Copyright (C) 1994 by Computational Logic, Inc. All Rights Reserved.

This script is hereby placed in the public domain, and therefore unlimited editing and redistribution is permitted.

NO WARRANTY

Computational Logic, Inc. PROVIDES ABSOLUTELY NO WARRANTY. THE EVENT SCRIPT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE SCRIPT IS WITH YOU. SHOULD THE SCRIPT PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

IN NO EVENT WILL Computational Logic, Inc. BE LIABLE TO YOU FOR ANY DAMAGES, ANY LOST PROFITS, LOST MONIES, OR OTHER SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THIS SCRIPT (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY THIRD PARTIES), EVEN IF YOU HAVE ADVISED US OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY OTHER PARTY.

|# |

; Matt Wilding

EVENT: Start with the library "extras" using the compiled version.

;;; Proof of Matijasevich's lemma constructed by MW week of 5-10-91. Inspired by ;;; proof in Concrete Mathematics by Graham, Knuth, and Patashnik and notes from ;;; Dijkstra's Capita Selecta Spring '90 course.

;;;; Some additional facts needed in the proof that appear useful
;;;; enough to add to libraries (and a few facts to prove them)

THEOREM: irem-igcd-arg1
\[(\text{irem} (\text{igcd} (a, b), c) = 0) = ((\text{irem} (a, c) = 0) \land (\text{irem} (b, c) = 0))\]

THEOREM: irem-x-x
\[\text{irem} (x, x) = 0\]

;;; The nonsensical induction hint is a silly trick to get the prover ;;; to throw an equality away after using it
Theorem: gcd-times-easy-proof-helper-helper
\[ ((a \in \mathbb{N}) \land (b \in \mathbb{N}) \land (c \in \mathbb{N})) \rightarrow (((c \cdot \gcd(a, b)) \mod \gcd(a, b \cdot c)) = 0) \]

Theorem: gcd-times-easy-proof-helper
\[ ((c \cdot \gcd(a, b)) \mod \gcd(a, b \cdot c)) = 0 \]

Theorem: gcd-times-easy-proof
\[ (\gcd(a, b) = 1) \rightarrow (\gcd(a, b \cdot c) = \gcd(a, c)) \]

;; Would the more general case of rewriting gcd-times regardless
;; of the value of gcd with one of the times arguments be superior?

Theorem: gcd-times-easy
\[ (\gcd(a, b) = 1) \rightarrow ((\gcd(a, b \cdot c) = \gcd(a, c)) \land (\gcd(a, c \cdot b) = \gcd(a, c))) \]

Theorem: lessp-gcd3
\[ ((\gcd(a, b) < a) = (((b \mod a) \neq 0) \land (a \neq 0) \land (b \neq 0))) \land ((\gcd(b, a) < a) = (((b \mod a) \neq 0) \land (a \neq 0) \land (b \neq 0))) \]

Theorem: equal-remainder-gcd-0-proof
\[ ((\gcd(a, b) \mod a) = 0) = ((b \mod a) = 0) \]

;; not used in proof (though equal-remainder-gcd-0-proof is)

Theorem: equal-remainder-gcd-0
\[ (((\gcd(a, b) \mod a) = 0) = ((b \mod a) = 0)) \land (((\gcd(b, a) \mod a) = 0) = ((b \mod a) = 0)) \]

Theorem: equal-remainder-x-y-x
\[ ((x \mod y) = x) = (((x \in \mathbb{N}) \land (x < y)) \lor (x = 0) \lor ((\text{fix}(y) = 0) \land (x \in \mathbb{N}))) \]

Theorem: gcd-a-gcd-exp-a
\[ \gcd(a, \exp(a, b)) = \begin{cases} 1 & \text{if } b \simeq 0 \\ \text{fix}(a) & \text{endif} \end{cases} \]

Theorem: equal-gcd-times-1-help1
\[ ((\gcd(a, b) = 1) \land (\gcd(a, c) = 1)) \rightarrow (\gcd(a, b \cdot c) = 1) \]

Theorem: remainder-gcd-gcd-0-hack
\[ (\gcd(a, b \cdot c) \mod \gcd(a, c)) = 0 \]
Theorem: lessp-gcd-times
\((a \not\equiv 0) \lor ((b * c) \not\equiv 0) \rightarrow (\gcd(a, b * c) \not< \gcd(a, c))\)

Theorem: lessp-1-hack
\((1 < x) = ((x \not\equiv 0) \land (x \not= 1))\)

Theorem: equal-gcd-times-1-help2
\((\gcd(a, b * c) = 1) \rightarrow (\gcd(a, c) = 1)\)

Theorem: equal-gcd-times-1
\((\gcd(a, b * c) = 1) = ((\gcd(a, b) = 1) \land (\gcd(a, c) = 1))\)

Theorem: gcd-exp-1
\((\gcd(a, \exp(b, c)) = 1) = ((\gcd(a, b) = 1) \lor (c \equiv 0))\)
\(\land\ (\gcd(\exp(b, c), a) = 1) = ((\gcd(a, b) = 1) \lor (c \equiv 0))\)

