```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; In an unpublished paper, Textbook Examples of Recursion, Donald E.
; Knuth of Stanford University gives the following generalization of
; McCarthy's 91 function:

; Let a be a real, let b and d be positive reals, and let c be a
; positive integer.

; Define K( x ) for integer inputs x by

;   K( x ) <== if  x > a  then  x - b
;                         else  K( ... K( x+d ) ... ).

; Here the else-clause in this definition has c applications of the
; function K.

; When a = 100, b = 10, c = 2, and d = 11, the definition specializes
; to McCarthy's original 91 function:

;   K( x ) <== if  x > 100  then  x - 10
;                           else  K( K( x+11 ) ).

; Knuth calls the first definition of K given above, the generalized
; 91 recursion scheme with parameters ( a,b,c,d ).

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; The purpose of this file of Boyer-Moore-Kaufmann events is to
; provide mechanical verification of the following theorem given by
; Knuth in his paper.

; Theorem.  The generalized 91 recursion with parameters ( a,b,c,d )
;           defines a total function on the integers if and only if
;           (c-1)b < d.  In such a case the values of K( x ) also
;           satisfy the much simpler recurrence

;           K( x )  =  if  x > a  then  x - b
;                                 else  K( x+d-(c-1)b ).

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; The first problem to solve is: How can Knuth's problem be stated so
; the theorem prover can work on it?
; Define two mutually recursive ( partial ) functions:

;   K( a,b,c,d,x ) <== if  x > a  then  x - b
;                                 else  IterateK( a,b,c,d,c,x+d ).

;   IterateK( a,b,c,d,e,x ) <== if  e <= 1
;                                   then  K( a,b,c,d,x )
;                                   else  K( a,b,c,d,
;                                            IterateK( a,b,c,d,e-1,x))

; Knuth's parameters a, b, c, and d are included in the formal
; parameters of both K and IterateK because the theorem prover does
; not allow functions with definitions which contain "global"
; variables.

; Intuitively, IterateK iterates K e times, that is, K is applied e
; times.

; When the specified number, e, of times K is be iterated is is not
; positive, K is iterated one time.  That is, when e<1 the result in
; IterateK is the same as if e were 1.

; Thus K( a,b,c,d,x ) = IterateK( a,b,c,d,1,x ).

; Since the theorem prover does not deal with reals, the parameters
; a,b,c, and d are assumed to be integers.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; To avoid the complications of dealing with mutually recursive
; partial functions, a suggestion of M. Kaufmann is followed:

; In the definition of IterateK, replace occurrences of K with the
; body of K.

; Define a recursive partial function by

;  IterK( a,b,c,d,e,x ) <== if  1 < e
;                               then  IterK( a,b,c,d,1,
;                                            IterK( a,b,c,d,e-1,x ))
;                               else if  a < x
;                                        then  x - b
;                                        else  IterK( a,b,c,d,c,x+d).
; Then the partial function K can be defined by

;   K( a,b,c,d,x ) <== IterK( a,b,c,d,1,x ).

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; K( a,b,c,d,x ) is said to exist just in case the tuple of integers
; ( a,b,c,d,x ) is in the domain of those input values where the
; partial function K halts and produces output values.

; Knuth's theorem follows from parts 1, 4, 6, 8, and 9 of the
; following Main Theorem.

;  1. If  x > a, then  K( a,b,c,d,x )  exists.

;  2. If  x <= a, d <= 0, and c <= 1, then  K( a,b,c,d,x )  does not
;     exist.

;  3. If  x <= a, d <= 0, and c > 1, then  K( a,b,c,d,x )  does not
;     exist.

;  4. If  x <= a, d > 0, and c <= 1, then  K( a,b,c,d,x )  exists.

;  5. If  x <= a, d > 0, c > 1, and b <= 0, then  K( a,b,c,d,x )
;     exists.

;  6. If  x <= a, d > 0, c > 1, and b > 0, then  K( a,b,c,d,x )
;     exists if and only if  K( a,b,c,d,x+d-(c-1)b )  exists and
;     K( a,b,c,d,x ) = K( a,b,c,d,x+d-(c-1)b ).

;  7. If  d > 0, c > 1, b > 0, and (c-1)b < d, then
;     K( a,b,c,d,x )  exists if and only if  K( a,b,1,d-(c-1)b,x )
;     exists and  K( a,b,c,d,x ) = K( a,b,1,d-(c-1)b,x ).

;  8. If  d > 0, c > 1, b > 0, and (c-1)b < d, then
;     K( a,b,c,d,x ) exists.

;  9. If  x <= a, d > 0, c > 1, b > 0, and (c-1)b >= d, then
;     K( a,b,c,d,x ) does not exist

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
; Use the library of integer facts.
```

EVENT: Start with the library "integers".

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Define the partial function IterK using EVAL$,
; and define the partial function K using IterK.
```

DEFINITION:

```
iterk (a, b, c, d, e, x)
=   eval$ (t,
          '(if
            (ilessp '1 e)
            (iterk a b c d
             '1
             (iterk a b c d (idifference e '1) x))
            (if
             (ilessp a x)
             (idifference x b)
             (iterk a b c d c (iplus x d)))),
          list (cons ('a, a),
                cons ('b, b),
                cons ('c, c),
                cons ('d, d),
                cons ('e, e),
                cons ('x, x)))
```

DEFINITION: k (*a*, *b*, *c*, *d*, *x*) = iterk (*a*, *b*, *c*, *d*, 1, *x*)

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Recursively distribute V&C-APPLY$ throughout the body of the
; defintion of ITERK. Break the definition into several cases.
```

THEOREM: iterk-exists-iff-body-exists

```
 v&c-apply$ ('iterk,
             list (cons (a, 0),
                   cons (b, 0),
                   cons (c, 0),
                   cons (d, 0),
                   cons (e, 0),
                   cons (x, 0)))
```

```
↔   v&c-apply$ ('if,
                list (v&c-apply$ ('ilessp, list ('(1 . 0), cons (e, 0))),
                     v&c-apply$ ('iterk,
                                 list (cons (a, 0),
                                      cons (b, 0),
                                      cons (c, 0),
                                      cons (d, 0),
                                      '(1 . 0),
                                      v&c-apply$ ('iterk,
                                                  list (cons (a, 0),
                                                       cons (b, 0),
                                                       cons (c, 0),
                                                       cons (d, 0),
                                                       v&c-apply$ ('idifference,
                                                                   cons (cons (e,
                                                                               0),
                                                                         '((1
                                                                            . 0)))),
                                                       cons (x, 0))))),
                     v&c-apply$ ('if,
                                 list (v&c-apply$ ('ilessp,
                                                   list (cons (a, 0), cons (x, 0))),
                                      v&c-apply$ ('idifference,
                                                  list (cons (x, 0), cons (b, 0))),
                                      v&c-apply$ ('iterk,
                                                  list (cons (a, 0),
                                                       cons (b, 0),
                                                       cons (c, 0),
                                                       cons (d, 0),
                                                       cons (c, 0),
                                                       v&c-apply$ ('iplus,
                                                                   list (cons (x,
                                                                               0),
                                                                        cons (d,
                                                                              0)))))))))
```

Theorem: iterk-value=body-value

```
car (v&c-apply$ ('iterk,
                 list (cons (a, 0),
                      cons (b, 0),
                      cons (c, 0),
                      cons (d, 0),
                      cons (e, 0),
                      cons (x, 0))))
```

```
=    car (v&c-apply$ ('if,
                         list (v&c-apply$ ('ilessp, list ('(1 . 0), cons (e, 0))),
                               v&c-apply$ ('iterk,
                                              list (cons (a, 0),
                                                   cons (b, 0),
                                                   cons (c, 0),
                                                   cons (d, 0),
                                                   '(1 . 0),
                                                   v&c-apply$ ('iterk,
                                                                  list (cons (a, 0),
                                                                       cons (b, 0),
                                                                       cons (c, 0),
                                                                       cons (d, 0),
                                                                       v&c-apply$ ('idifference,
                                                                                      cons (cons (e,
                                                                                                  0),
                                                                                            '((1
                                                                                                . 0)))),
                                                                       cons (x, 0))))),
                               v&c-apply$ ('if,
                                              list (v&c-apply$ ('ilessp,
                                                                  list (cons (a, 0),
                                                                       cons (x, 0))),
                                                   v&c-apply$ ('idifference,
                                                                  list (cons (x, 0),
                                                                       cons (b, 0))),
                                                   v&c-apply$ ('iterk,
                                                                  list (cons (a, 0),
                                                                       cons (b, 0),
                                                                       cons (c, 0),
                                                                       cons (d, 0),
                                                                       cons (c, 0),
                                                                       v&c-apply$ ('iplus,
                                                                                      list (cons (x,
                                                                                                 0),
                                                                                           cons (d,
                                                                                                 0))))))))))
```

THEOREM: cost-is-a-numberp
cdr (v&c-apply$ (*fn*, *args*)) $\in \mathbf{N}$

THEOREM: cost>0-if-fn-exists
v&c-apply$ (*fn*, *args*) → (0 < cdr (v&c-apply$ (*fn*, *args*)))

THEOREM: iterk-cost>body-cost

```
v&c-apply$ ('iterk,
            list (cons (a, 0),
                  cons (b, 0),
                  cons (c, 0),
                  cons (d, 0),
                  cons (e, 0),
                  cons (x, 0)))
→    (cdr (v&c-apply$ ('if,
                      list (v&c-apply$ ('ilessp, list ('(1 . 0), cons (e, 0))),
                            v&c-apply$ ('iterk,
                                        list (cons (a, 0),
                                              cons (b, 0),
                                              cons (c, 0),
                                              cons (d, 0),
                                              '(1 . 0),
                                              v&c-apply$ ('iterk,
                                                          list (cons (a, 0),
                                                                cons (b, 0),
                                                                cons (c, 0),
                                                                cons (d, 0),
                                                                v&c-apply$ ('idifference,
                                                                            cons (cons (e,
                                                                                        0),
                                                                                  '((1
                                                                                     . 0)))),
                                                                cons (x, 0))))),
                            v&c-apply$ ('if,
                                        list (v&c-apply$ ('ilessp,
                                                          list (cons (a, 0),
                                                                cons (x, 0))),
                                              v&c-apply$ ('idifference,
                                                          list (cons (x, 0),
                                                                cons (b, 0))),
                                              v&c-apply$ ('iterk,
                                                          list (cons (a, 0),
                                                                cons (b, 0),
                                                                cons (c, 0),
                                                                cons (d, 0),
                                                                cons (c, 0),
                                                                v&c-apply$ ('iplus,
                                                                            list (cons (x,
                                                                                        0),
                                                                                  cons (d,
                                                                                        0))))))))))
```

```
  <   cdr (v&c-apply$ ('iterk,
                       list (cons (a, 0),
                             cons (b, 0),
                             cons (c, 0),
                             cons (d, 0),
                             cons (e, 0),
                             cons (x, 0)))))
```

THEOREM: v&c-apply$-if

```
 v&c-apply$ ('if, args)
 =   if car (args)
     then if caar (args)  then fix-cost (cadr (args), 1 + cdar (args))
          else fix-cost (caddr (args), 1 + cdar (args)) endif
     else f endif
```

THEOREM: iterk-exists-iff-body-exists-when-e>1

```
 ilessp (1, e)
 →   (v&c-apply$ ('iterk,
                  list (cons (a, 0),
                        cons (b, 0),
                        cons (c, 0),
                        cons (d, 0),
                        cons (e, 0),
                        cons (x, 0)))
      ↔   v&c-apply$ ('iterk,
                      list (cons (a, 0),
                            cons (b, 0),
                            cons (c, 0),
                            cons (d, 0),
                            '(1 . 0),
                            v&c-apply$ ('iterk,
                                        list (cons (a, 0),
                                              cons (b, 0),
                                              cons (c, 0),
                                              cons (d, 0),
                                              v&c-apply$ ('idifference,
                                                          cons (cons (e, 0),
                                                                '((1 . 0)))),
                                              cons (x, 0))))))
```

THEOREM: iterk-value=body-value-when-e>1

```
 ilessp (1, e)
 →   (car (v&c-apply$ ('iterk,
                       list (cons (a, 0),
                             cons (b, 0),
                             cons (c, 0),
                             cons (d, 0),
                             cons (e, 0),
                             cons (x, 0))))
    =    car (v&c-apply$ ('iterk,
                             list (cons (a, 0),
                                   cons (b, 0),
                                   cons (c, 0),
                                   cons (d, 0),
                                   '(1 . 0),
                                   v&c-apply$ ('iterk,
                                                  list (cons (a, 0),
                                                       cons (b, 0),
                                                       cons (c, 0),
                                                       cons (d, 0),
                                                       v&c-apply$ ('idifference,
                                                                      cons (cons (e, 0),
                                                                             '((1
                                                                                 . 0)))),
                                                       cons (x, 0)))))))
```

THEOREM: iterk-cost>body-cost-when-e>1

```
 (ilessp (1, e)
  ∧    v&c-apply$ ('iterk,
                      list (cons (a, 0),
                           cons (b, 0),
                           cons (c, 0),
                           cons (d, 0),
                           cons (e, 0),
                           cons (x, 0))))
 →    (cdr (v&c-apply$ ('iterk,
                            list (cons (a, 0),
                                 cons (b, 0),
                                 cons (c, 0),
                                 cons (d, 0),
                                 '(1 . 0),
                                 v&c-apply$ ('iterk,
                                                list (cons (a, 0),
                                                     cons (b, 0),
                                                     cons (c, 0),
                                                     cons (d, 0),
                                                     v&c-apply$ ('idifference,
                                                                    cons (cons (e, 0),
                                                                           '((1 . 0)))),
                                        cons (x, 0))))))
    <   cdr (v&c-apply$ ('iterk,
                          list (cons (a, 0),
                                cons (b, 0),
                                cons (c, 0),
                                cons (d, 0),
                                cons (e, 0),
                                cons (x, 0)))))
```

THEOREM: eq-args-give-eq-existence

```
((fn ≠ 'quote)
 ∧  (fn ≠ 'if)
 ∧  (strip-cars (args1) = strip-cars (args2))
 ∧  ((f ∈ args1) = (f ∈ args2)))
→   (v&c-apply$ (fn, args1) ↔ v&c-apply$ (fn, args2))
```

THEOREM: eq-args-give-eq-values

```
((fn ≠ 'quote)
 ∧  (fn ≠ 'if)
 ∧  (strip-cars (args1) = strip-cars (args2))
 ∧  ((f ∈ args1) = (f ∈ args2)))
→   (car (v&c-apply$ (fn, args1)) = car (v&c-apply$ (fn, args2)))
```

THEOREM: eq-args-cost-depends-on-cost-of-args

```
((fn ≠ 'if)
 ∧  (sum-cdrs (args1) < sum-cdrs (args2))
 ∧  (strip-cars (args1) = strip-cars (args2))
 ∧  v&c-apply$ (fn, args1)
 ∧  v&c-apply$ (fn, args2))
→   (cdr (v&c-apply$ (fn, args1)) < cdr (v&c-apply$ (fn, args2)))
```

THEOREM: iterk-v&c-apply$-idifference-exists

```
v&c-apply$ ('iterk,
             list (cons (a, 0),
                   cons (b, 0),
                   cons (c, 0),
                   cons (d, 0),
                   v&c-apply$ ('idifference, cons (cons (e, 0), '((1 . 0)))),
                   cons (x, 0)))
↔   v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       cons (idifference (e, 1), 0),
                       cons (x, 0)))
```

THEOREM: iterk-v&c-apply$-idifference-value

```
car (v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       v&c-apply$ ('idifference,
                                   cons (cons (e, 0), '((1 . 0)))),
                       cons (x, 0))))
=   car (v&c-apply$ ('iterk,
                     list (cons (a, 0),
                           cons (b, 0),
                           cons (c, 0),
                           cons (d, 0),
                           cons (idifference (e, 1), 0),
                           cons (x, 0))))
```

THEOREM: iterk-v&c-apply$-idifference-cost

```
v&c-apply$ ('iterk,
            list (cons (a, 0),
                  cons (b, 0),
                  cons (c, 0),
                  cons (d, 0),
                  cons (idifference (e, 1), 0),
                  cons (x, 0)))
→   (cdr (v&c-apply$ ('iterk,
                      list (cons (a, 0),
                            cons (b, 0),
                            cons (c, 0),
                            cons (d, 0),
                            cons (idifference (e, 1), 0),
                            cons (x, 0))))
     <   cdr (v&c-apply$ ('iterk,
                          list (cons (a, 0),
                                cons (b, 0),
                                cons (c, 0),
                                cons (d, 0),
                                v&c-apply$ ('idifference,
                                            cons (cons (e, 0), '((1 . 0)))),
                                cons (x, 0)))))
```

