A Mechanical Proof of the Chinese Remainder Theorem

David M. Russinoff

August 28, 2000

Since antiquity, the Chinese Remainder Theorem (CRT) has been regarded as one of the jewels of mathematics. An elegant result of considerable intrinsic mathematical interest, it has continually found new applications in a variety of disciplines, most notably in cryptology, information theory, and computing[4]. One recent example of a novel application is a verification system for a translator of safety-critical railroad software, implemented in ACL2 by P. Bertoli and P. Traverso [1], the correctness of which depends on CRT. That project provided an opportunity for the author to contribute to the solution of a problem of practical significance through an exercise that would otherwise have been a mere diversion. This note summarizes the results of this exercise: an ACL2 formalization and mechanical proof of CRT.

Let $m_1, \ldots, m_k$ be pairwise relatively prime moduli and let $a_1, \ldots, a_k$ be arbitrary integers. The theorem guarantees the existence of an integer x that simultaneously satisfies the congruences $x \equiv a_i \pmod{m_i}$, $i = 1, \ldots, k$.

The usual proof of CRT proceeds as follows: For $i = 1, \ldots, k$, let

$$M_i = \prod_{\substack{j=1 \\ j \neq i}}^{k} m_j.$$

Then m_i and M_i are relatively prime, and hence we can find integers r_i and s_i such that $r_i m_i + s_i M_i = 1$. Thus,

$$s_i M_i = 1 - r_i m_i \equiv 1 \pmod{m_i}$$

and for each $j \neq i$,

$$s_i M_i \equiv 0 \pmod{m_j}.$$

Let $x = \sum_{i=1}^{k} a_i s_i M_i$. Then for $i = 1, \ldots, k$,

$$x = a_i s_i M_i + \sum_{\substack{j=1 \\ j \neq i}}^{k} a_i s_i M_i \equiv a_i \cdot 1 + 0 \equiv a_i \pmod{m_i}. \square$$

Although the constructive nature of this proof suggests its suitability for formalization in the ACL2 logic, it implicitly appeals to a number of somewhat nontrivial number-theoretic results that would require some effort to prove formally. For example, the traditional proof of the proposition that an integer that is relatively prime to each of a set of integers must also be relatively prime to their product depends on Euclid's Theorem, which states that if a prime p divides a product ab, then p must divide either a or b. This theorem, among others that would facilitate the formalization of the above proof, is included in a fairly extensive library of elementary number theory[3, 5, 6] that was developed with the Nqthm prover[2], from which ACL2 has descended. None of this, however, has yet been ported to ACL2. In the interest of expediency, therefore, our first objective was to design a proof of CRT that proceeds from first principles as directly as possible. An outline of this proof and its formalization is presented below.

We shall be concerned with natural numbers and lists of natural numbers, for which we define the following recognizers:

```
(defun natp (n)
  (and (integerp n) (>= n 0)))

(defun natp-all (l)
  (if (endp l)
      t
    (and (natp (car l))
         (natp-all (cdr l))))).
```

Our formulation of CRT begins with a recursive algorithmic definition of the *greatest common divisor* (g.c.d.):

```
(defun g-c-d (x y)
  (declare (xargs :measure (nfix (+ x y))))
  (if (zp x)
      y
    (if (zp y)
        x
      (if (<= x y)
          (g-c-d x (- y x))
        (g-c-d (- x y) y))))).
```

Two naturals are *relatively prime* if their g.c.d. is 1:

```
(defun rel-prime (x y)
  (= (g-c-d x y) 1))
```

Next, we define the notion of a set of *pairwise relatively prime moduli*:

```
(defun rel-prime-all (x l)
  (if (endp l)
      t
    (and (rel-prime x (car l))
```

```
        (rel-prime-all x (cdr l)))))

(defun rel-prime-moduli (l)
  (if (endp l)
      t
    (and (integerp (car l))
         (>= (car l) 2)
         (rel-prime-all (car l) (cdr l))
         (rel-prime-moduli (cdr l))))).
```

x and y are *congruent modulo* m if they leave the same remainder upon division by m:

```
(defun congruent (x y m)
  (= (rem x m) (rem y m))).
```

