Constitution of the Proof using the Isabelle Theorem Prover

Fumiya Iwama
d1023001@konan-u.ac.jp
Graduate School of Natural Science
Konan University
JAPAN

Abstract: Congruence relation have been widely used in science. We use an Isabelle theorem prover to verify and specify some lemmata in proving a certain theorem of congruence relation. This paper is about using the Isabelle theorem prover to establish the proofs of some elementary number theoretic lemmata for modulo arithmetic.

1. Introduction

1.1. Congruence relation

In abstract algebra, a congruence relation (or simply congruence) is an equivalence relation on an algebraic structure that is compatible with the structure [1]. Every congruence relation has a corresponding quotient structure, whose elements are the equivalence classes (or congruence classes) for the relation. The prototypical example of a congruence relation is congruence modulo n on the set of integers. It is an important basic principle that $A - B$ is a multiple of n. For A given positive integer n, two integers A and B are called congruent modulo n, written

$$A \equiv B \quad (mod\, n)$$

If the value of n is clear from the context, we write simply $A \equiv B$. Congruence modulo n satisfies the following theorem:

Theorem
1. $A \equiv B$ and $C \equiv D$ implies $A + C \equiv B + D$;
2. $A \equiv B$ and $C \equiv D$ implies $A - C \equiv B - D$;
3. $A \equiv B$ and $C \equiv D$ implies $AC \equiv BD$;

1.2. Isabelle theorem prover

The Isabelle theorem prover is an interactive theorem prover, a Higher Order Logic (HOL) theorem prover. It is an LCF-style theorem prover (written in Standard ML), so it is based on a small logical core to ease logical correctness. it provides a meta-logic (a weak type theory), which is used to encode object logics like first-order logic (FOL), higher-order logic (HOL) or Zermelo Fraenkel set theory (ZFC) [2]. It is characterized by being able to describe proofs in a form close to natural language. However, because there are few people to handle, the information is scarce.

2. Verification and specification of some lemmata

We prove the above theorem by using the Isabelle theorem prover.

2.1. Proof 1 of the above theorem

At first, we define a congruence.

definition
godo :: ”int ⇒ int ⇒ int ⇒ bool” where
”godo A B n ⟷ n ≤ 1 ∧ (A - B) mod n = 0”

we define first lemma (combination).

lemma ketugou_1:
fixes A :: int
and B :: int
and C :: int
and D :: int
shows”(A+C)-(B+D) = (A-B)+(C-D)”
apply simp
done

we define second lemma.

lemma le_teiri_1:
fixes A :: int
and B :: int
and n :: int
shows”A mod n = 0 ∧ B mod n = 0 → (A+B) mod n = 0”
apply auto
done

We verify the above theorem 1.

theorem mod_teiri1:
fixes A :: int
and B :: int
and C :: int
and D :: int
and n :: int
assumes ”n ≥ 1”
and ”godo A B n”
and ”godo C D n”
shows ”godo (A+C) (B+D) n”
using assms
apply (simp add:godo_def)

Here, the following problems occur.

goal (1 subgoal): (A - B) mod n = 0 ⇒ (C - D) mod n = 0
⇒ (A + C - (B + D)) mod n = 0

So, we make the lemma ketugou_1 that holds the following equation.

”(A+C)-(B+D) = (A-B)+(C-D)”

And we do the following:

```
apply (simp add:ketugou_1)
```

It is in this way renewed by the following problems.

goal (1 subgoal): (A - B) mod n = 0 ⇒ (C - D) mod n = 0
⇒ (A - B + (C - D)) mod n = 0

Then we do the following:

A mod n = 0 ∧ B mod n = 0 → (A+B) mod n = 0”

We can prove 1 of the above theorem.

```
apply (simp add:le_teiri_1)
done
```

2.2. Proof 2 of the above theorem

we define lemma 3 and 4.

```
lemma ketugou_2:
fixes A :: int
and B :: int
and C :: int
and D :: int
shows”(A-C)-(B-D) = (A-B)-(C-D)”
apply simp
done

lemma le_teiri_2:
fixes A :: int
and B :: int
and n :: int
shows”A mod n = 0 ∧ B mod n = 0 → (A-B) mod n = 0”
apply auto
```

done

We can prove the above theorem 2.