THEOREM: iterk-exists-iff-body-exists-when-e>1-version-2

```
ilessp (1, e)
→   (v&c-apply$ ('iterk,
                    list (cons (a, 0),
                          cons (b, 0),
                          cons (c, 0),
                          cons (d, 0),
                          cons (e, 0),
                          cons (x, 0)))
     ↔    v&c-apply$ ('iterk,
                      list (cons (a, 0),
                            cons (b, 0),
                            cons (c, 0),
                            cons (d, 0),
                            '(1 . 0),
                            v&c-apply$ ('iterk,
                                        list (cons (a, 0),
                                              cons (b, 0),
                                              cons (c, 0),
                                              cons (d, 0),
                                              cons (idifference (e, 1), 0),
                                              cons (x, 0))))))
```

THEOREM: iterk-value=body-value-when-e>1-version-2

```
 ilessp (1, e)
 →    (car (v&c-apply$ ('iterk,
                        list (cons (a, 0),
                              cons (b, 0),
                              cons (c, 0),
                              cons (d, 0),
                              cons (e, 0),
                              cons (x, 0))))
      =    car (v&c-apply$ ('iterk,
                            list (cons (a, 0),
                                  cons (b, 0),
                                  cons (c, 0),
                                  cons (d, 0),
                                  '(1 . 0),
                                  v&c-apply$ ('iterk,
                                              list (cons (a, 0),
                                                    cons (b, 0),
                                                    cons (c, 0),
                                                    cons (d, 0),
                                                    cons (idifference (e, 1), 0),
                                                    cons (x, 0)))))))
```

THEOREM: args-exist-when-fn-exists

$((fn \neq \texttt{'if}) \land \text{v\&c-apply\$}\,(fn,\ args)) \rightarrow (\mathbf{f} \notin args)$

Theorem: iterk-cost>body-cost-when-e>1-version-2

```
 (ilessp (1, e)
  ∧  v&c-apply$ ('iterk,
                      list (cons (a, 0),
                            cons (b, 0),
                            cons (c, 0),
                            cons (d, 0),
                            cons (e, 0),
                            cons (x, 0))))
 →   (cdr (v&c-apply$ ('iterk,
                          list (cons (a, 0),
                                cons (b, 0),
                                cons (c, 0),
                                cons (d, 0),
                                '(1 . 0),
                                v&c-apply$ ('iterk,
                                                list (cons (a, 0),
                                                      cons (b, 0),
                                                      cons (c, 0),
                                                      cons (d, 0),
                                                      cons (idifference (e, 1), 0),
                                                      cons (x, 0))))))
        <    cdr (v&c-apply$ ('iterk,
                                list (cons (a, 0),
                                      cons (b, 0),
                                      cons (c, 0),
                                      cons (d, 0),
                                      cons (e, 0),
                                      cons (x, 0)))))
```

Theorem: iterk-exists-iff-body-exists-when-e>1-version-3-case-1

```
 (vc-x ∧ ilessp (1, e))
 →   (v&c-apply$ ('iterk,
                     list (cons (a, 0),
                           cons (b, 0),
                           cons (c, 0),
                           cons (d, 0),
                           cons (e, 0),
                           vc-x))
        ↔    v&c-apply$ ('iterk,
                            list (cons (a, 0),
                                  cons (b, 0),
                                  cons (c, 0),
                                  cons (d, 0),
                             '(1 . 0),
                             v&c-apply$ ('iterk,
                                         list (cons (a, 0),
                                               cons (b, 0),
                                               cons (c, 0),
                                               cons (d, 0),
                                               cons (idifference (e, 1), 0),
                                               vc-x)))))
```

THEOREM: iterk-exists-iff-body-exists-when-e>1-version-3-case-2

```
((vc-x = f) ∧ ilessp (1, e))
→   (v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       cons (e, 0),
                       vc-x))
     ↔   v&c-apply$ ('iterk,
                     list (cons (a, 0),
                           cons (b, 0),
                           cons (c, 0),
                           cons (d, 0),
                           '(1 . 0),
                           v&c-apply$ ('iterk,
                                       list (cons (a, 0),
                                             cons (b, 0),
                                             cons (c, 0),
                                             cons (d, 0),
                                             cons (idifference (e, 1), 0),
                                             vc-x)))))
```

THEOREM: iterk-exists-iff-body-exists-when-e>1-version-3

```
ilessp (1, e)
→   (v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       cons (e, 0),
                       vc-x))
     ↔   v&c-apply$ ('iterk,
                     list (cons (a, 0),
                           cons (b, 0),
                           cons (c, 0),
                           cons (d, 0),
                           '(1 . 0),
                           v&c-apply$ ('iterk,
                                       list (cons (a, 0),
                                             cons (b, 0),
                                             cons (c, 0),
                                             cons (d, 0),
                                             cons (idifference (e, 1), 0),
                                             vc-x)))))
```

THEOREM: iterk-value=body-value-when-e>1-version-3-case-1

```
(vc-x ∧ ilessp (1, e))
→   (car (v&c-apply$ ('iterk,
                      list (cons (a, 0),
                            cons (b, 0),
                            cons (c, 0),
                            cons (d, 0),
                            cons (e, 0),
                            vc-x)))
     =   car (v&c-apply$ ('iterk,
                          list (cons (a, 0),
                                cons (b, 0),
                                cons (c, 0),
                                cons (d, 0),
                                '(1 . 0),
                                v&c-apply$ ('iterk,
                                            list (cons (a, 0),
                                                  cons (b, 0),
                                                  cons (c, 0),
                                                  cons (d, 0),
                                                  cons (idifference (e, 1), 0),
                                                  vc-x))))))
```

THEOREM: iterk-value=body-value-when-e>1-version-3-case-2

```
((vc-x = f) ∧ ilessp (1, e))
→   (car (v&c-apply$ ('iterk,
                      list (cons (a, 0),
                            cons (b, 0),
                            cons (c, 0),
                            cons (d, 0),
                            cons (e, 0),
                            vc-x)))
     =   car (v&c-apply$ ('iterk,
                           list (cons (a, 0),
                                 cons (b, 0),
                                 cons (c, 0),
                                 cons (d, 0),
                                 '(1 . 0),
                                 v&c-apply$ ('iterk,
                                             list (cons (a, 0),
                                                   cons (b, 0),
                                                   cons (c, 0),
                                                   cons (d, 0),
                                                   cons (idifference (e, 1), 0),
                                                   vc-x))))))
```

THEOREM: iterk-value=body-value-when-e>1-version-3

```
 ilessp (1, e)
→   (car (v&c-apply$ ('iterk,
                      list (cons (a, 0),
                            cons (b, 0),
                            cons (c, 0),
                            cons (d, 0),
                            cons (e, 0),
                            vc-x)))
      =    car (v&c-apply$ ('iterk,
                            list (cons (a, 0),
                                  cons (b, 0),
                                  cons (c, 0),
                                  cons (d, 0),
                                  '(1 . 0),
                                  v&c-apply$ ('iterk,
                                              list (cons (a, 0),
                                                    cons (b, 0),
                                                    cons (c, 0),
                                                    cons (d, 0),
                                                    cons (idifference (e, 1), 0),
                                                    vc-x))))))
```

THEOREM: iterk-exists-when-e<=1&a<x

```
 ((¬ ilessp (1, e)) ∧ ilessp (a, x))
→   v&c-apply$ ('iterk,
                list (cons (a, 0),
                      cons (b, 0),
                      cons (c, 0),
                      cons (d, 0),
                      cons (e, 0),
                      cons (x, 0)))
```

Theorem: iterk-value-when-e<=1&a<x

$$
\begin{array}{l}
((\neg\ \text{ilessp}\,(\texttt{1},\ e)) \land \text{ilessp}\,(a,\ x)) \\
\rightarrow \quad (\text{car}\,(\text{v\&c-apply\$}\,(\texttt{'iterk}, \\
\qquad\qquad\qquad\qquad \text{list}\,(\text{cons}\,(a,\ \texttt{0}), \\
\qquad\qquad\qquad\qquad\qquad \text{cons}\,(b,\ \texttt{0}), \\
\qquad\qquad\qquad\qquad\qquad \text{cons}\,(c,\ \texttt{0}), \\
\qquad\qquad\qquad\qquad\qquad \text{cons}\,(d,\ \texttt{0}), \\
\qquad\qquad\qquad\qquad\qquad \text{cons}\,(e,\ \texttt{0}), \\
\qquad\qquad\qquad\qquad\qquad \text{cons}\,(x,\ \texttt{0})))) \\
\qquad = \quad \text{idifference}\,(x,\ b))
\end{array}
$$

Theorem: iterk-exists-iff-body-exists-when-e<=1&a>=x

$$
\begin{array}{l}
((\neg\ \text{ilessp}\,(\texttt{1},\ e)) \land (\neg\ \text{ilessp}\,(a,\ x))) \\
\rightarrow \quad (\text{v\&c-apply\$}\,(\texttt{'iterk}, \\
\qquad\qquad\qquad \text{list}\,(\text{cons}\,(a,\ \texttt{0}), \\
\qquad\qquad\qquad\qquad \text{cons}\,(b,\ \texttt{0}), \\
\qquad\qquad\qquad\qquad \text{cons}\,(c,\ \texttt{0}), \\
\qquad\qquad\qquad\qquad \text{cons}\,(d,\ \texttt{0}), \\
\qquad\qquad\qquad\qquad \text{cons}\,(e,\ \texttt{0}), \\
\qquad\qquad\qquad\qquad \text{cons}\,(x,\ \texttt{0}))) \\
\qquad \leftrightarrow \quad \text{v\&c-apply\$}\,(\texttt{'iterk}, \\
\qquad\qquad\qquad\qquad \text{list}\,(\text{cons}\,(a,\ \texttt{0}), \\
\qquad\qquad\qquad\qquad\qquad \text{cons}\,(b,\ \texttt{0}), \\
\qquad\qquad\qquad\qquad\qquad \text{cons}\,(c,\ \texttt{0}), \\
\qquad\qquad\qquad\qquad\qquad \text{cons}\,(d,\ \texttt{0}), \\
\qquad\qquad\qquad\qquad\qquad \text{cons}\,(c,\ \texttt{0}), \\
\qquad\qquad\qquad\qquad\qquad \text{v\&c-apply\$}\,(\texttt{'iplus}, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{list}\,(\text{cons}\,(x,\ \texttt{0}),\ \text{cons}\,(d,\ \texttt{0}))))))
\end{array}
$$

Theorem: iterk-value=body-value-when-e<=1&a>=x

$$
\begin{array}{l}
((\neg\ \text{ilessp}\,(\texttt{1},\ e)) \land (\neg\ \text{ilessp}\,(a,\ x))) \\
\rightarrow \quad (\text{car}\,(\text{v\&c-apply\$}\,(\texttt{'iterk}, \\
\qquad\qquad\qquad\qquad \text{list}\,(\text{cons}\,(a,\ \texttt{0}), \\
\qquad\qquad\qquad\qquad\qquad \text{cons}\,(b,\ \texttt{0}), \\
\qquad\qquad\qquad\qquad\qquad \text{cons}\,(c,\ \texttt{0}), \\
\qquad\qquad\qquad\qquad\qquad \text{cons}\,(d,\ \texttt{0}), \\
\qquad\qquad\qquad\qquad\qquad \text{cons}\,(e,\ \texttt{0}), \\
\qquad\qquad\qquad\qquad\qquad \text{cons}\,(x,\ \texttt{0})))) \\
\qquad = \quad \text{car}\,(\text{v\&c-apply\$}\,(\texttt{'iterk}, \\
\qquad\qquad\qquad\qquad\qquad \text{list}\,(\text{cons}\,(a,\ \texttt{0}), \\
\qquad\qquad\qquad\qquad\qquad\qquad \text{cons}\,(b,\ \texttt{0}), \\
\qquad\qquad\qquad\qquad\qquad\qquad \text{cons}\,(c,\ \texttt{0}), \\
\qquad\qquad\qquad\qquad\qquad\qquad \text{cons}\,(d,\ \texttt{0}), \\
\qquad\qquad\qquad\qquad\qquad\qquad \text{cons}\,(c,\ \texttt{0}),
\end{array}
$$

```
                                   v&c-apply$ ('iplus,
                                               list (cons (x, 0), cons (d, 0)))))))
```

Theorem: iterk-cost>body-cost-when-e<=1&a>=x

```
((¬ ilessp (1, e))
 ∧   (¬ ilessp (a, x))
 ∧   v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       cons (e, 0),
                       cons (x, 0))))
→   (cdr (v&c-apply$ ('iterk,
                      list (cons (a, 0),
                            cons (b, 0),
                            cons (c, 0),
                            cons (d, 0),
                            cons (c, 0),
                            v&c-apply$ ('iplus, list (cons (x, 0), cons (d, 0))))))
      <    cdr (v&c-apply$ ('iterk,
                            list (cons (a, 0),
                                  cons (b, 0),
                                  cons (c, 0),
                                  cons (d, 0),
                                  cons (e, 0),
                                  cons (x, 0)))))
```

Event: Disable iterk-exists-iff-body-exists.

Event: Disable iterk-value=body-value.

Event: Disable iterk-cost>body-cost.

Theorem: iterk-exists-iff-body-exists-when-e<=1&a>=x-version-2

```
((¬ ilessp (1, e)) ∧ (¬ ilessp (a, x)))
→   (v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       cons (e, 0),
                       cons (x, 0)))
```

```
      ↔   v&c-apply$ ('iterk,
                      list (cons (a, 0),
                            cons (b, 0),
                            cons (c, 0),
                            cons (d, 0),
                            cons (c, 0),
                            cons (iplus (x, d), 0))))
```

THEOREM: iterk-value=body-value-when-e<=1&a>=x-version-2

```
 ((¬ ilessp (1, e)) ∧ (¬ ilessp (a, x)))
 →   (car (v&c-apply$ ('iterk,
                       list (cons (a, 0),
                             cons (b, 0),
                             cons (c, 0),
                             cons (d, 0),
                             cons (e, 0),
                             cons (x, 0))))
      =   car (v&c-apply$ ('iterk,
                           list (cons (a, 0),
                                 cons (b, 0),
                                 cons (c, 0),
                                 cons (d, 0),
                                 cons (c, 0),
                                 cons (iplus (x, d), 0)))))
```

THEOREM: iterk-cost>body-cost-when-e<=1&a>=x-version-2

```
 ((¬ ilessp (1, e))
  ∧   (¬ ilessp (a, x))
  ∧   v&c-apply$ ('iterk,
                  list (cons (a, 0),
                        cons (b, 0),
                        cons (c, 0),
                        cons (d, 0),
                        cons (e, 0),
                        cons (x, 0))))
 →   (cdr (v&c-apply$ ('iterk,
                       list (cons (a, 0),
                             cons (b, 0),
                             cons (c, 0),
                             cons (d, 0),
                             cons (c, 0),
                             cons (iplus (x, d), 0))))
      <   cdr (v&c-apply$ ('iterk,
                           list (cons (a, 0),
```

cons (*b*, 0),
cons (*c*, 0),
cons (*d*, 0),
cons (*e*, 0),
cons (*x*, 0)))))

EVENT: Disable iterk-exists-iff-body-exists-when-e>1.

EVENT: Disable iterk-value=body-value-when-e>1.

EVENT: Disable iterk-cost>body-cost-when-e>1.

EVENT: Disable iterk-exists-iff-body-exists-when-e<=1&a>=x.

EVENT: Disable iterk-value=body-value-when-e<=1&a>=x.

EVENT: Disable iterk-cost>body-cost-when-e<=1&a>=x.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; The next two events are versions of Part 1 of the
; Main Theorem given above in the introduction.

;  1. If  x > a, then  K( a,b,c,d,x )  exists.

