3] The Coq Proof Assistant, https://coq.inria.fr/

Acknowledgments

We would like to express our deepest gratitude to Prof. Yoshihiro Mizoguchi, Mr. Hisaharu Tanaka and Prof. Shuichi Inokuchi, who provided helpful comments and suggestions.