Given a list of naturals and a list of moduli of the same length, the following predicate determines whether a natural x is congruent to each member of the first list modulo the corresponding member of the second:

```
(defun congruent-all (x a m)
  (if (endp m)
      t
    (and (congruent x (car a) (car m))
         (congruent-all x (cdr a) (cdr m))))).
```

CRT may now be stated as follows:

```
(defthm chinese-remainder-theorem
    (implies (and (natp-all a)
                  (rel-prime-moduli m)
                  (= (len a) (len m)))
             (and (natp (crt-witness a m))
                  (congruent-all (crt-witness a m) a m)))).
```

Note that this formulation requires an explicit solution, given by the function `crt-witness`. The definition of this function will be based on a function `one-mod` of two arguments: a natural x and a list of naturals ℓ. If each member of ℓ is relatively prime to x, then `one-mod` will return a natural that is congruent to 1 modulo x and congruent to 0 modulo each member of ℓ. Once this function is defined, the definition of `crt-witness` will be straightforward.

Sixty-five lemmas are used to lead the ACL2 prover to a proof of this theorem. Only the main lemmas will be listed here. These are taken directly from the file of events that generated the proof, except that keyword arguments (`:hints` and `:rule-classes`) are omitted.

Our first lemma states that the g.c.d. of x and y can be expressed as a linear combination $rx + sy$. We first define mutually recursive functions that compute the coefficients r and s, following the scheme of the definition of `g-c-d`. The lemma can then be stated constructively and proved automatically.

```
(mutual-recursion
 (defun r (x y)
   (declare (xargs :measure (nfix (+ x y))))
   (if (zp x)
       0
     (if (zp y)
         1
       (if (<= x y)
           (- (r x (- y x)) (s x (- y x)))
         (r (- x y) y)))))
 (defun s (x y)
   (declare (xargs :measure (nfix (+ x y))))
   (if (zp x)
       1
     (if (zp y)
         0
       (if (<= x y)
           (s x (- y x))
         (- (s (- x y) y) (r (- x y) y))))))
 )
```

```
(defthm r-s-lemma
    (implies (and (natp x)
                  (natp y))
             (= (+ (* (r x y) x)
                   (* (s x y) y))
                (g-c-d x y))))
```

More generally, if x is relatively prime to each member of a list ℓ, and p is the product of the members of ℓ, then we can find c and d such that $cx + dp = 1$. The reason for this, of course, is that x and p are relatively prime, which follows from Euclid's Theorem and the divisibility properties of the g.c.d. Rather than to prove all of this, however, we take a more direct route to the desired result.

Let $\ell = (y \ . \ \ell')$ and let p' be the product of the members of ℓ'. Using induction, suppose we have r, s, c', and d' such that

$$rx + sy = c'x + d'p' = 1.$$

We must find c and d such that

$$cx + dp = 1,$$

where $p = yp'$. But since

$$(sd')p = (sy)(d'p') = (1 - rx)(1 - c'x) = 1 - (r + c' - rc'x)x,$$

we have the solution

$$c = r + c' - rc'x$$

and

$$d = sd'.$$

This leads to the following definitions and lemma:

```
(defun c (x l)
  (if (endp l)
      0
    (- (+ (r x (car l))
          (c x (cdr l)))
       (* (r x (car l))
          (c x (cdr l))
          x))))

(defun d (x l)
  (if (endp l)
      1
    (* (s x (car l))
       (d x (cdr l)))))

(defun prod (l)
  (if (endp l)
      1
    (* (car l) (prod (cdr l)))))

(defthm c-d-lemma
    (implies (and (natp x)
                  (natp-all l)
                  (rel-prime-all x l))
             (= (+ (* (c x l) x)
                   (* (d x l) (prod l)))
                1)))
```