;     Proof.  By the definitions of IterK and K.
```

THEOREM: k-exists-when-x>a
ilessp (*a*, *x*)
→ v&c-apply$ ('k,
list (cons (*a*, 0), cons (*b*, 0), cons (*c*, 0), cons (*d*, 0), cons (*x*, 0)))

THEOREM: k-halts-when-x>a
(*vc-a* ∧ *vc-b* ∧ *vc-c* ∧ *vc-d* ∧ *vc-x* ∧ ilessp (car (*vc-a*), car (*vc-x*)))
→ v&c-apply$ ('k, list (*vc-a*, *vc-b*, *vc-c*, *vc-d*, *vc-x*))

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Lemma 1. The cost of computing IterK is unbounded
;          when x <= a, d <= 0, and c <= 1:

;  Assume  x <= a, d <= 0, c <= 1, i > 0,
;   and
;  IterK( a,b,c,d,1,x ) exists.

;  Then IterK( a,b,c,d,c,x+id ) exists

;   and

;  cost[ IterK( a,b,c,d,c,x+id ) ] + i-1
;   <
;  cost[ IterK( a,b,c,d,1,x ) ].

; Proof. By induction on i.
```

Theorem: count-sub1-x<count-x-when-x>0&x<>1
(ilessp (0, x) $\wedge$ ($x \neq 1$)) → (count (iplus (-1, x)) $<$ count (x))

Definition:
induct-hint-positive-int (x)

```
=   if ¬ ilessp (0, x)  then t
    elseif x = 1  then t
    else induct-hint-positive-int (iplus (-1, x)) endif
```

Theorem: sub1-x>0-when-x>0&x<>1
(ilessp (0, x) $\wedge$ ($x \neq 1$)) → ilessp (0, iplus (-1, x))

Theorem: x+_i-1_d<=a-when-x<=a&d<=0&i>0
(($\neg$ ilessp (a, x)) $\wedge$ ($\neg$ ilessp (0, d)) $\wedge$ ilessp (0, i))
→ ($\neg$ ilessp (a, iplus (x, itimes (iplus (-1, i), d))))

Theorem: iterk_x+_i-1_d_implies_iterk_x+id-when-x<=a&d<=0&c<=1&i>0

```
((¬ ilessp (a, x))
 ∧  (¬ ilessp (0, d))
 ∧  (¬ ilessp (1, c))
 ∧  ilessp (0, i)
 ∧  v&c-apply$ ('iterk,
                list (cons (a, 0),
                      cons (b, 0),
                      cons (c, 0),
                      cons (d, 0),
                      cons (c, 0),
                      cons (iplus (x, itimes (iplus (-1, i), d)), 0))))
 →  v&c-apply$ ('iterk,
                list (cons (a, 0),
                      cons (b, 0),
                      cons (c, 0),
                      cons (d, 0),
                      cons (c, 0),
                      cons (iplus (x, itimes (d, i)), 0)))
```

THEOREM: iterk-e=1&x-implies-iterk-e=c&x+id-when-x<=a&d<=0&c<=1

```
((¬ ilessp (a, x))
 ∧   (¬ ilessp (0, d))
 ∧   (¬ ilessp (1, c))
 ∧   ilessp (0, i)
 ∧   v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       cons (1, 0),
                       cons (x, 0))))
→   v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       cons (c, 0),
                       cons (iplus (x, itimes (i, d)), 0)))
```

THEOREM: w-1+y<z&x<y-implies-w+x<z

```
(ilessp (0, w) ∧ (x < y) ∧ ((iplus (-1, w) + y) < z))
→   (((w + x) < z) = t)
```

THEOREM: cost-induction-step-when-x<=a&d<=0&c<=1

```
((¬ ilessp (a, x))
 ∧   (¬ ilessp (0, d))
 ∧   (¬ ilessp (1, c))
 ∧   ilessp (0, i)
 ∧   (i ≠ 1)
 ∧   v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       '(1 . 0),
                       cons (x, 0)))
 ∧   ((cdr (v&c-apply$ ('iterk,
                        list (cons (a, 0),
                              cons (b, 0),
                              cons (c, 0),
                                    cons (d, 0),
                                    cons (c, 0),
                                    cons (iplus (x, itimes (iplus (-1, i), d)), 0))))
         +   iplus (-1, iplus (-1, i)))
         <   cdr (v&c-apply$ ('iterk,
                              list (cons (a, 0),
                                    cons (b, 0),
                                    cons (c, 0),
                                    cons (d, 0),
                                    '(1 . 0),
                                    cons (x, 0))))))
 →   ((cdr (v&c-apply$ ('iterk,
                         list (cons (a, 0),
                               cons (b, 0),
                               cons (c, 0),
                               cons (d, 0),
                               cons (c, 0),
                               cons (iplus (x, itimes (d, i)), 0))))
         +   iplus (-1, i))
         <   cdr (v&c-apply$ ('iterk,
                              list (cons (a, 0),
                                    cons (b, 0),
                                    cons (c, 0),
                                    cons (d, 0),
                                    '(1 . 0),
                                    cons (x, 0)))))
```

THEOREM: cost-base-step-when-x<=a&d<=0&c<=1

```
 ((¬ ilessp (a, x))
  ∧   v&c-apply$ ('iterk,
                  list (cons (a, 0),
                        cons (b, 0),
                        cons (c, 0),
                        cons (d, 0),
                        cons (1, 0),
                        cons (x, 0))))
 →   (cdr (v&c-apply$ ('iterk,
                       list (cons (a, 0),
                             cons (b, 0),
                             cons (c, 0),
                             cons (d, 0),
                             cons (c, 0),
                             cons (iplus (d, x), 0))))
         <   cdr (v&c-apply$ ('iterk,
                        list (cons (a, 0),
                              cons (b, 0),
                              cons (c, 0),
                              cons (d, 0),
                              cons (1, 0),
                              cons (x, 0)))))
```

THEOREM: iterk-e=1&x-cost>i-1+cost-iterk-e=c&x+id-when-x<=a&d<=0&c<=1

```
((¬ ilessp (a, x))
 ∧   (¬ ilessp (0, d))
 ∧   (¬ ilessp (1, c))
 ∧   ilessp (0, i)
 ∧   v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       '(1 . 0),
                       cons (x, 0))))
→   ((cdr (v&c-apply$ ('iterk,
                       list (cons (a, 0),
                             cons (b, 0),
                             cons (c, 0),
                             cons (d, 0),
                             cons (c, 0),
                             cons (iplus (x, itimes (i, d)), 0))))
      +   iplus (-1, i))
     <   cdr (v&c-apply$ ('iterk,
                          list (cons (a, 0),
                                cons (b, 0),
                                cons (c, 0),
                                cons (d, 0),
                                '(1 . 0),
                                cons (x, 0)))))
```

THEOREM: y<=z-when-x+y-1<z&x>=0&y>0

```
((x ∈ N) ∧ ilessp (0, y) ∧ ((x + iplus (-1, y)) < z))
→   ((z ≮ y) = t)
```

THEOREM: iterk-cost-is-unbounded-when-x<=a&d<=0&c<=1

```
((¬ ilessp (a, x))
 ∧   (¬ ilessp (0, d))
 ∧   (¬ ilessp (1, c))
 ∧   ilessp (0, i)
 ∧   v&c-apply$ ('iterk,
                    list (cons (a, 0),
                          cons (b, 0),
                          cons (c, 0),
                          cons (d, 0),
                          '(1 . 0),
                          cons (x, 0))))
 →    (cdr (v&c-apply$ ('iterk,
                         list (cons (a, 0),
                               cons (b, 0),
                               cons (c, 0),
                               cons (d, 0),
                               '(1 . 0),
                               cons (x, 0))))
        ≮    i)
```

Theorem: ilessp-0-add1-x

```
 ilessp (0, 1 + x)
```

Theorem: iterk-does-not-exist-when-x<=a&d<=0&c<=1

```
 ((¬ ilessp (a, x)) ∧ (¬ ilessp (0, d)) ∧ (¬ ilessp (1, c)))
 →    (¬ v&c-apply$ ('iterk,
                      list (cons (a, 0),
                            cons (b, 0),
                            cons (c, 0),
                            cons (d, 0),
                            '(1 . 0),
                            cons (x, 0))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; The next two events are versions of Part 2 of the
; Main Theorem given above in the introduction.

;  2. If  x <= a, d <= 0, and c <= 1, then
;     K( a,b,c,d,x )  does not exist.

;     Proof. By Lemma 1 and the definition of K.
```

Theorem: k-does-not-exist-when-x<=a&d<=0&c<=1

```
 ((¬ ilessp (a, x)) ∧ (¬ ilessp (0, d)) ∧ (¬ ilessp (1, c)))
 →    (¬ v&c-apply$ ('k,
                      list (cons (a, 0),
                            cons (b, 0),
                            cons (c, 0),
                            cons (d, 0),
                            cons (x, 0))))
```

Theorem: k-does-not-halt-when-x<=a&d<=0&c<=1

$$\begin{array}{ll}
(\mathit{vc\text{-}a} & \\
\ \land & \mathit{vc\text{-}b} \\
\ \land & \mathit{vc\text{-}c} \\
\ \land & \mathit{vc\text{-}d} \\
\ \land & \mathit{vc\text{-}x} \\
\ \land & (\neg\ \text{ilessp}\,(\text{car}\,(\mathit{vc\text{-}a}),\ \text{car}\,(\mathit{vc\text{-}x}))) \\
\ \land & (\neg\ \text{ilessp}\,(\texttt{0},\ \text{car}\,(\mathit{vc\text{-}d}))) \\
\ \land & (\neg\ \text{ilessp}\,(\texttt{1},\ \text{car}\,(\mathit{vc\text{-}c})))) \\
\to & (\neg\ \text{v\&c-apply\$}\,(\texttt{'k},\ \text{list}\,(\mathit{vc\text{-}a},\ \mathit{vc\text{-}b},\ \mathit{vc\text{-}c},\ \mathit{vc\text{-}d},\ \mathit{vc\text{-}x})))
\end{array}$$

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Lemma 2. When the number of iterates is positive,
;          the cost of computing IterK is an order
;          homomorhism of the number of iterates:

;  Assume that  0 < e1 < e2
;   and
;  IterK( a,b,c,d,e2,x ) exists.

;  Then  IterK( a,b,c,d,e1,x ) also exists

;   and

;  cost[ IterK( a,b,c,d,e1,x ) ]
;   <
;  cost[ IterK( a,b,c,d,e2,x ) ].

; Proof.  Hold e1 constant and induct on e2.
```

Theorem: count-sub1-y<count-y-when-0<x<y

$(\text{ilessp}\,(\texttt{0},\ x) \land \text{ilessp}\,(x,\ y)) \to (\text{count}\,(\text{iplus}\,(\texttt{-1},\ y)) < \text{count}\,(y))$

Definition:

induct-hint-cost-hom $(x,\ y)$

$=$ **if** $\neg$ ilessp $(\texttt{0},\ x)$ **then t**

elseif $\neg$ ilessp $(x,\ y)$ **then t**

elseif $x =$ iplus $(\texttt{-1},\ y)$ **then t**

else induct-hint-cost-hom $(x,\ \text{iplus}\,(\texttt{-1},\ y))$ **endif**

Theorem: y>1-when-y>x&x>0

$(\text{ilessp}\,(\texttt{0},\ x) \land \text{ilessp}\,(x,\ y)) \to \text{ilessp}\,(\texttt{1},\ y)$

Theorem: iterk-at-e1-exists-when-iterk-at-e2-exists&0<e1<e2-basic-step

```
(ilessp (0, e1)
 ∧   ilessp (e1, e2)
 ∧   v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       cons (e2, 0),
                       cons (x, 0))))
→   v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       cons (iplus (-1, e2), 0),
                       cons (x, 0)))
```

Theorem: x<y-1-when-0<x&x<y&x<>y-1

```
(ilessp (0, x) ∧ ilessp (x, y) ∧ (x ≠ iplus (-1, y)))
→   ilessp (x, iplus (-1, y))
```

Theorem: iterk-at-e1-exists-when-iterk-at-e2-exists&0<e1<e2

```
(ilessp (0, e1)
 ∧   ilessp (e1, e2)
 ∧   v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       cons (e2, 0),
                       cons (x, 0))))
→   v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       cons (e1, 0),
                       cons (x, 0)))
```

Theorem: cost>cost-of-args

```
((fn ≠ 'if) ∧ (f ∉ args) ∧ v&c-apply$ (fn, args))
→   (sum-cdrs (args) < cdr (v&c-apply$ (fn, args)))
```

Theorem: sum-cdrs>=member

```
(x ∈ l) → (sum-cdrs (l) ≮ cdr (x))
```

THEOREM: cost>member

```
(v&c-apply$ (fn, args) ∧ (fn ≠ 'if) ∧ (x ∈ args))
→  (cdr (x) < cdr (v&c-apply$ (fn, args)))
```

THEOREM: cost-iterk-at-e1<cost-iterk-at-e2-when-0<e1<e2-basic-step

```
(ilessp (0, e1)
 ∧  ilessp (e1, e2)
 ∧  v&c-apply$ ('iterk,
                list (cons (a, 0),
                      cons (b, 0),
                      cons (c, 0),
                      cons (d, 0),
                      cons (e2, 0),
                      cons (x, 0))))
→  (cdr (v&c-apply$ ('iterk,
                     list (cons (a, 0),
                           cons (b, 0),
                           cons (c, 0),
                           cons (d, 0),
                           cons (iplus (-1, e2), 0),
                           cons (x, 0))))
      <  cdr (v&c-apply$ ('iterk,
                          list (cons (a, 0),
                                cons (b, 0),
                                cons (c, 0),
                                cons (d, 0),
                                cons (e2, 0),
                                cons (x, 0)))))
```

EVENT: Disable y>1-when-y>x&x>0.

THEOREM: cost-iterk-at-e1<cost-iterk-at-e2-when-0<e1<e2

```
(ilessp (0, e1)
 ∧  ilessp (e1, e2)
 ∧  v&c-apply$ ('iterk,
                list (cons (a, 0),
                      cons (b, 0),
                      cons (c, 0),
                      cons (d, 0),
                      cons (e2, 0),
                      cons (x, 0))))
→  (cdr (v&c-apply$ ('iterk,
                     list (cons (a, 0),
                           cons (b, 0),
                           cons (c, 0),
                           cons (d, 0),
                           cons (e1, 0),
                           cons (x, 0))))
         <   cdr (v&c-apply$ ('iterk,
                           list (cons (a, 0),
                                 cons (b, 0),
                                 cons (c, 0),
                                 cons (d, 0),
                                 cons (e2, 0),
                                 cons (x, 0)))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Lemma 3. The cost of computing IterK is unbounded
;          when x <= a, d <= 0, and c > 1:

;  Assume  x <= a, d <= 0, c > 1, i > 0,
;   and
;  IterK( a,b,c,d,1,x ) exists.

;  Then IterK( a,b,c,d,1,x+id ) exists

;   and

;  cost[ IterK( a,b,c,d,1,x+id ) ] + i
;   <
;  cost[ IterK( a,b,c,d,1,x ) ].