theorem mod_teiri_2:
fixes A :: int
and B :: int
and C :: int
and D :: int
and n :: int
assumes "n $\geq$ 1"
and "godo A B n"
and "godo C D n"
shows "godo (A-C) (B-D) n"
using assms
apply (simp add:godo_def)
apply (simp add:ketugou_2)
apply (simp add:le_teiri_2)
done

2.3. Proof 3 of the above theorem

we define lemma 5.

lemma le_teiri_3_1:
fixes A :: int
and B :: int
and n :: int
assumes "n $\geq$ 1"
shows"A mod n = 0 $\rightarrow$ A*B mod n = 0"
apply auto
done

Next, we define the distributive property.

definition
bunpai :: "int $\Rightarrow$ bool" where
"bunpai M $\longleftrightarrow$ ($\forall$X. $\forall$Y. X*M - Y*M = (X-Y)*M)"

And we define lemma 6,7,8 and 9.

lemma koukan_1:
fixes A :: int
and B :: int
shows"A*B = B*A"
apply simp
done

```
lemma bunpai_1:
fixes A :: int
and B :: int
and C :: int
assumes ”bunpai C”
shows”(C*A - C*B) = (A-B)*C”
using assms
apply (simp add:bunpai_def add:koukan_1)
done

lemma le_teiri3_2:
fixes A :: int
and B :: int
and C :: int
and n :: int
assumes ”bunpai C”
and ”godo A B n”
and ”n ≥ 1”
shows”A*C mod n = 0 ∧ B*C mod n = 0 → (A-B)*C mod n = 0”
using assms
apply (simp add:bunpai_def add:godo_def add:le_teiri_3_1)
done

lemma ketugou_3:
fixes A :: int
and B :: int
and C :: int
and D :: int
assumes ”bunpai C”
and ”bunpai B”
shows”A*C - B*D = (A-B)*C+(C-D)*B”
proof (induct,simp)
have ”A*C - B*D = (A*C - B*C) + (B*C - B*D)” by simp
also have ”. . . = (A-B)*C + (C-D)*B” using assms and bunpai_def and bunpai_1 by simp
finally show ”A*C - B*D = (A-B)*C+(C-D)*B” by simp
qed
```

We verify the above theorem 3.

```
theorem mod_teiri_3:
fixes A :: int
and B :: int
and C :: int
and D :: int
and n :: int
assumes ”n ≥ 1”
```

and ”godo A B n”
and ”godo C D n”
and ”bunpai C”
and ”bunpai B”
shows ”godo (A*C) (B*D) n”
using assms
apply (simp add:godo_def add:bunpai_def)

Here, the following problems occur.

(A - B) mod n = 0 ⇒ (C - D) mod n = 0 ⇒ A * C - B * C = (A - B) * C
⇒ C * B - D * B = (C - D) * B ⇒ (A * C - B * D) mod n = 0

So, we make the lemma ketugou_3 that holds the following equation.

”(A*C) - (B*D) = (A-B)*C + (C-D)*B”

And we do the following:

apply (simp add:ketugou_3 add:bunpai_def)

In this way it is rewrited like the following.

(A - B) mod n = 0 ⇒ (C - D) mod n = 0 ⇒ A * C - B * C = (A - B) * C
⇒ C * B - D * B = (C - D) * B ⇒ ((A - B) * C + (C - D) * B) mod n = 0

Then

“A mod n = 0 ∧ B mod n = 0 → (A+B) mod n = 0”
“A mod n = 0 → A*B mod n = 0”
“A*C mod n = 0 ∧ B*C mod n = 0 → (A-B)*C mod n = 0”

We can prove the above theorem 3.

apply (simp add:le_teiri_1 add:le_teiri_3_1 add:le_teiri_3_2)
done

3. Conclusions

We have formulated the specification of some lemmata and proof of a certain congruence relation using the Isabelle theorem prover. In the use of Isabelle for this work, we feel that theorems in some basic theories are not rich enough. We have to prove more theorems that are very general.

References

[1] Hungerford, Thomas W. (1974). *Algebra.* Springer-Verlag.

[2] Lawrence C. Paulson. (1989). *The foundation of a generic theorem prover.* Journal of Automated Reasoning, Volume 5, Issue 3, pp. 363-397.