Now, if x and the members of ℓ form a set of pairwise relatively prime moduli, then we can construct a natural that is congruent to 1 modulo x and congruent to 0 modulo each member of ℓ, namely,

$$(dp)^2 = (1 - cx)^2.$$

Thus, the formal proof proceeds as follows:

```
(defun one-mod (x l) (* (d x l) (prod l) (d x l) (prod l)))

(defthm rem-one-mod-1
    (implies (and (natp x)
                  (> x 1)
                  (natp-all l)
                  (rel-prime-all x l))
             (= (rem (one-mod x l) x) 1)))
```

```
(defthm rem-one-mod-0
    (implies (and (natp x)
                  (> x 1)
                  (rel-prime-moduli l)
                  (rel-prime-all x l)
                  (member y l))
             (= (rem (one-mod x l) y) 0)))
```

The definition of the CRT witness function is now straightforward, but requires an auxiliary recursive definition:

```
(defun crt1 (a m l)
  (if (endp a)
      0
    (+ (* (car a) (one-mod (car m) (remove (car m) l)))
       (crt1 (cdr a) (cdr m) l))))

(defun crt-witness (a m) (crt1 a m m))
```

Thus, in the computation of `crt-witness`, each member of a is multiplied by a number that is congruent to 1 modulo the corresponding member of m and congruent to 0 modulo every other member of m, and the sum of these products is returned. It is clear that this value satisfies CRT; the only remaining trick required to complete the proof is to formulate an appropriate induction-loading generalization of the theorem:

```
(defthm crt1-lemma
    (implies (and (natp-all a)
                  (rel-prime-moduli m)
                  (= (len a) (len m))
                  (rel-prime-moduli l)
                  (sublistp m l))
             (congruent-all (crt1 a m l) a m))).
```

The proof of this lemma is based on `rem-one-mod-1` and `rem-one-mod-0`, and uses the induction scheme suggested by the definitions of `congruent-all` and `crt1`. Note that when these two definitions are expanded in the inductive case, the conclusion of the lemma becomes

```
(and (congruent (+ (* (car a)
                      (one-mod (car m) (remove (car m) l)))
                   (crt1 (cdr a) (cdr m) l))
                (car a)
                (car m))
     (congruent-all (+ (* (car a)
                          (one-mod (car m) (remove (car m) l)))
                       (crt1 (cdr a) (cdr m) l))
                    (cdr a)
                    (cdr m))).
```

An auxiliary lemma states that `(crt1 a m l)` has remainder 0 modulo each member of ℓ that does not belong to m. This, together with `one-mod-1`, may be used to establish the first conjunct of the above goal. Using `one-mod-0`, we may reduce the second conjunct to

```
(congruent-all (crt1 (cdr a) (cdr m) l)
               (cdr a)
               (cdr m))),
```

which then follows by induction.

Finally, CRT is easily derived from the last lemma, simply by substituting m for ℓ:

```
(defthm chinese-remainder-theorem
    (implies (and (natp-all a)
                  (rel-prime-moduli m)
                  (= (len a) (len m)))
             (and (natp (crt-witness a m))
                  (congruent-all (crt a m) a m))))
```

References

[1] P. Bertoli and P. Traverso, "Design Verification of a Safety-Critical Embedded Verifier", in M. Kaufmann, P. Manolios, and J Moore, eds., *Computer-Aided Reasoning: ACL2 Case Studies*, Kluwer Academic Press, 2000.

[2] R. S. Boyer and J S. Moore. *A Computational Logic*. Academic Press, New York, 1979.

[3] R. S. Boyer and J S. Moore. "Proof Checking the RSA Public Key Encryption Algorithm", *American Mathematical Monthly* 91:3 (1984), 181-189.

[4] C. Ding, D. Pei, and A. Salomaa, *Chinese Remainder Theorem: Applications in Computing, Coding, Cryptography*, World Scientific Publishing Co., 1996.

[5] D. M. Russinoff, "An Experiment with the Boyer-Moore Theorem Prover: A Proof of Wilson's Theorem", *Journal of Automated Reasoning* 1:2 (1985), 121-139.

[6] D. M. Russinoff, "A Mechanical Proof of Quadratic Reciprocity", *Journal of Automated Reasoning* 8:1 (1992), 3-21.