; Proof. By induction on i.
```

Theorem: iterk_x+_i-1_d_implies_iterk_x+id-when-x<=a&d<=0&c>1&i>0&e=1

```
((¬ ilessp (a, x))
 ∧  (¬ ilessp (0, d))
 ∧  ilessp (1, c)
 ∧  ilessp (0, i)
 ∧  v&c-apply$ ('iterk,
               list (cons (a, 0),
                     cons (b, 0),
                     cons (c, 0),
                     cons (d, 0),
                     '(1 . 0),
                     cons (iplus (x, itimes (iplus (-1, i), d)), 0))))
→   v&c-apply$ ('iterk,
               list (cons (a, 0),
                     cons (b, 0),
                     cons (c, 0),
                     cons (d, 0),
                     '(1 . 0),
                     cons (iplus (x, itimes (d, i)), 0)))
```

Theorem: iterk-x-implies-iterk-x+id-when-x<=a&d<=0&c>1&e=1

```
 ((¬ ilessp (a, x))
  ∧    (¬ ilessp (0, d))
  ∧    ilessp (1, c)
  ∧    ilessp (0, i)
  ∧    v&c-apply$ ('iterk,
                        list (cons (a, 0),
                              cons (b, 0),
                              cons (c, 0),
                              cons (d, 0),
                              '(1 . 0),
                              cons (x, 0))))
 →    v&c-apply$ ('iterk,
                        list (cons (a, 0),
                              cons (b, 0),
                              cons (c, 0),
                              cons (d, 0),
                              '(1 . 0),
                              cons (iplus (x, itimes (i, d)), 0)))
```

Theorem: iterk-x+d-cost<cost-iterk-x-when-a>=x&1<c

```
 ((¬ ilessp (a, x))
  ∧    ilessp (1, c)
  ∧    v&c-apply$ ('iterk,
                        list (cons (a, 0),
                              cons (b, 0),
                              cons (c, 0),
                              cons (d, 0),
                              '(1 . 0),
                              cons (x, 0))))
 →    (cdr (v&c-apply$ ('iterk,
                               list (cons (a, 0),
                                     cons (b, 0),
                                     cons (c, 0),
                                     cons (d, 0),
                                     '(1 . 0),
                                     cons (iplus (x, d), 0))))
```

```
       <   cdr (v&c-apply$ ('iterk,
                             list (cons (a, 0),
                                   cons (b, 0),
                                   cons (c, 0),
                                   cons (d, 0),
                                   '(1 . 0),
                                   cons (x, 0)))))
```

Theorem: cost-induction-step-when-x<=a&d<=0&c>1

```
((¬ ilessp (a, x))
 ∧  (¬ ilessp (0, d))
 ∧  ilessp (1, c)
 ∧  ilessp (0, i)
 ∧  (i ≠ 1)
 ∧  v&c-apply$ ('iterk,
                list (cons (a, 0),
                      cons (b, 0),
                      cons (c, 0),
                      cons (d, 0),
                      '(1 . 0),
                      cons (x, 0)))
 ∧  ((cdr (v&c-apply$ ('iterk,
                        list (cons (a, 0),
                              cons (b, 0),
                              cons (c, 0),
                              cons (d, 0),
                              '(1 . 0),
                              cons (iplus (x, itimes (iplus (-1, i), d)), 0))))
      +   iplus (-1, i))
      <   cdr (v&c-apply$ ('iterk,
                            list (cons (a, 0),
                                  cons (b, 0),
                                  cons (c, 0),
                                  cons (d, 0),
                                  '(1 . 0),
                                  cons (x, 0))))))
 →  ((cdr (v&c-apply$ ('iterk,
                        list (cons (a, 0),
                              cons (b, 0),
                              cons (c, 0),
                              cons (d, 0),
                              '(1 . 0),
                              cons (iplus (x, itimes (d, i)), 0))))
      +   i)
```

```
        <    cdr (v&c-apply$ ('iterk,
                              list (cons (a, 0),
                                    cons (b, 0),
                                    cons (c, 0),
                                    cons (d, 0),
                                    '(1 . 0),
                                    cons (x, 0)))))
```

THEOREM: iterk-x-cost>i+cost-iterk-x+id-when-x<=a&d<=0&c>1&e=1

```
 ((¬ ilessp (a, x))
  ∧   (¬ ilessp (0, d))
  ∧   ilessp (1, c)
  ∧   ilessp (0, i)
  ∧   v&c-apply$ ('iterk,
                  list (cons (a, 0),
                        cons (b, 0),
                        cons (c, 0),
                        cons (d, 0),
                        '(1 . 0),
                        cons (x, 0))))
 →    ((cdr (v&c-apply$ ('iterk,
                          list (cons (a, 0),
                                cons (b, 0),
                                cons (c, 0),
                                cons (d, 0),
                                '(1 . 0),
                                cons (iplus (x, itimes (i, d)), 0))))
          +    i)
        <    cdr (v&c-apply$ ('iterk,
                              list (cons (a, 0),
                                    cons (b, 0),
                                    cons (c, 0),
                                    cons (d, 0),
                                    '(1 . 0),
                                    cons (x, 0)))))
```

THEOREM: iterk-cost-is-unbounded-when-x<=a&d<=0&c>1

```
 ((¬ ilessp (a, x))
  ∧   (¬ ilessp (0, d))
  ∧   ilessp (1, c)
  ∧   ilessp (0, i)
  ∧   v&c-apply$ ('iterk,
                  list (cons (a, 0),
                        cons (b, 0),
                              cons (c, 0),
                              cons (d, 0),
                              '(1 . 0),
                              cons (x, 0))))
 →   (i < cdr (v&c-apply$ ('iterk,
                           list (cons (a, 0),
                                 cons (b, 0),
                                 cons (c, 0),
                                 cons (d, 0),
                                 '(1 . 0),
                                 cons (x, 0)))))
```

THEOREM: iterk-does-not-exist-when-x<=a&d<=0&c>1

```
 ((¬ ilessp (a, x)) ∧ (¬ ilessp (0, d)) ∧ ilessp (1, c))
 →   (¬ v&c-apply$ ('iterk,
                    list (cons (a, 0),
                          cons (b, 0),
                          cons (c, 0),
                          cons (d, 0),
                          '(1 . 0),
                          cons (x, 0))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; The next two events are versions of Part 3 of the
; Main Theorem given above in the introduction.

;  3. If  x <= a, d <= 0, and c > 1, then
;     K( a,b,c,d,x )  does not exist.

;     Proof. By Lemma 3 and the definition of K.
```

THEOREM: k-does-not-exist-when-x<=a&d<=0&c>1

```
 ((¬ ilessp (a, x)) ∧ (¬ ilessp (0, d)) ∧ ilessp (1, c))
 →   (¬ v&c-apply$ ('k,
                    list (cons (a, 0),
                          cons (b, 0),
                          cons (c, 0),
                          cons (d, 0),
                          cons (x, 0))))
```

THEOREM: k-does-not-halt-when-x<=a&d<=0&c>1

```
 (vc-a
  ∧   vc-b
  ∧   vc-c
```

∧ *vc-d*
∧ *vc-x*
∧ (¬ ilessp (car (*vc-a*), car (*vc-x*)))
∧ (¬ ilessp (0, car (*vc-d*)))
∧ ilessp (1, car (*vc-c*)))
→ (¬ v&c-apply$ ('k, list (*vc-a*, *vc-b*, *vc-c*, *vc-d*, *vc-x*)))

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Lemma 4. IterK exists when d > 0,
;          c <= 1, and e = c:

;  Assume  d > 0, and c <= 1.

;  Then IterK( a,b,c,d,c,x ) exists.

; Proof. Hold the parameter a fixed and
;        induct on the value given by

;        if  x > a  then  0
;                   else  1 + a - x.
```

Definition:
k-measure (*a*, *x*)
= **if** ilessp (*a*, *x*) **then** 0
else iplus (1, iplus (*a*, ineg (*x*))) **endif**

Theorem: k-measure_x+d<k-measure_x-when-0<d&a>=x
(ilessp (0, *d*) ∧ (¬ ilessp (*a*, *x*)))
→ (k-measure (*a*, iplus (*d*, *x*)) < k-measure (*a*, *x*))

Event: Disable k-measure.

Definition:
induct-hint-k-measure (*a*, *b*, *c*, *d*, *x*)
= **if** ¬ ilessp (0, *d*) **then t**
elseif ilessp (*a*, *x*) **then t**
else induct-hint-k-measure (*a*, *b*, *c*, *d*, iplus (*d*, *x*)) **endif**

Theorem: iterk-exists-when-d>0&c<=1&e=c
(ilessp (0, *d*) ∧ (¬ ilessp (1, *c*)))
→ v&c-apply$ ('iterk,
list (cons (*a*, 0),
cons (*b*, 0),

```
                    cons (c, 0),
                    cons (d, 0),
                    cons (c, 0),
                    cons (x, 0)))
```

THEOREM: iterk-exists-when-x<=a&d>0&c<=1&e=1

```
((¬ ilessp (a, x)) ∧ ilessp (0, d) ∧ (¬ ilessp (1, c)))
→   v&c-apply$ ('iterk,
                list (cons (a, 0),
                      cons (b, 0),
                      cons (c, 0),
                      cons (d, 0),
                      '(1 . 0),
                      cons (x, 0)))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; The next two events are versions of Part 4 of the
; Main Theorem given above in the introduction.

;  4. If  x <= a, d > 0, and c <= 1, then
;      K( a,b,c,d,x )  exists.

;     Proof. By Lemma 4 and the definition of K.
```

THEOREM: k-exists-when-x<=a&d>0&c<=1

```
((¬ ilessp (a, x)) ∧ ilessp (0, d) ∧ (¬ ilessp (1, c)))
→   v&c-apply$ ('k,
                list (cons (a, 0), cons (b, 0), cons (c, 0), cons (d, 0), cons (x, 0)))
```

THEOREM: k-halts-when-x<=a&d>0&c<=1

```
(vc-a
 ∧   vc-b
 ∧   vc-c
 ∧   vc-d
 ∧   vc-x
 ∧   (¬ ilessp (car (vc-a), car (vc-x)))
 ∧   ilessp (0, car (vc-d))
 ∧   (¬ ilessp (1, car (vc-c))))
→   v&c-apply$ ('k, list (vc-a, vc-b, vc-c, vc-d, vc-x))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Lemma 5. IterK exists and has value > a
;          when x > a, b <= 0, and e > 0:
```

```
;  Assume  x > a, b <= 0, and e > 0.

;  Then IterK( a,b,c,d,e,x ) exists

;   and

;  IterK( a,b,c,d,e,x ) > a.

; Proof.  By induction on e.
```

Theorem: iterk-exists-when-e<=1&a<x-version-2

$$\begin{array}{l}
(\mathit{vc\text{-}x} \land (\lnot\ \text{ilessp}\,(\mathtt{1},\ e)) \land \text{ilessp}\,(a,\ \text{car}\,(\mathit{vc\text{-}x})))\\
\to\quad \text{v\&c-apply\$}\,(\mathtt{'iterk},\\
\qquad\qquad\qquad\quad \text{list}\,(\text{cons}\,(a,\ \mathtt{0}),\\
\qquad\qquad\qquad\qquad\quad \text{cons}\,(b,\ \mathtt{0}),\\
\qquad\qquad\qquad\qquad\quad \text{cons}\,(c,\ \mathtt{0}),\\
\qquad\qquad\qquad\qquad\quad \text{cons}\,(d,\ \mathtt{0}),\\
\qquad\qquad\qquad\qquad\quad \text{cons}\,(e,\ \mathtt{0}),\\
\qquad\qquad\qquad\qquad\quad \mathit{vc\text{-}x}))
\end{array}$$

Theorem: iterk-value-when-e<=1&a<x-version-2

$$\begin{array}{l}
(\mathit{vc\text{-}x} \land (\lnot\ \text{ilessp}\,(\mathtt{1},\ e)) \land \text{ilessp}\,(a,\ \text{car}\,(\mathit{vc\text{-}x})))\\
\to\quad (\text{car}\,(\text{v\&c-apply\$}\,(\mathtt{'iterk},\\
\qquad\qquad\qquad\qquad\quad \text{list}\,(\text{cons}\,(a,\ \mathtt{0}),\\
\qquad\qquad\qquad\qquad\qquad\quad \text{cons}\,(b,\ \mathtt{0}),\\
\qquad\qquad\qquad\qquad\qquad\quad \text{cons}\,(c,\ \mathtt{0}),\\
\qquad\qquad\qquad\qquad\qquad\quad \text{cons}\,(d,\ \mathtt{0}),\\
\qquad\qquad\qquad\qquad\qquad\quad \text{cons}\,(e,\ \mathtt{0}),\\
\qquad\qquad\qquad\qquad\qquad\quad \mathit{vc\text{-}x})))\\
\qquad\quad =\quad \text{idifference}\,(\text{car}\,(\mathit{vc\text{-}x}),\ b))
\end{array}$$

Theorem: a+b<x-when-a<x&b<=0

$$(\text{ilessp}\,(a,\ x) \land (\lnot\ \text{ilessp}\,(\mathtt{0},\ b))) \to \text{ilessp}\,(\text{iplus}\,(a,\ b),\ x)$$

Theorem: a<-b+x-when-a<x&b<=0

$$(\text{ilessp}\,(a,\ x) \land (\lnot\ \text{ilessp}\,(\mathtt{0},\ b))) \to \text{ilessp}\,(a,\ \text{iplus}\,(\text{ineg}\,(b),\ x))$$

Theorem: x>1-when-x>0&x<>1

$$(\text{ilessp}\,(\mathtt{0},\ x) \land (x \neq \mathtt{1})) \to \text{ilessp}\,(\mathtt{1},\ x)$$

Theorem: iterk-exists&iterk>a-when-x>a&b<=0&e>0

$$\begin{array}{l}
(\text{ilessp}\,(a,\ x) \land (\lnot\ \text{ilessp}\,(\mathtt{0},\ b)) \land \text{ilessp}\,(\mathtt{0},\ e))\\
\to\quad (\text{v\&c-apply\$}\,(\mathtt{'iterk},\\
\qquad\qquad\qquad\quad \text{list}\,(\text{cons}\,(a,\ \mathtt{0}),
\end{array}$$

```
                 cons (b, 0),
                 cons (c, 0),
                 cons (d, 0),
                 cons (e, 0),
                 cons (x, 0)))
   ∧  ilessp (a,
             car (v&c-apply$ ('iterk,
                              list (cons (a, 0),
                                    cons (b, 0),
                                    cons (c, 0),
                                    cons (d, 0),
                                    cons (e, 0),
                                    cons (x, 0))))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Lemma 6. The number of iterates in the composition
;          of IterK with itself add up:

;  Let  e1 > 0  and  e2 > 0.  Then

;  IterK( a,b,c,d,e1,IterK( a,b,c,d,e2,x ) ) exists
;   iff
;  IterK( a,b,c,d,e1+e2,x ) exists

;  and

;  IterK( a,b,c,d,e1,IterK( a,b,c,d,e2,x ) )
;   =
;  IterK( a,b,c,d,e1+e2,x )

; Proof.  By induction on e1.
```

Theorem: iplus_1_-1_x=x_when_x>0
ilessp (0, x) → (iplus (1, iplus (-1, x)) = x)

Theorem: x+y>1-when-x>0&x<>1&0<y
(ilessp (0, x) ∧ ($x \neq 1$) ∧ ilessp (0, y)) → ilessp (1, iplus (x, y))

Theorem: nbr-of-iterates-sum-exists-step-1

```
 (ilessp (0, e1) ∧ (e1 ≠ 1))
 →   (v&c-apply$ ('iterk,
                  list (cons (a, 0),
                        cons (b, 0),
                        cons (c, 0),
                        cons (d, 0),
                        cons (e1, 0),
                        v&c-apply$ ('iterk,
                                    list (cons (a, 0),
                                         cons (b, 0),
                                         cons (c, 0),
                                         cons (d, 0),
                                         cons (e2, 0),
                                         cons (x, 0)))))
      ↔   v&c-apply$ ('iterk,
                        list (cons (a, 0),
                            cons (b, 0),
                            cons (c, 0),
                            cons (d, 0),
                            '(1 . 0),
                            v&c-apply$ ('iterk,
                                         list (cons (a, 0),
                                             cons (b, 0),
                                             cons (c, 0),
                                             cons (d, 0),
                                             cons (iplus (-1, e1), 0),
                                             v&c-apply$ ('iterk,
                                                          list (cons (a, 0),
                                                              cons (b, 0),
                                                              cons (c, 0),
                                                              cons (d, 0),
                                                              cons (e2, 0),
                                                              cons (x, 0))))))))

THEOREM: nbr-of-iterates-sum-values-step-1
 (ilessp (0, e1) ∧ (e1 ≠ 1))
 →    (car (v&c-apply$ ('iterk,
                        list (cons (a, 0),
                            cons (b, 0),
                            cons (c, 0),
                            cons (d, 0),
                            cons (e1, 0),
                            v&c-apply$ ('iterk,
                                         list (cons (a, 0),
                                             cons (b, 0),
                                             cons (c, 0),
                                             cons (d, 0),
                                             cons (e2, 0),
                                             cons (x, 0))))))
```

```
=   car (v&c-apply$ ('iterk,
                     list (cons (a, 0),
                           cons (b, 0),
                           cons (c, 0),
                           cons (d, 0),
                           '(1 . 0),
                           v&c-apply$ ('iterk,
                                       list (cons (a, 0),
                                             cons (b, 0),
                                             cons (c, 0),
                                             cons (d, 0),
                                             cons (iplus (-1, e1), 0),
                                             v&c-apply$ ('iterk,
                                                         list (cons (a, 0),
                                                               cons (b, 0),
                                                               cons (c, 0),
                                                               cons (d, 0),
                                                               cons (e2,
                                                                     0),
                                                               cons (x, 0)))))))))
```