;; mistake in other version of this lemma limits applicability and
;; makes this version needed

Theorem: gcd-remainder-times-fact1-proof2
\((\gcd(a, b) = 1) \rightarrow (((c * b) \mod a) = 0) = ((c \mod a) = 0)\)

Theorem: times-quotient-better
\((x \mod y = 0) \rightarrow (((x \div y) * y) = \text{if } y \equiv 0 \text{ then } 0 \text{ else } \text{fix}(x) \text{ endif}) \land (((y * (x \div y)) = \text{if } y \equiv 0 \text{ then } 0 \text{ else } \text{fix}(x) \text{ endif}))\)

Theorem: equal-remainder-exp-0
\(((a \mod c) = 0) \land (b \not\equiv 0) \rightarrow ((\exp(a, b) \mod c) = 0)\)

Theorem: remainder-2-hack
\((2 \mod x) = \text{if } (x = 1) \lor (x = 2) \text{ then } 0 \text{ else } 2 \text{ endif}\)

Theorem: remainder-times-hack2
\((a \mod b) \not= 0) \rightarrow (((a \mod (b * c)) \not= 0) \land ((a \mod (c * b)) \not= 0))\)

Theorem: gcd-a-add1-a
\(\gcd(a, 1 + a) = 1\)
These facts should perhaps be generalized into a meta lemma that
factors first

**Theorem:** remainder-times-times-hack
\[(b \ast (a \ast c)) \mod (a \ast x) = (a \ast ((b \ast c) \mod x))\]

**Theorem:** remainder-plus-times-hack
\[(((a \ast x) + (b \ast (c \ast (a \ast y)))) \mod (a \ast z)) = (a \ast ((x + (b \ast (c \ast y))) \mod z))\]

Now, on to the proof

**Definition:**
\[
\text{fib (} x \text{)} = \begin{cases} 
0 & \text{if } x \simeq 0 \\
1 & \text{if } x = 1 \\
\text{fib (} (x - 1) - 1 \text{)} + \text{fib (} x - 1 \text{)} & \text{else}
\end{cases}
\]

**Theorem:** fib-plus
\[
\text{fib (} j + k \text{)} = \begin{cases} 
\text{fib (} k \text{)} & \text{if } j \simeq 0 \\
\text{fib (} 1 + k \ast \text{fib (} j \text{)} \text{)} + \text{fib (} j - 1 \ast \text{fib (} k \text{)} \text{)} & \text{else}
\end{cases}
\]

**Theorem:** gcd-fib-next-fib
\[\gcd (\text{fib (} n \text{)}, \text{fib (} 1 + n \text{)}) = 1\]

**Theorem:** gcd-fib
\[\gcd (\text{fib (} a \text{)}, \text{fib (} b \text{)}) = \text{fib (} \gcd (a, b) \text{)}\]

**Theorem:** equal-fib-constant
\[\ ((\text{fib (} a \text{)} = 0) = (a \simeq 0)) \land ((\text{fib (} a \text{)} = 1) = ((a = 1) \lor (a = 2)))\]

**Theorem:** fib-small
\[a < 3 \rightarrow (\text{fib (} a \text{)} = \begin{cases} 
0 & \text{if } a \simeq 0 \\
1 & \text{else}
\end{cases})\]

**Definition:**
\[\text{double-fib-induction (} a, b \text{)} = \begin{cases} 
\text{if } (a < 3) \lor (b < 3) \text{ then } t & \text{else}
\end{cases} \begin{cases} 
\text{double-fib-induction (} a - 1, b - 1 \text{)} & \text{double-fib-induction (} (a - 1) - 1, (b - 1) - 1 \text{)}
\end{cases}\]
Theorem: lessp-fib-fib
\[(\text{fib}(a) < \text{fib}(b)) = ((a < b) \land (\neg ((a = 1) \land (b = 2))))\]

Theorem: remainder-fib-fib-0
\[(\text{fib}(a) \mod \text{fib}(b)) = 0 = (((a \mod b) = 0) \lor (b = 2))\]

;; fib-times-open and fib-add1-times-open apply fib-plus to
;; particular terms in a way that will be useful in the proof
;; of fib-times-special

Theorem: fib-times-open
\[(0 < k) \rightarrow (\text{fib}(k \ast n) = ((\text{fib}(n) \ast \text{fib}(1 + ((k - 1) \ast n)))
+ (\text{fib}(n - 1) \ast \text{fib}((k - 1) \ast n))))\]

Event: Disable fib-times-open.

Theorem: fib-add1-times-open
\[(0 < k) \rightarrow (\text{fib}(1 + (k \ast n)) = ((\text{fib}(1 + ((k - 1) \ast n)) \ast \text{fib}(1 + n))
+ (\text{fib}((k - 1) \ast n) \ast \text{fib}(n))))\]

Event: Disable fib-add1-times-open.