THEOREM: nbr-of-iterates-sum-exists-step-2

```
 (ilessp (0, e1) ∧ (e1 ≠ 1) ∧ ilessp (0, e2))
 →   (v&c-apply$ ('iterk,
                  list (cons (a, 0),
                        cons (b, 0),
                        cons (c, 0),
                        cons (d, 0),
                        cons (iplus (e1, e2), 0),
                        cons (x, 0)))
     ↔   v&c-apply$ ('iterk,
                     list (cons (a, 0),
                           cons (b, 0),
                           cons (c, 0),
                           cons (d, 0),
                           '(1 . 0),
                           v&c-apply$ ('iterk,
                                       list (cons (a, 0),
                                             cons (b, 0),
                                             cons (c, 0),
                                             cons (d, 0),
                                             cons (iplus (iplus (-1, e1), e2),
                                                   0),
                                             cons (x, 0))))))
```

THEOREM: nbr-of-iterates-sum-values-step-2

```
(ilessp (0, e1) ∧ (e1 ≠ 1) ∧ ilessp (0, e2))
→   (car (v&c-apply$ ('iterk,
                      list (cons (a, 0),
                            cons (b, 0),
                            cons (c, 0),
                            cons (d, 0),
                            cons (iplus (e1, e2), 0),
                            cons (x, 0))))
      =   car (v&c-apply$ ('iterk,
                           list (cons (a, 0),
                                 cons (b, 0),
                                 cons (c, 0),
                                 cons (d, 0),
                                 '(1 . 0),
                                 v&c-apply$ ('iterk,
                                             list (cons (a, 0),
                                                   cons (b, 0),
                                                   cons (c, 0),
                                                   cons (d, 0),
                                                   cons (iplus (iplus (-1, e1),
                                                                e2),
                                                         0),
                                                   cons (x, 0)))))))
```

THEOREM: nbr-of-iterates-sum-exists-step-3

```
((v&c-apply$ ('iterk,
              list (cons (a, 0),
                    cons (b, 0),
                    cons (c, 0),
                    cons (d, 0),
                    cons (iplus (-1, e1), 0),
                    v&c-apply$ ('iterk,
                                list (cons (a, 0),
                                      cons (b, 0),
                                      cons (c, 0),
                                      cons (d, 0),
                                      cons (e2, 0),
                                      cons (x, 0)))))
  ↔   v&c-apply$ ('iterk,
                  list (cons (a, 0),
                        cons (b, 0),
                        cons (c, 0),
                        cons (d, 0),
                    cons (iplus (iplus (-1, e1), e2), 0),
                    cons (x, 0))))
∧   (car (v&c-apply$ ('iterk,
                   list (cons (a, 0),
                         cons (b, 0),
                         cons (c, 0),
                         cons (d, 0),
                         cons (iplus (-1, e1), 0),
                         v&c-apply$ ('iterk,
                                     list (cons (a, 0),
                                           cons (b, 0),
                                           cons (c, 0),
                                           cons (d, 0),
                                           cons (e2, 0),
                                           cons (x, 0))))))
       =   car (v&c-apply$ ('iterk,
                          list (cons (a, 0),
                                cons (b, 0),
                                cons (c, 0),
                                cons (d, 0),
                                cons (iplus (iplus (-1, e1), e2), 0),
                                cons (x, 0))))))
→   (v&c-apply$ ('iterk,
               list (cons (a, 0),
                     cons (b, 0),
                     cons (c, 0),
                     cons (d, 0),
                     '(1 . 0),
                     v&c-apply$ ('iterk,
                                 list (cons (a, 0),
                                       cons (b, 0),
                                       cons (c, 0),
                                       cons (d, 0),
                                       cons (iplus (-1, e1), 0),
                                       v&c-apply$ ('iterk,
                                                   list (cons (a, 0),
                                                         cons (b, 0),
                                                         cons (c, 0),
                                                         cons (d, 0),
                                                         cons (e2, 0),
                                                         cons (x, 0)))))))
       ↔   v&c-apply$ ('iterk,
                      list (cons (a, 0),
                            cons (b, 0),
                                 cons (c, 0),
                                 cons (d, 0),
                                 '(1 . 0),
                                 v&c-apply$ ('iterk,
                                                list (cons (a, 0),
                                                      cons (b, 0),
                                                      cons (c, 0),
                                                      cons (d, 0),
                                                      cons (iplus (-1, iplus (e1, e2)),
                                                            0),
                                                      cons (x, 0))))))
```

THEOREM: nbr-of-iterates-sum-values-step-3

```
 ((v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       cons (iplus (-1, e1), 0),
                       v&c-apply$ ('iterk,
                                      list (cons (a, 0),
                                            cons (b, 0),
                                            cons (c, 0),
                                            cons (d, 0),
                                            cons (e2, 0),
                                            cons (x, 0)))))
  ↔    v&c-apply$ ('iterk,
                       list (cons (a, 0),
                            cons (b, 0),
                            cons (c, 0),
                            cons (d, 0),
                            cons (iplus (iplus (-1, e1), e2), 0),
                            cons (x, 0))))
 ∧    (car (v&c-apply$ ('iterk,
                           list (cons (a, 0),
                                 cons (b, 0),
                                 cons (c, 0),
                                 cons (d, 0),
                                 cons (iplus (-1, e1), 0),
                                 v&c-apply$ ('iterk,
                                                list (cons (a, 0),
                                                     cons (b, 0),
                                                     cons (c, 0),
                                                     cons (d, 0),
                                                     cons (e2, 0),
                                                     cons (x, 0))))))
       =    car (v&c-apply$ ('iterk,
                                 list (cons (a, 0),
                                      cons (b, 0),
                                      cons (c, 0),
                                      cons (d, 0),
                                      cons (iplus (iplus (-1, e1), e2), 0),
                                      cons (x, 0))))))
→     (car (v&c-apply$ ('iterk,
                            list (cons (a, 0),
                                 cons (b, 0),
                                 cons (c, 0),
                                 cons (d, 0),
                                 '(1 . 0),
                                 v&c-apply$ ('iterk,
                                                list (cons (a, 0),
                                                     cons (b, 0),
                                                     cons (c, 0),
                                                     cons (d, 0),
                                                     cons (iplus (-1, e1), 0),
                                                     v&c-apply$ ('iterk,
                                                                    list (cons (a, 0),
                                                                         cons (b, 0),
                                                                         cons (c, 0),
                                                                         cons (d, 0),
                                                                         cons (e2, 0),
                                                                         cons (x, 0))))))))
       =    car (v&c-apply$ ('iterk,
                                 list (cons (a, 0),
                                      cons (b, 0),
                                      cons (c, 0),
                                      cons (d, 0),
                                      '(1 . 0),
                                      v&c-apply$ ('iterk,
                                                     list (cons (a, 0),
                                                          cons (b, 0),
                                                          cons (c, 0),
                                                          cons (d, 0),
                                                          cons (iplus (-1,
                                                                        iplus (e1, e2)),
                                                                 0),
                                                          cons (x, 0)))))))
```

Theorem: nbr-of-iterates-sum-exists&values

```
 (ilessp (0, e1) ∧ ilessp (0, e2))
 →   ((v&c-apply$ ('iterk,
                   list (cons (a, 0),
                         cons (b, 0),
                         cons (c, 0),
                         cons (d, 0),
                         cons (e1, 0),
                         v&c-apply$ ('iterk,
                                     list (cons (a, 0),
                                           cons (b, 0),
                                           cons (c, 0),
                                           cons (d, 0),
                                           cons (e2, 0),
                                           cons (x, 0)))))
       ↔   v&c-apply$ ('iterk,
                       list (cons (a, 0),
                             cons (b, 0),
                             cons (c, 0),
                             cons (d, 0),
                             cons (iplus (e1, e2), 0),
                             cons (x, 0))))
      ∧   (car (v&c-apply$ ('iterk,
                            list (cons (a, 0),
                                  cons (b, 0),
                                  cons (c, 0),
                                  cons (d, 0),
                                  cons (e1, 0),
                                  v&c-apply$ ('iterk,
                                              list (cons (a, 0),
                                                    cons (b, 0),
                                                    cons (c, 0),
                                                    cons (d, 0),
                                                    cons (e2, 0),
                                                    cons (x, 0))))))
           =   car (v&c-apply$ ('iterk,
                                list (cons (a, 0),
                                      cons (b, 0),
                                      cons (c, 0),
                                      cons (d, 0),
                                      cons (iplus (e1, e2), 0),
                                      cons (x, 0))))))
```

;;;

```
; Lemma 7. The number of iterates of K accumulate
;          when 1 < c, 0 < d, and x is "small."

;  Assume 1 < c, 0 < d, 0 < n, and a+d >= x+nd.

;  Then

;  IterK( a,b,c,d,1,x ) exists
;   iff
;  IterK( a,b,c,d,1+n(c-1),x+nd ) exists

;  and

;  IterK( a,b,c,d,1,x )
;   =
;  IterK( a,b,c,d,1+n(c-1),x+nd ).

; Proof.  By induction on n.
```

THEOREM: n_c-1_+1=c-when-n=1&c>1

$$((n = \mathtt{1}) \land \text{ilessp}\,(\mathtt{1},\ c)) \rightarrow (\text{iplus}\,(\mathtt{1},\ \text{itimes}\,(n,\ \text{iplus}\,(\mathtt{-1},\ c))) = c)$$

THEOREM: nd=d-when-n=1&d>0

$$((n = \mathtt{1}) \land \text{ilessp}\,(\mathtt{0},\ d)) \rightarrow (\text{itimes}\,(n,\ d) = d)$$

THEOREM: a>=x-when-a+d>=x+d

$$(\neg\ \text{ilessp}\,(\text{iplus}\,(a,\ d),\ \text{iplus}\,(x,\ d))) \rightarrow (\neg\ \text{ilessp}\,(a,\ x))$$

THEOREM: iterk_e=1+_n_c-1&x+nd-exists-base-step

$$\begin{array}{ll}
((n = \mathtt{1}) & \\
\ \land & \text{ilessp}\,(\mathtt{1},\ c) \\
\ \land & \text{ilessp}\,(\mathtt{0},\ d) \\
\ \land & (\neg\ \text{ilessp}\,(\text{iplus}\,(a,\ d),\ \text{iplus}\,(x,\ \text{itimes}\,(n,\ d))))) \\
\rightarrow & (\text{v\&c-apply\$}\,(\mathtt{'iterk}, \\
 & \qquad \text{list}\,(\text{cons}\,(a,\ \mathtt{0}), \\
 & \qquad\quad \text{cons}\,(b,\ \mathtt{0}), \\
 & \qquad\quad \text{cons}\,(c,\ \mathtt{0}), \\
 & \qquad\quad \text{cons}\,(d,\ \mathtt{0}), \\
 & \qquad\quad \mathtt{'(1\ .\ 0)}, \\
 & \qquad\quad \text{cons}\,(x,\ \mathtt{0}))) \\
 & \quad \leftrightarrow \quad \text{v\&c-apply\$}\,(\mathtt{'iterk}, \\
 & \qquad\qquad \text{list}\,(\text{cons}\,(a,\ \mathtt{0}), \\
 & \qquad\qquad\quad \text{cons}\,(b,\ \mathtt{0}), \\
 & \qquad\qquad\quad \text{cons}\,(c,\ \mathtt{0}),
\end{array}$$

```
                                   cons (d, 0),
                                   cons (iplus (1, itimes (n, iplus (-1, c))), 0),
                                   cons (iplus (x, itimes (n, d)), 0))))
```

THEOREM: iterk_e=1+_n_c-1&x+nd-value-base-step

```
((n = 1)
 ∧  ilessp (1, c)
 ∧  ilessp (0, d)
 ∧  (¬ ilessp (iplus (a, d), iplus (x, itimes (n, d)))))
→   (car (v&c-apply$ ('iterk,
                      list (cons (a, 0),
                            cons (b, 0),
                            cons (c, 0),
                            cons (d, 0),
                            '(1 . 0),
                            cons (x, 0))))
       =   car (v&c-apply$ ('iterk,
                            list (cons (a, 0),
                                  cons (b, 0),
                                  cons (c, 0),
                                  cons (d, 0),
                                  cons (iplus (1, itimes (n, iplus (-1, c))), 0),
                                  cons (iplus (x, itimes (n, d)), 0)))))
```

THEOREM: n-1_c-1>0-when-1<c&0<n&n<>1

```
(ilessp (1, c) ∧ ilessp (0, n) ∧ (n ≠ 1))
→   ilessp (0, itimes (iplus (-1, n), iplus (-1, c)))
```

THEOREM: iterk_e=1+_n_c-1&x+nd-exists-step-1

```
(ilessp (1, c) ∧ ilessp (0, n) ∧ (n ≠ 1))
→   (v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       cons (iplus (1, itimes (iplus (-1, n), iplus (-1, c))), 0),
                       cons (iplus (x, itimes (iplus (-1, n), d)), 0)))
       ↔   v&c-apply$ ('iterk,
                       list (cons (a, 0),
                             cons (b, 0),
                             cons (c, 0),
                             cons (d, 0),
                             cons (itimes (iplus (-1, n), iplus (-1, c)), 0),
                             v&c-apply$ ('iterk,
                                         list (cons (a, 0),
                                             cons (b, 0),
                                             cons (c, 0),
                                             cons (d, 0),
                                             '(1 . 0),
                                             cons (iplus (x,
                                                         itimes (iplus (-1, n),
                                                                d)),
                                                   0))))))
```

```
THEOREM: iterk_e=1+_n_c-1&x+nd-value-step-1
 (ilessp (1, c) ∧ ilessp (0, n) ∧ (n ≠ 1))
 →    (car (v&c-apply$ ('iterk,
                       list (cons (a, 0),
                            cons (b, 0),
                            cons (c, 0),
                            cons (d, 0),
                            cons (iplus (1, itimes (iplus (-1, n), iplus (-1, c))),
                                  0),
                            cons (iplus (x, itimes (iplus (-1, n), d)), 0))))
       =    car (v&c-apply$ ('iterk,
                            list (cons (a, 0),
                                 cons (b, 0),
                                 cons (c, 0),
                                 cons (d, 0),
                                 cons (itimes (iplus (-1, n), iplus (-1, c)), 0),
                                 v&c-apply$ ('iterk,
                                             list (cons (a, 0),
                                                  cons (b, 0),
                                                  cons (c, 0),
                                                  cons (d, 0),
                                                  '(1 . 0),
                                                  cons (iplus (x,
                                                              itimes (iplus (-1,
                                                                             n),
                                                                     d)),
                                                        0)))))))
```

```
THEOREM: iterk_e=1+_n_c-1&x+nd-exists-step-2
 (ilessp (0, n)
  ∧   ilessp (0, d)
  ∧   (¬ ilessp (iplus (a, d), iplus (x, itimes (n, d)))))
 →    (v&c-apply$ ('iterk,
                  list (cons (a, 0),
                       cons (b, 0),
                        cons (c, 0),
                        cons (d, 0),
                        cons (itimes (iplus (-1, n), iplus (-1, c)), 0),
                        v&c-apply$ ('iterk,
                                    list (cons (a, 0),
                                          cons (b, 0),
                                          cons (c, 0),
                                          cons (d, 0),
                                          '(1 . 0),
                                          cons (iplus (x, itimes (iplus (-1, n), d)),
                                                0)))))
      ↔   v&c-apply$ ('iterk,
                      list (cons (a, 0),
                            cons (b, 0),
                            cons (c, 0),
                            cons (d, 0),
                            cons (itimes (iplus (-1, n), iplus (-1, c)), 0),
                            v&c-apply$ ('iterk,
                                        list (cons (a, 0),
                                              cons (b, 0),
                                              cons (c, 0),
                                              cons (d, 0),
                                              cons (c, 0),
                                              cons (iplus (x, itimes (n, d)), 0))))))
```

```
THEOREM: iterk_e=1+_n_c-1&x+nd-value-step-2
 (ilessp (0, n)
  ∧   ilessp (0, d)
  ∧   (¬ ilessp (iplus (a, d), iplus (x, itimes (n, d)))))
 →   (car (v&c-apply$ ('iterk,
                        list (cons (a, 0),
                              cons (b, 0),
                              cons (c, 0),
                              cons (d, 0),
                              cons (itimes (iplus (-1, n), iplus (-1, c)), 0),
                              v&c-apply$ ('iterk,
                                          list (cons (a, 0),
                                                cons (b, 0),
                                                cons (c, 0),
                                                cons (d, 0),
                                                '(1 . 0),
                                                cons (iplus (x,
                                                             itimes (iplus (-1, n), d)),
                                                      0))))))
=    car (v&c-apply$ ('iterk,
                       list (cons (a, 0),
                             cons (b, 0),
                             cons (c, 0),
                             cons (d, 0),
                             cons (itimes (iplus (-1, n), iplus (-1, c)), 0),
                             v&c-apply$ ('iterk,
                                         list (cons (a, 0),
                                               cons (b, 0),
                                               cons (c, 0),
                                               cons (d, 0),
                                               cons (c, 0),
                                               cons (iplus (x, itimes (n, d)),
                                                     0)))))))
```

THEOREM: iterk_e=1+_n_c-1&x+nd-exists-step-3

```
 (ilessp (0, n) ∧ (n ≠ 1) ∧ ilessp (1, c))
 →    (v&c-apply$ ('iterk,
                   list (cons (a, 0),
                         cons (b, 0),
                         cons (c, 0),
                         cons (d, 0),
                         cons (itimes (iplus (-1, n), iplus (-1, c)), 0),
                         v&c-apply$ ('iterk,
                                     list (cons (a, 0),
                                           cons (b, 0),
                                           cons (c, 0),
                                           cons (d, 0),
                                           cons (c, 0),
                                           cons (iplus (x, itimes (n, d)), 0)))))
      ↔    v&c-apply$ ('iterk,
                       list (cons (a, 0),
                             cons (b, 0),
                             cons (c, 0),
                             cons (d, 0),
                             cons (iplus (1, itimes (n, iplus (-1, c))), 0),
                             cons (iplus (x, itimes (n, d)), 0))))
```

THEOREM: iterk_e=1+_n_c-1&x+nd-value-step-3

```
 (ilessp (0, n) ∧ (n ≠ 1) ∧ ilessp (1, c))
 →    (car (v&c-apply$ ('iterk,
                        list (cons (a, 0),
                              cons (b, 0),
                              cons (c, 0),
                          cons (d, 0),
                          cons (itimes (iplus (-1, n), iplus (-1, c)), 0),
                          v&c-apply$ ('iterk,
                                      list (cons (a, 0),
                                            cons (b, 0),
                                            cons (c, 0),
                                            cons (d, 0),
                                            cons (c, 0),
                                            cons (iplus (x, itimes (n, d)), 0))))))
          =   car (v&c-apply$ ('iterk,
                               list (cons (a, 0),
                                     cons (b, 0),
                                     cons (c, 0),
                                     cons (d, 0),
                                     cons (iplus (1, itimes (n, iplus (-1, c))), 0),
                                     cons (iplus (x, itimes (n, d)), 0)))))
```

THEOREM: a+d>=x+_n-1_d-when-a+d>=x+nd&0<d&0<n

```
 (ilessp (0, n)
  ∧   ilessp (0, d)
  ∧   (¬ ilessp (iplus (a, d), iplus (x, itimes (n, d)))))
 →    (¬ ilessp (iplus (a, d), iplus (x, itimes (iplus (-1, n), d))))
```

THEOREM: iterk_e=1+_n_c-1&x+nd-exists-when-1<c&0<d&0<n&a+d>=x+nd

```
 (ilessp (1, c)
  ∧   ilessp (0, d)
  ∧   ilessp (0, n)
  ∧   (¬ ilessp (iplus (a, d), iplus (x, itimes (n, d)))))
 →    (v&c-apply$ ('iterk,
                   list (cons (a, 0),
                         cons (b, 0),
                         cons (c, 0),
                         cons (d, 0),
                         '(1 . 0),
                         cons (x, 0)))
        ↔   v&c-apply$ ('iterk,
                        list (cons (a, 0),
                              cons (b, 0),
                              cons (c, 0),
                              cons (d, 0),
                              cons (iplus (1, itimes (n, iplus (-1, c))), 0),
                              cons (iplus (x, itimes (n, d)), 0))))
```

THEOREM: iterk_e=1+_n_c-1&x+nd-value-when-1<c&0<d&0<n&a+d>=x+nd

```
 (ilessp (1, c)
 ∧    ilessp (0, d)
 ∧    ilessp (0, n)
 ∧    (¬ ilessp (iplus (a, d), iplus (x, itimes (n, d)))))
→    (car (v&c-apply$ ('iterk,
                       list (cons (a, 0),
                             cons (b, 0),
                             cons (c, 0),
                             cons (d, 0),
                             '(1 . 0),
                             cons (x, 0))))
         =    car (v&c-apply$ ('iterk,
                             list (cons (a, 0),
                                   cons (b, 0),
                                   cons (c, 0),
                                   cons (d, 0),
                                   cons (iplus (1, itimes (n, iplus (-1, c))), 0),
                                   cons (iplus (x, itimes (n, d)), 0)))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Define the function N( a,d,x ) recursively,
;  so that whenever d > 0, N( a,d,x ) is the smallest
;  nonnegative integer i such that x + id > a.
```

Definition:
n (*a*, *d*, *x*)
= **if** ¬ ilessp (0, *d*) **then** 0
elseif ilessp (*a*, *x*) **then** 0
else iplus (1, n (*a*, *d*, iplus (*x*, *d*))) **endif**

Theorem: n>=0
¬ ilessp (n (*a*, *d*, *x*), 0)

Theorem: n>0-when-d>o&x<=a
(ilessp (0, *d*) ∧ (¬ ilessp (*a*, *x*))) → ilessp (0, n (*a*, *d*, *x*))

Theorem: a<x+nd
ilessp (0, *d*) → ilessp (*a*, iplus (*x*, itimes (n (*a*, *d*, *x*), *d*)))

Theorem: a+d>=x+nd-when-d>0&a>=x
(ilessp (0, *d*) ∧ (¬ ilessp (*a*, *x*)))
→ (¬ ilessp (iplus (*a*, *d*), iplus (*x*, itimes (n (*a*, *d*, *x*), *d*))))

Theorem: iterk-e=1&x-iff-iterk_e=1+_n_c-1&x+nd-when-1<c&0<d&a>=x
(ilessp (1, *c*) ∧ ilessp (0, *d*) ∧ (¬ ilessp (*a*, *x*)))

```
 →   (v&c-apply$ ('iterk,
                  list (cons (a, 0),
                        cons (b, 0),
                        cons (c, 0),
                        cons (d, 0),
                        '(1 . 0),
                        cons (x, 0)))
        ↔   v&c-apply$ ('iterk,
                        list (cons (a, 0),
                              cons (b, 0),
                              cons (c, 0),
                              cons (d, 0),
                              cons (iplus (1, itimes (n (a, d, x), iplus (-1, c))),
                                    0),
                              cons (iplus (x, itimes (n (a, d, x), d)), 0))))
```

THEOREM: iterk-exists-when-e=1+n_c-1&x=x+nd&b<=0

```
((¬ ilessp (0, b)) ∧ ilessp (1, c) ∧ ilessp (0, d))
 →   v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       cons (iplus (1, itimes (n (a, d, x), iplus (-1, c))), 0),
                       cons (iplus (x, itimes (n (a, d, x), d)), 0)))
```

THEOREM: iterk-exists-when-x<=a&d>0&c<1&b<=0

```
((¬ ilessp (0, b)) ∧ ilessp (1, c) ∧ ilessp (0, d) ∧ (¬ ilessp (a, x)))
 →   v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       '(1 . 0),
                       cons (x, 0)))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; The next two events are versions of Part 5 of the
; Main Theorem given above in the introduction.

;  5. If  x <= a, d > 0, c > 1, and b <= 0, then
;     K( a,b,c,d,x ) exists.

;     Proof. By Lemma 7, the definition of the
;            function N, Part 1 of the Main Theorem,
;            and the definition of K.
```

Theorem: k-exists-when-x<=a&d>0&c<1&b<=0

$((\neg\ \text{ilessp}\,(0,\ b)) \land \text{ilessp}\,(1,\ c) \land \text{ilessp}\,(0,\ d) \land (\neg\ \text{ilessp}\,(a,\ x)))$
$\rightarrow\quad \text{v\&c-apply\$}\,(\texttt{'k},$
$\qquad\qquad \text{list}\,(\text{cons}\,(a,\ 0),\ \text{cons}\,(b,\ 0),\ \text{cons}\,(c,\ 0),\ \text{cons}\,(d,\ 0),\ \text{cons}\,(x,\ 0)))$

Theorem: k-halts-when-x<=a&d>0&c<1&b<=0

$(\textit{vc-a}$
$\ \land\quad \textit{vc-b}$
$\ \land\quad \textit{vc-c}$
$\ \land\quad \textit{vc-d}$
$\ \land\quad \textit{vc-x}$
$\ \land\quad (\neg\ \text{ilessp}\,(0,\ \text{car}\,(\textit{vc-b})))$
$\ \land\quad \text{ilessp}\,(1,\ \text{car}\,(\textit{vc-c}))$
$\ \land\quad \text{ilessp}\,(0,\ \text{car}\,(\textit{vc-d}))$
$\ \land\quad (\neg\ \text{ilessp}\,(\text{car}\,(\textit{vc-a}),\ \text{car}\,(\textit{vc-x}))))$
$\rightarrow\quad \text{v\&c-apply\$}\,(\texttt{'k},\ \text{list}\,(\textit{vc-a},\ \textit{vc-b},\ \textit{vc-c},\ \textit{vc-d},\ \textit{vc-x}))$

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Lemma 8. Generalize Lemma 7 to the case when e > 1.

;  Assume 1 < c, 0 < d, 1 < e, 0 < n, and a+d >= x+nd.

;  Then

;  IterK( a,b,c,d,e,x ) exists
;   iff
;  IterK( a,b,c,d,e+n(c-1),x+nd ) exists

;  and

;  IterK( a,b,c,d,e,x )
;   =
;  IterK( a,b,c,d,e+n(c-1),x+nd ).

; Proof.  By Lemma 6 (with e2 = 1) and Lemma 7.
```

Theorem: x+-x+y=fix-int-y

$\text{ilessp}\,(0,\ x) \rightarrow (\text{iplus}\,(x,\ \text{iplus}\,(-\ x,\ y)) = \text{fix-int}\,(y))$

Theorem: fix-int-x=x-when-x>1

$\text{ilessp}\,(1,\ x) \rightarrow (\text{fix-int}\,(x) = x)$

THEOREM: iterk-e-x-iff-iterk-e-1-iterk-1-x

```
 ilessp (1, e)
 →    (v&c-apply$ ('iterk,
                   list (cons (a, 0),
                         cons (b, 0),
                         cons (c, 0),
                         cons (d, 0),
                         cons (e, 0),
                         cons (x, 0)))
        ↔   v&c-apply$ ('iterk,
                        list (cons (a, 0),
                              cons (b, 0),
                              cons (c, 0),
                              cons (d, 0),
                              cons (iplus (-1, e), 0),
                              v&c-apply$ ('iterk,
                                          list (cons (a, 0),
                                                cons (b, 0),
                                                cons (c, 0),
                                                cons (d, 0),
                                                '(1 . 0),
                                                cons (x, 0))))))
```

THEOREM: iterk-e-x=iterk-e-1-iterk-1-x

```
 ilessp (1, e)
 →    (car (v&c-apply$ ('iterk,
                        list (cons (a, 0),
                              cons (b, 0),
                              cons (c, 0),
                              cons (d, 0),
                              cons (e, 0),
                              cons (x, 0))))
        =   car (v&c-apply$ ('iterk,
                             list (cons (a, 0),
                                   cons (b, 0),
                                   cons (c, 0),
                                   cons (d, 0),
                                   cons (iplus (-1, e), 0),
                                   v&c-apply$ ('iterk,
                                               list (cons (a, 0),
                                                     cons (b, 0),
                                                     cons (c, 0),
                                                     cons (d, 0),
                                                     '(1 . 0),
                                        cons (x, 0)))))))
```

THEOREM: iterk_e+_n_c-1&x+nd-exists-when-1<c&0<d&1<e&0<n&a+d>=x+nd-step-1

```
(ilessp (1, c)
 ∧  ilessp (0, d)
 ∧  ilessp (0, n)
 ∧  (¬ ilessp (iplus (a, d), iplus (x, itimes (n, d)))))
→   (v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       cons (iplus (-1, e), 0),
                       v&c-apply$ ('iterk,
                                   list (cons (a, 0),
                                         cons (b, 0),
                                         cons (c, 0),
                                         cons (d, 0),
                                         '(1 . 0),
                                         cons (x, 0)))))
       ↔   v&c-apply$ ('iterk,
                       list (cons (a, 0),
                             cons (b, 0),
                             cons (c, 0),
                             cons (d, 0),
                             cons (iplus (-1, e), 0),
                             v&c-apply$ ('iterk,
                                         list (cons (a, 0),
                                               cons (b, 0),
                                               cons (c, 0),
                                               cons (d, 0),
                                               cons (iplus (1,
                                                            itimes (n,
                                                                    iplus (-1, c))),
                                                     0),
                                               cons (iplus (x, itimes (n, d)), 0))))))
```

THEOREM: iterk_e+_n_c-1&x+nd-value-when-1<c&0<d&1<e&0<n&a+d>=x+nd-step-1

```
(ilessp (1, c)
 ∧  ilessp (0, d)
 ∧  ilessp (0, n)
 ∧  (¬ ilessp (iplus (a, d), iplus (x, itimes (n, d)))))
→   (car (v&c-apply$ ('iterk,
                      list (cons (a, 0),
                          cons (b, 0),
                          cons (c, 0),
                          cons (d, 0),
                          cons (iplus (-1, e), 0),
                          v&c-apply$ ('iterk,
                                      list (cons (a, 0),
                                            cons (b, 0),
                                            cons (c, 0),
                                            cons (d, 0),
                                            '(1 . 0),
                                            cons (x, 0))))))
  =   car (v&c-apply$ ('iterk,
                       list (cons (a, 0),
                             cons (b, 0),
                             cons (c, 0),
                             cons (d, 0),
                             cons (iplus (-1, e), 0),
                             v&c-apply$ ('iterk,
                                         list (cons (a, 0),
                                               cons (b, 0),
                                               cons (c, 0),
                                               cons (d, 0),
                                               cons (iplus (1,
                                                            itimes (n,
                                                                    iplus (-1,
                                                                           c))),
                                                     0),
                                               cons (iplus (x, itimes (n, d)),
                                                     0)))))))
```

THEOREM: iterk_e+_n_c-1&x+nd-exists-when-1<c&0<d&1<e&0<n&a+d>=x+nd-step-2

```
 (ilessp (1, c) ∧ ilessp (1, e) ∧ ilessp (0, n))
 →   (v&c-apply$ ('iterk,
                  list (cons (a, 0),
                        cons (b, 0),
                        cons (c, 0),
                        cons (d, 0),
                        cons (iplus (-1, e), 0),
                        v&c-apply$ ('iterk,
                                    list (cons (a, 0),
                                          cons (b, 0),
                                          cons (c, 0),
                                          cons (d, 0),
                                          cons (iplus (1, itimes (n, iplus (-1, c))),
                                        0),
                                   cons (iplus (x, itimes (n, d)), 0)))))
    ↔   v&c-apply$ ('iterk,
                    list (cons (a, 0),
                          cons (b, 0),
                          cons (c, 0),
                          cons (d, 0),
                          cons (iplus (e, itimes (n, iplus (-1, c))), 0),
                          cons (iplus (x, itimes (n, d)), 0))))
```

THEOREM: iterk_e+_n_c-1&x+nd-value-when-1<c&0<d&1<e&0<n&a+d>=x+nd-step-2

```
(ilessp (1, c) ∧ ilessp (1, e) ∧ ilessp (0, n))
→  (car (v&c-apply$ ('iterk,
                     list (cons (a, 0),
                           cons (b, 0),
                           cons (c, 0),
                           cons (d, 0),
                           cons (iplus (-1, e), 0),
                           v&c-apply$ ('iterk,
                                       list (cons (a, 0),
                                             cons (b, 0),
                                             cons (c, 0),
                                             cons (d, 0),
                                             cons (iplus (1,
                                                          itimes (n, iplus (-1, c))),
                                                   0),
                                             cons (iplus (x, itimes (n, d)), 0))))))
    =   car (v&c-apply$ ('iterk,
                         list (cons (a, 0),
                               cons (b, 0),
                               cons (c, 0),
                               cons (d, 0),
                               cons (iplus (e, itimes (n, iplus (-1, c))), 0),
                               cons (iplus (x, itimes (n, d)), 0)))))
```

THEOREM: iterk_e+_n_c-1&x+nd-exists-when-1<c&0<d&1<e&0<n&a+d>=x+nd