Theorem: fib-times-special-step
\[(1 < k) \rightarrow ((k \ast (\text{fib}(n) \ast \exp(\text{fib}(1 + n), k - 1))) \mod (\text{fib}(n) \ast \text{fib}(n))) = ((\text{fib}(n) \ast \exp(\text{fib}(1 + n), k - 1))
+ (\text{fib}(n - 1)
* (k - 1)
* (\text{fib}(n)
* \exp(\text{fib}(1 + n), (k - 1) - 1)))) \mod (\text{fib}(n) \ast \text{fib}(n)))\]

Theorem: fib-remainder-hack
\[\text{fib}(x \ast y) \mod \text{fib}(x) = 0\]

;; We wish the rewriter to use a modulo arithmetic equality to rewrite
;; a term. Fortunately, we can construct REMAINDER-FIB-BACKCHAIN
;; to do it the way we wish since the terms are of a specialized form

Theorem: remainder-backchain-proof1
\[((a \mod c) = (b \mod c)) \rightarrow (((a \ast d) \mod c) = ((b \ast d) \mod c))\]
Theorem: remainder-backchain-proof^2

\[ (((a \mod c) = (b \mod c)) \land ((d \mod c) = (e \mod c))) \rightarrow (((a + d) \mod c) = ((b + e) \mod c)) \]

Theorem: remainder-fib-backchain

\[ (((\text{fib} (x) \mod c) = (\exp (p, q) \mod c)) \land ((\text{fib} (y) \mod c) = ((r \ast s) \mod c))) \rightarrow (((a \ast \text{fib} (x)) + (b \ast \text{fib} (y))) \mod c) = (((a \ast \exp (p, q)) + (b \ast (r \ast s))) \mod c) \]

;; This is the guts of the proof

Theorem: fib-times-special

\[ (1 < k) \rightarrow (((\text{fib} (k \ast n) \mod \exp (\text{fib} (n), 2)) = ((k \ast (\text{fib} (n) \ast \exp (\text{fib} (1 + n), k - 1))) \mod \exp (\text{fib} (n), 2))) \land ((\text{fib} (1 + (k \ast n)) \mod \exp (\text{fib} (n), 2)) = (\exp (\text{fib} (1 + n), k) \mod \exp (\text{fib} (n), 2)))) \]

Theorem: fib-remainder-times-special

\[ (\text{fib} (n \ast k) \mod (\text{fib} (n) \ast \text{fib} (n))) = (((k \ast (\text{fib} (n) \ast \exp (\text{fib} (1 + n), k - 1))) \mod (\text{fib} (n) \ast \text{fib} (n)))) \]

Theorem: remainder-times-times-hack^2

\[ ((k \ast n) \mod (n \ast p)) = (n \ast (k \mod p)) \]

Theorem: matijasevich-lemma-helper

\[ (2 < n) \land ((m \mod n) = 0) \rightarrow (((\text{fib} (m) \mod (\text{fib} (n) \ast \text{fib} (n))) = 0) = ((m \mod (n \ast \text{fib} (n))) = 0)) \]

Theorem: matijasevich-lemma

\[ (2 < n) \rightarrow (((\text{fib} (m) \mod (\text{fib} (n) \ast \text{fib} (n))) = 0) = ((m \mod (n \ast \text{fib} (n))) = 0)) \]
Index

double-fib-induction, 4

equal-fib-constant, 4
equal-gcd-times-1, 3
equal-gcd-times-1-help1, 2
equal-gcd-times-1-help2, 3
equal-remainder-exp-0, 3
equal-remainder-gcd-0, 2
equal-remainder-gcd-0-proof, 2
equal-remainder-x-y-x, 2
exp, 2, 3, 5, 6

fib, 4–6
fib-add1-times-open, 5
fib-plus, 4
fib-remainder-hack, 5
fib-remainder-times-special, 6
fib-small, 4
fib-times-open, 5
fib-times-special, 6
fib-times-special-step, 5

gcd, 2–4
gcd-a-add1-a, 3
gcd-a-gcd-exp-a, 2
gcd-exp-1, 3
gcd-fib, 4
gcd-fib-next-fib, 4
gcd-remainder-times-fact1-proof
2, 3
gcd-times-easy, 2
gcd-times-easy-proof, 2
gcd-times-easy-proof-helper, 2
gcd-times-easy-proof-helper-helper, 2

igcd, 1
irem, 1
irem-igcd-arg1, 1
irem-x-x, 1

lessp-fib-fib, 5
lessp-gcd-times, 3
lessp-gcd3, 2

matijasevich-lemma, 6
matijasevich-lemma-helper, 6

remainder-2-hack, 3
remainder-backchain-proof1, 5
remainder-backchain-proof2, 6
remainder-fib-backchain, 6
remainder-fib-fib-0, 5
remainder-gcd-gcd-0-hack, 2
remainder-plus-times-hack, 4
remainder-times-hack2, 3
remainder-times-times-hack, 4
remainder-times-times-hack2, 6

times-quotient-better, 3