```
(ilessp (1, c)
 ∧  ilessp (0, d)
 ∧  ilessp (1, e)
 ∧  ilessp (0, n)
 ∧  (¬ ilessp (iplus (a, d), iplus (x, itimes (n, d)))))
→  (v&c-apply$ ('iterk,
                list (cons (a, 0),
                      cons (b, 0),
                      cons (c, 0),
                      cons (d, 0),
                      cons (e, 0),
                      cons (x, 0)))
      ↔   v&c-apply$ ('iterk,
                      list (cons (a, 0),
                            cons (b, 0),
                            cons (c, 0),
                            cons (d, 0),
                            cons (iplus (e, itimes (n, iplus (-1, c))), 0),
                            cons (iplus (x, itimes (n, d)), 0))))
```

THEOREM: iterk_e+_n_c-1&x+nd-value-when-1<c&0<d&1<e&0<n&a+d>=x+nd

```
(ilessp (1, c)
 ∧  ilessp (0, d)
 ∧  ilessp (1, e)
 ∧  ilessp (0, n)
 ∧  (¬ ilessp (iplus (a, d), iplus (x, itimes (n, d)))))
→   (car (v&c-apply$ ('iterk,
                      list (cons (a, 0),
                            cons (b, 0),
                            cons (c, 0),
                            cons (d, 0),
                            cons (e, 0),
                            cons (x, 0))))
      =   car (v&c-apply$ ('iterk,
                           list (cons (a, 0),
                                 cons (b, 0),
                                 cons (c, 0),
                                 cons (d, 0),
                                 cons (iplus (e, itimes (n, iplus (-1, c))), 0),
                                 cons (iplus (x, itimes (n, d)), 0)))))
```

THEOREM: iterk-e&x-iff-iterk_e+_n_c-1&x+nd-when-1<c&0<d&a>=x&e>1

```
(ilessp (1, c) ∧ ilessp (0, d) ∧ ilessp (1, e) ∧ (¬ ilessp (a, x)))
→   (v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       cons (e, 0),
                       cons (x, 0)))
      ↔   v&c-apply$ ('iterk,
                      list (cons (a, 0),
                                 cons (b, 0),
                                 cons (c, 0),
                                 cons (d, 0),
                                 cons (iplus (e, itimes (n (a, d, x), iplus (-1, c))), 0),
                                 cons (iplus (x, itimes (n (a, d, x), d)), 0))))
```

Theorem: iterk-e&x=iterk_e+_n_c-1&x+nd-when-1<c&0<d&a>=x&e>1

```
 (ilessp (1, c) ∧ ilessp (0, d) ∧ ilessp (1, e) ∧ (¬ ilessp (a, x)))
 →    (car (v&c-apply$ ('iterk,
                         list (cons (a, 0),
                               cons (b, 0),
                               cons (c, 0),
                               cons (d, 0),
                               cons (e, 0),
                               cons (x, 0))))
        =    car (v&c-apply$ ('iterk,
                               list (cons (a, 0),
                                     cons (b, 0),
                                     cons (c, 0),
                                     cons (d, 0),
                                     cons (iplus (e,
                                                  itimes (n (a, d, x), iplus (-1, c))),
                                           0),
                                     cons (iplus (x, itimes (n (a, d, x), d)), 0)))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Lemma 9. The number of iterates of K can be reduced
;          by one if the value of x is reduced by b
;          when a < x.

;  Assume  1 < e, and  a < x.

;  Then

;  IterK( a,b,c,d,e,x ) exists
;   iff
;  IterK( a,b,c,d,e-1,x-b ) exists

;  and

;  IterK( a,b,c,d,e,x )
;   =
;  IterK( a,b,c,d,e-1,x-b ).
; Proof.  By Lemma 6 (with e2 = 1), and the definition
;         of IterK for values of x larger than a.
```

THEOREM: iterk_e_x-iff-iterk_e-1_x-b-when-1<e&x>a-step-1

```
 (ilessp (a, x) ∧ ilessp (1, e))
 →    (v&c-apply$ ('iterk,
                       list (cons (a, 0),
                             cons (b, 0),
                             cons (c, 0),
                             cons (d, 0),
                             cons (iplus (-1, e), 0),
                             v&c-apply$ ('iterk,
                                           list (cons (a, 0),
                                                 cons (b, 0),
                                                 cons (c, 0),
                                                 cons (d, 0),
                                                 '(1 . 0),
                                                 cons (x, 0)))))
        ↔    v&c-apply$ ('iterk,
                            list (cons (a, 0),
                                  cons (b, 0),
                                  cons (c, 0),
                                  cons (d, 0),
                                  cons (iplus (-1, e), 0),
                                  cons (idifference (x, b), 0))))
```

THEOREM: iterk_e_x=iterk_e-1_x-b-when-1<e&x>a-step-1

```
 (ilessp (a, x) ∧ ilessp (1, e))
 →    (car (v&c-apply$ ('iterk,
                            list (cons (a, 0),
                                  cons (b, 0),
                                  cons (c, 0),
                                  cons (d, 0),
                                  cons (iplus (-1, e), 0),
                                  v&c-apply$ ('iterk,
                                                list (cons (a, 0),
                                                      cons (b, 0),
                                                      cons (c, 0),
                                                      cons (d, 0),
                                                      '(1 . 0),
                                                      cons (x, 0))))))
        =    car (v&c-apply$ ('iterk,
                                list (cons (a, 0),
                                   cons (b, 0),
                                   cons (c, 0),
                                   cons (d, 0),
                                   cons (iplus (-1, e), 0),
                                   cons (idifference (x, b), 0)))))
```

THEOREM: iterk_e_x-iff-iterk_e-1_x-b-when-1<e&x>a

```
 (ilessp (a, x) ∧ ilessp (1, e))
 →   (v&c-apply$ ('iterk,
                  list (cons (a, 0),
                        cons (b, 0),
                        cons (c, 0),
                        cons (d, 0),
                        cons (e, 0),
                        cons (x, 0)))
        ↔   v&c-apply$ ('iterk,
                        list (cons (a, 0),
                              cons (b, 0),
                              cons (c, 0),
                              cons (d, 0),
                              cons (iplus (-1, e), 0),
                              cons (idifference (x, b), 0))))
```

THEOREM: iterk_e_x=iterk_e-1_x-b-when-1<e&x>a

```
 (ilessp (a, x) ∧ ilessp (1, e))
 →   (car (v&c-apply$ ('iterk,
                       list (cons (a, 0),
                             cons (b, 0),
                             cons (c, 0),
                             cons (d, 0),
                             cons (e, 0),
                             cons (x, 0))))
        =   car (v&c-apply$ ('iterk,
                             list (cons (a, 0),
                                   cons (b, 0),
                                   cons (c, 0),
                                   cons (d, 0),
                                   cons (iplus (-1, e), 0),
                                   cons (idifference (x, b), 0)))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Lemma 10. A key fact noted by Knuth in his proof.

;  Assume 1 < c, 0 < d, 1 < e, and  x <= a.
;  Then

;  IterK( a,b,c,d,e,x ) exists
;   iff
;  IterK( a,b,c,d,e-1+N( a,d,x )(c-1),x+N( a,d,x )d-b )
;  exists

;  and

;  IterK( a,b,c,d,e,x )
;   =
;  IterK( a,b,c,d,e-1+N( a,d,x )(c-1),x+N( a,d,x )d-b ).

; Proof.  By Lemma 8, the definition of the function N,
;         and Lemma 9.
```

THEOREM: e+n_c-1_>1-when-1<c&1<e&0<n

$$\begin{array}{l}
(\mathrm{ilessp}\,(\mathtt{1},\ c) \land \mathrm{ilessp}\,(\mathtt{1},\ e) \land \mathrm{ilessp}\,(\mathtt{0},\ n)) \\
\to \quad \mathrm{ilessp}\,(\mathtt{1},\ \mathrm{iplus}\,(e,\ \mathrm{itimes}\,(n,\ \mathrm{iplus}\,(\mathtt{-1},\ c))))
\end{array}$$

THEOREM: iterk-e&x-iff-iterk_-1+e+_n_c-1&x+nd-b-when-1<c&0<d&a>=x&e>1

$$\begin{array}{l}
(\mathrm{ilessp}\,(\mathtt{1},\ c) \land \mathrm{ilessp}\,(\mathtt{0},\ d) \land \mathrm{ilessp}\,(\mathtt{1},\ e) \land (\neg\ \mathrm{ilessp}\,(a,\ x))) \\
\to \quad (\mathrm{v\&c\text{-}apply\$}\,(\mathtt{'iterk}, \\
\qquad\qquad\qquad \mathrm{list}\,(\mathrm{cons}\,(a,\ \mathtt{0}), \\
\qquad\qquad\qquad\qquad \mathrm{cons}\,(b,\ \mathtt{0}), \\
\qquad\qquad\qquad\qquad \mathrm{cons}\,(c,\ \mathtt{0}), \\
\qquad\qquad\qquad\qquad \mathrm{cons}\,(d,\ \mathtt{0}), \\
\qquad\qquad\qquad\qquad \mathrm{cons}\,(e,\ \mathtt{0}), \\
\qquad\qquad\qquad\qquad \mathrm{cons}\,(x,\ \mathtt{0}))) \\
\qquad \leftrightarrow \quad \mathrm{v\&c\text{-}apply\$}\,(\mathtt{'iterk}, \\
\qquad\qquad\qquad\qquad \mathrm{list}\,(\mathrm{cons}\,(a,\ \mathtt{0}), \\
\qquad\qquad\qquad\qquad\quad \mathrm{cons}\,(b,\ \mathtt{0}), \\
\qquad\qquad\qquad\qquad\quad \mathrm{cons}\,(c,\ \mathtt{0}), \\
\qquad\qquad\qquad\qquad\quad \mathrm{cons}\,(d,\ \mathtt{0}), \\
\qquad\qquad\qquad\qquad\quad \mathrm{cons}\,(\mathrm{iplus}\,(\mathtt{-1}, \\
\qquad\qquad\qquad\qquad\qquad\qquad \mathrm{iplus}\,(e, \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathrm{itimes}\,(\mathrm{n}\,(a,\ d,\ x),\ \mathrm{iplus}\,(\mathtt{-1},\ c)))), \\
\qquad\qquad\qquad\qquad\qquad \mathtt{0}), \\
\qquad\qquad\qquad\qquad\quad \mathrm{cons}\,(\mathrm{idifference}\,(\mathrm{iplus}\,(x,\ \mathrm{itimes}\,(\mathrm{n}\,(a,\ d,\ x),\ d)), \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad b), \\
\qquad\qquad\qquad\qquad\qquad \mathtt{0}))))
\end{array}$$

THEOREM: iterk-e&x=iterk_-1+e+_n_c-1&x+nd-b-when-1<c&0<d&a>=x&e>1

$$(\mathrm{ilessp}\,(\mathtt{1},\ c) \land \mathrm{ilessp}\,(\mathtt{0},\ d) \land \mathrm{ilessp}\,(\mathtt{1},\ e) \land (\neg\ \mathrm{ilessp}\,(a,\ x)))$$

```
→    (car (v&c-apply$ ('iterk,
                       list (cons (a, 0),
                             cons (b, 0),
                             cons (c, 0),
                             cons (d, 0),
                             cons (e, 0),
                             cons (x, 0))))
        =    car (v&c-apply$ ('iterk,
                              list (cons (a, 0),
                                    cons (b, 0),
                                    cons (c, 0),
                                    cons (d, 0),
                                    cons (iplus (-1,
                                                 iplus (e,
                                                        itimes (n (a, d, x),
                                                                iplus (-1, c)))),
                                          0),
                                    cons (idifference (iplus (x,
                                                              itimes (n (a, d, x), d)),
                                                       b),
                                          0)))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Lemma 11. Another key fact noted by Knuth.

;  Assume 0 < b, 1 < c, 0 < d, 1 < e, and  x <= a.

;  Then

;  IterK( a,b,c,d,e-1,x-b ) exists
;   iff
;  IterK( a,b,c,d,e-1+N( a,d,x )(c-1),x-b+N( a,d,x )d )
;  exists

;  and

;  IterK( a,b,c,d,e-1,x-b )
;   =
;  IterK( a,b,c,d,e-1+N( a,d,x )(c-1),x-b+N( a,d,x )d ).

; Proof.  By Lemma 7 (if e = 2), Lemma 8 (if e > 2), and
;         the definition of the function N.
```

THEOREM: a+d>=x-b+nd-when-b>0&a+d>=x+nd&d>0

```
(ilessp (0, b)
 ∧   ilessp (0, d)
 ∧   ilessp (0, n)
 ∧   (¬ ilessp (iplus (a, d), iplus (x, itimes (n, d)))))
→    (¬ ilessp (iplus (a, d), iplus (idifference (x, b), itimes (n, d))))
```

THEOREM: iterk-e-1&x-b-iff-iterk_e-1+_n_c-1&x-b+nd-when-1<c&0<d&a>=x&b>0&e>2

```
(ilessp (0, b)
 ∧   ilessp (1, c)
 ∧   ilessp (0, d)
 ∧   ilessp (2, e)
 ∧   (¬ ilessp (a, x)))
→    (v&c-apply$ ('iterk,
                  list (cons (a, 0),
                        cons (b, 0),
                        cons (c, 0),
                        cons (d, 0),
                        cons (iplus (-1, e), 0),
                        cons (idifference (x, b), 0)))
        ↔    v&c-apply$ ('iterk,
                         list (cons (a, 0),
                               cons (b, 0),
                               cons (c, 0),
                               cons (d, 0),
                               cons (iplus (-1,
                                            iplus (e,
                                                   itimes (n (a, d, x), iplus (-1, c)))),
                                     0),
                               cons (idifference (iplus (x, itimes (n (a, d, x), d)),
                                                  b),
                                     0))))
```

THEOREM: iterk-e-1&x-b=iterk_e-1+_n_c-1&x-b+nd-when-1<c&0<d&a>=x&e>2

```
(ilessp (0, b)
 ∧   ilessp (1, c)
 ∧   ilessp (0, d)
 ∧   ilessp (2, e)
 ∧   (¬ ilessp (a, x)))
→    (car (v&c-apply$ ('iterk,
                       list (cons (a, 0),
                             cons (b, 0),
                             cons (c, 0),
                             cons (d, 0),
                             cons (iplus (-1, e), 0),
                          cons (idifference (x, b), 0))))
    =   car (v&c-apply$ ('iterk,
                          list (cons (a, 0),
                                cons (b, 0),
                                cons (c, 0),
                                cons (d, 0),
                                cons (iplus (-1,
                                             iplus (e,
                                                    itimes (n (a, d, x),
                                                            iplus (-1, c)))),
                                      0),
                                cons (idifference (iplus (x,
                                                          itimes (n (a, d, x), d)),
                                                   b),
                                      0)))))
```

Theorem: x+2-1=x+1

```
iplus (-1, iplus (2, x)) = iplus (1, x)
```

Theorem: iterk-e-1&x-b-iff-iterk_e-1+_n_c-1&x-b+nd-when-1<c&0<d&a>=x&b>0&e=2

```
(ilessp (0, b)
 ∧  ilessp (1, c)
 ∧  ilessp (0, d)
 ∧  (e = 2)
 ∧  (¬ ilessp (a, x)))
→  (v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       cons (iplus (-1, e), 0),
                       cons (idifference (x, b), 0)))
     ↔  v&c-apply$ ('iterk,
                     list (cons (a, 0),
                           cons (b, 0),
                           cons (c, 0),
                           cons (d, 0),
                           cons (iplus (-1,
                                        iplus (e,
                                               itimes (n (a, d, x), iplus (-1, c)))),
                                 0),
                           cons (idifference (iplus (x, itimes (n (a, d, x), d)),
                                              b),
                                 0))))
```

THEOREM: iterk-e-1&x-b=iterk_e-1+_n_c-1&x-b+nd-when-1<c&0<d&a>=x&b>0&e=2

```
 (ilessp (0, b)
  ∧   ilessp (1, c)
  ∧   ilessp (0, d)
  ∧   (e = 2)
  ∧   (¬ ilessp (a, x)))
 →   (car (v&c-apply$ ('iterk,
                           list (cons (a, 0),
                                 cons (b, 0),
                                 cons (c, 0),
                                 cons (d, 0),
                                 cons (iplus (-1, e), 0),
                                 cons (idifference (x, b), 0))))
        =   car (v&c-apply$ ('iterk,
                                 list (cons (a, 0),
                                       cons (b, 0),
                                       cons (c, 0),
                                       cons (d, 0),
                                       cons (iplus (-1,
                                                    iplus (e,
                                                           itimes (n (a, d, x),
                                                                   iplus (-1, c)))),
                                             0),
                                       cons (idifference (iplus (x,
                                                                 itimes (n (a, d, x), d)),
                                                          b),
                                             0)))))
```

THEOREM: iterk-e-1&x-b-iff-iterk_e-1+_n_c-1&x-b+nd-when-1<c&0<d&a>=x&b>0&e>1

```
 (ilessp (0, b)
  ∧   ilessp (1, c)
  ∧   ilessp (0, d)
  ∧   ilessp (1, e)
  ∧   (¬ ilessp (a, x)))
 →   (v&c-apply$ ('iterk,
                    list (cons (a, 0),
                          cons (b, 0),
                          cons (c, 0),
                          cons (d, 0),
                          cons (iplus (-1, e), 0),
                          cons (idifference (x, b), 0)))
        ↔   v&c-apply$ ('iterk,
                          list (cons (a, 0),
                                cons (b, 0),
                                cons (c, 0),
                                cons (d, 0),
                                cons (iplus (-1,
                                             iplus (e,
                                                    itimes (n (a, d, x), iplus (-1, c)))),
                                      0),
                                cons (idifference (iplus (x, itimes (n (a, d, x), d)),
                                                   b),
                                      0))))
```

THEOREM: iterk-e-1&x-b=iterk_e-1+_n_c-1&x-b+nd-when-1<c&0<d&a>=x&b>0&e>1

```
 (ilessp (0, b)
  ∧  ilessp (1, c)
  ∧  ilessp (0, d)
  ∧  ilessp (1, e)
  ∧  (¬ ilessp (a, x)))
 →  (car (v&c-apply$ ('iterk,
                      list (cons (a, 0),
                            cons (b, 0),
                            cons (c, 0),
                            cons (d, 0),
                            cons (iplus (-1, e), 0),
                            cons (idifference (x, b), 0))))
       =  car (v&c-apply$ ('iterk,
                           list (cons (a, 0),
                                 cons (b, 0),
                                 cons (c, 0),
                                 cons (d, 0),
                                 cons (iplus (-1,
                                              iplus (e,
                                                     itimes (n (a, d, x),
                                                             iplus (-1, c)))),
                                       0),
                                 cons (idifference (iplus (x,
                                                           itimes (n (a, d, x), d)),
                                                    b),
                                       0)))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Lemma 12. The number of iterates of K can be reduced
;           by one if the value of x is reduced by b
;           when a >= x and restrictions are placed
;           on the parameters a,b,c, and d.
;  Assume  0 < b, 1 < c, 0 < d, 1 < e, and  a >= x.

;  Then

;  IterK( a,b,c,d,e,x ) exists
;   iff
;  IterK( a,b,c,d,e-1,x-b ) exists

;  and

;  IterK( a,b,c,d,e,x )
;   =
;  IterK( a,b,c,d,e-1,x-b ).

; Proof.  By Lemma 10 and Lemma 11.
```

Theorem: iterk_e_x-iff-iterk_e-1_x-b-when-0<b&1<c&0<d&1<e&a>=x

```
(ilessp (0, b)
 ∧   ilessp (1, c)
 ∧   ilessp (0, d)
 ∧   ilessp (1, e)
 ∧   (¬ ilessp (a, x)))
→    (v&c-apply$ ('iterk,
                  list (cons (a, 0),
                        cons (b, 0),
                        cons (c, 0),
                        cons (d, 0),
                        cons (e, 0),
                        cons (x, 0)))
      ↔   v&c-apply$ ('iterk,
                       list (cons (a, 0),
                             cons (b, 0),
                             cons (c, 0),
                             cons (d, 0),
                             cons (iplus (-1, e), 0),
                             cons (idifference (x, b), 0))))
```

Theorem: iterk_e_x=iterk_e-1_x-b-when-0<b&1<c&0<d&1<e&a>=x

```
(ilessp (0, b)
 ∧   ilessp (1, c)
 ∧   ilessp (0, d)
 ∧   ilessp (1, e)
 ∧   (¬ ilessp (a, x)))
```

```
→   (car (v&c-apply$ ('iterk,
                      list (cons (a, 0),
                            cons (b, 0),
                            cons (c, 0),
                            cons (d, 0),
                            cons (e, 0),
                            cons (x, 0))))
        =   car (v&c-apply$ ('iterk,
                            list (cons (a, 0),
                                  cons (b, 0),
                                  cons (c, 0),
                                  cons (d, 0),
                                  cons (iplus (-1, e), 0),
                                  cons (idifference (x, b), 0)))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Lemma 13. Combine Lemma 9 and Lemma 12 about reducing
;           the number of iterates of K by one.

;  Assume  0 < b, 1 < c, 0 < d, and 1 < e.

;  Then

;  IterK( a,b,c,d,e,x ) exists
;   iff
;  IterK( a,b,c,d,e-1,x-b ) exists

;  and

;  IterK( a,b,c,d,e,x )
;   =
;  IterK( a,b,c,d,e-1,x-b ).

; Proof.  By Lemma 9 (if x > a) and Lemma 12 (if x <= a).
```

THEOREM: iterk_e_x-iff-iterk_e-1_x-b-when-0<b&1<c&0<d&1<e

```
(ilessp (0, b) ∧ ilessp (1, c) ∧ ilessp (0, d) ∧ ilessp (1, e))
→   (v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       cons (e, 0),
                         cons (x, 0)))
     ↔   v&c-apply$ ('iterk,
                         list (cons (a, 0),
                               cons (b, 0),
                               cons (c, 0),
                               cons (d, 0),
                               cons (iplus (-1, e), 0),
                               cons (idifference (x, b), 0))))
```

THEOREM: iterk_e_x=iterk_e-1_x-b-when-0<b&1<c&0<d&1<e

```
(ilessp (0, b) ∧ ilessp (1, c) ∧ ilessp (0, d) ∧ ilessp (1, e))
→   (car (v&c-apply$ ('iterk,
                       list (cons (a, 0),
                             cons (b, 0),
                             cons (c, 0),
                             cons (d, 0),
                             cons (e, 0),
                             cons (x, 0))))
     =   car (v&c-apply$ ('iterk,
                           list (cons (a, 0),
                                 cons (b, 0),
                                 cons (c, 0),
                                 cons (d, 0),
                                 cons (iplus (-1, e), 0),
                                 cons (idifference (x, b), 0)))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Lemma 14. Generalize Lemma 13 by reducing the number
;           of iterates of K by more than one.

;  Assume  0 < b, 1 < c, 0 < d, 1 < e, and 0 < j < e.

;  Then

;  IterK( a,b,c,d,e,x ) exists
;   iff
;  IterK( a,b,c,d,e-j,x-jb ) exists

;  and

;  IterK( a,b,c,d,e,x )
;   =
;  IterK( a,b,c,d,e-j,x-jb ).
; Proof.  By induction on j.
```

Theorem: iterk_e_x-iff-iterk_e-j_x-jb-when-0<b&1<c&0<d&1<e&0<j<e

$$
\begin{array}{l}
(\text{ilessp}\,(\mathtt{0},\ b) \\
\quad \land \quad \text{ilessp}\,(\mathtt{1},\ c) \\
\quad \land \quad \text{ilessp}\,(\mathtt{0},\ d) \\
\quad \land \quad \text{ilessp}\,(\mathtt{1},\ e) \\
\quad \land \quad \text{ilessp}\,(\mathtt{0},\ j) \\
\quad \land \quad \text{ilessp}\,(j,\ e)) \\
\to \quad (\text{v\&c-apply\$}\,(\mathtt{'iterk}, \\
\qquad\qquad\qquad\quad \text{list}\,(\text{cons}\,(a,\ \mathtt{0}), \\
\qquad\qquad\qquad\qquad\quad \text{cons}\,(b,\ \mathtt{0}), \\
\qquad\qquad\qquad\qquad\quad \text{cons}\,(c,\ \mathtt{0}), \\
\qquad\qquad\qquad\qquad\quad \text{cons}\,(d,\ \mathtt{0}), \\
\qquad\qquad\qquad\qquad\quad \text{cons}\,(e,\ \mathtt{0}), \\
\qquad\qquad\qquad\qquad\quad \text{cons}\,(x,\ \mathtt{0}))) \\
\qquad \leftrightarrow \quad \text{v\&c-apply\$}\,(\mathtt{'iterk}, \\
\qquad\qquad\qquad\qquad\quad \text{list}\,(\text{cons}\,(a,\ \mathtt{0}), \\
\qquad\qquad\qquad\qquad\qquad\quad \text{cons}\,(b,\ \mathtt{0}), \\
\qquad\qquad\qquad\qquad\qquad\quad \text{cons}\,(c,\ \mathtt{0}), \\
\qquad\qquad\qquad\qquad\qquad\quad \text{cons}\,(d,\ \mathtt{0}), \\
\qquad\qquad\qquad\qquad\qquad\quad \text{cons}\,(\text{idifference}\,(e,\ j),\ \mathtt{0}), \\
\qquad\qquad\qquad\qquad\qquad\quad \text{cons}\,(\text{idifference}\,(x,\ \text{itimes}\,(b,\ j)),\ \mathtt{0}))))
\end{array}
$$

Theorem: iterk_e_x=iterk_e-j_x-jb-when-0<b&1<c&0<d&1<e&0<j<e

$$
\begin{array}{l}
(\text{ilessp}\,(\mathtt{0},\ b) \\
\quad \land \quad \text{ilessp}\,(\mathtt{1},\ c) \\
\quad \land \quad \text{ilessp}\,(\mathtt{0},\ d) \\
\quad \land \quad \text{ilessp}\,(\mathtt{1},\ e) \\
\quad \land \quad \text{ilessp}\,(\mathtt{0},\ j) \\
\quad \land \quad \text{ilessp}\,(j,\ e)) \\
\to \quad (\text{car}\,(\text{v\&c-apply\$}\,(\mathtt{'iterk}, \\
\qquad\qquad\qquad\qquad\quad \text{list}\,(\text{cons}\,(a,\ \mathtt{0}), \\
\qquad\qquad\qquad\qquad\qquad\quad \text{cons}\,(b,\ \mathtt{0}), \\
\qquad\qquad\qquad\qquad\qquad\quad \text{cons}\,(c,\ \mathtt{0}), \\
\qquad\qquad\qquad\qquad\qquad\quad \text{cons}\,(d,\ \mathtt{0}), \\
\qquad\qquad\qquad\qquad\qquad\quad \text{cons}\,(e,\ \mathtt{0}), \\
\qquad\qquad\qquad\qquad\qquad\quad \text{cons}\,(x,\ \mathtt{0})))) \\
\qquad = \quad \text{car}\,(\text{v\&c-apply\$}\,(\mathtt{'iterk}, \\
\qquad\qquad\qquad\qquad\qquad\quad \text{list}\,(\text{cons}\,(a,\ \mathtt{0}), \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{cons}\,(b,\ \mathtt{0}), \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{cons}\,(c,\ \mathtt{0}), \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \text{cons}\,(d,\ \mathtt{0}),
\end{array}
$$

```
                                   cons (idifference (e, j), 0),
                                   cons (idifference (x, itimes (b, j)), 0)))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Lemma 15. A special case of Lemma 14.

;  Assume  0 < b, 1 < c, 0 < d, and  a >= x.

;  Then

;  IterK( a,b,c,d,1,x ) exists
;   iff
;  IterK( a,b,c,d,1,x+d-(c-1)b ) exists

;  and

;  IterK( a,b,c,d,1,x )
;   =
;  IterK( a,b,c,d,1,x+d-(c-1)b ).

; Proof.  By Lemma 14 with e = c, j = c-1, and
;         x replaced by  x + d.
```

Theorem: iterk_x-iff-iterk_x+d_c-1_b-when-0<b&1<c&0<d&a>=x

```
(ilessp (0, b) ∧ ilessp (1, c) ∧ ilessp (0, d) ∧ (¬ ilessp (a, x)))
 →   (v&c-apply$ ('iterk,
                  list (cons (a, 0),
                        cons (b, 0),
                        cons (c, 0),
                        cons (d, 0),
                        '(1 . 0),
                        cons (x, 0)))
        ↔   v&c-apply$ ('iterk,
                        list (cons (a, 0),
                              cons (b, 0),
                              cons (c, 0),
                              cons (d, 0),
                              '(1 . 0),
                              cons (iplus (x,
                                           idifference (d,
                                                        itimes (b, iplus (-1, c)))),
                                    0))))
```

Theorem: iterk_x=iterk_x+d_c-1_b-when-0<b&1<c&0<d&a>=x

```
(ilessp (0, b) ∧ ilessp (1, c) ∧ ilessp (0, d) ∧ (¬ ilessp (a, x)))
→   (car (v&c-apply$ ('iterk,
                      list (cons (a, 0),
                            cons (b, 0),
                            cons (c, 0),
                            cons (d, 0),
                            '(1 . 0),
                            cons (x, 0))))
        =   car (v&c-apply$ ('iterk,
                             list (cons (a, 0),
                                   cons (b, 0),
                                   cons (c, 0),
                                   cons (d, 0),
                                   '(1 . 0),
                                   cons (iplus (x,
                                                idifference (d,
                                                             itimes (b,
                                                                     iplus (-1, c)))),
                                         0)))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; The next two events give a version of Part 6  of
; the Main Theorem given above in the introduction.

;  6. If  x <= a, d > 0, c > 1, and b > 0, then
;     K( a,b,c,d,x ) exists if and only if
;     K( a,b,c,d,x+d-(c-1)b )  exists and
;     K( a,b,c,d,x ) = K( a,b,c,d,x+d-(c-1)b ).

;     Proof. By Lemma 15 and the definition of K.
```

THEOREM: k_x-iff-k_x+d_c-1_b-when-0<b&1<c&0<d&a>=x

```
(ilessp (0, b) ∧ ilessp (1, c) ∧ ilessp (0, d) ∧ (¬ ilessp (a, x)))
→   (v&c-apply$ ('k,
                 list (cons (a, 0), cons (b, 0), cons (c, 0), cons (d, 0), cons (x, 0)))
        ↔   v&c-apply$ ('k,
                        list (cons (a, 0),
                              cons (b, 0),
                              cons (c, 0),
                              cons (d, 0),
                              cons (iplus (x,
                                           idifference (d,
                                                        itimes (b, iplus (-1, c)))),
                                    0))))
```

THEOREM: k_x=k_x+d_c-1_b-when-0<b&1<c&0<d&a>=x

```
(ilessp (0, b) ∧ ilessp (1, c) ∧ ilessp (0, d) ∧ (¬ ilessp (a, x)))
 →    (car (v&c-apply$ ('k,
                        list (cons (a, 0),
                              cons (b, 0),
                              cons (c, 0),
                              cons (d, 0),
                              cons (x, 0))))
        =    car (v&c-apply$ ('k,
                              list (cons (a, 0),
                                    cons (b, 0),
                                    cons (c, 0),
                                    cons (d, 0),
                                    cons (iplus (x,
                                                 idifference (d,
                                                              itimes (b,
                                                                      iplus (-1, c)))),
                                          0)))))
```

THEOREM: k_x-iff-k_x+d_c-1_b-when-0<b&1<c&0<d&a>=x-version-2

```
(vc-a
 ∧   vc-b
 ∧   vc-c
 ∧   vc-d
 ∧   vc-x
 ∧   ilessp (0, car (vc-b))
 ∧   ilessp (1, car (vc-c))
 ∧   ilessp (0, car (vc-d))
 ∧   (¬ ilessp (car (vc-a), car (vc-x))))
 →   (v&c-apply$ ('k, list (vc-a, vc-b, vc-c, vc-d, vc-x))
       ↔   v&c-apply$ ('k,
                       list (cons (car (vc-a), 0),
                             cons (car (vc-b), 0),
                             cons (car (vc-c), 0),
                             cons (car (vc-d), 0),
                             cons (iplus (car (vc-x),
                                          idifference (car (vc-d),
                                                       itimes (car (vc-b),
                                                               iplus (-1,
                                                                      car (vc-c))))),
                                   0))))
```

THEOREM: k_x=k_x+d_c-1_b-when-0<b&1<c&0<d&a>=x-version-2

```
(vc-a
  ∧   vc-b
  ∧   vc-c
  ∧   vc-d
  ∧   vc-x
  ∧   ilessp (0, car (vc-b))
  ∧   ilessp (1, car (vc-c))
  ∧   ilessp (0, car (vc-d))
  ∧   (¬ ilessp (car (vc-a), car (vc-x))))
 →   (car (v&c-apply$ ('k, list (vc-a, vc-b, vc-c, vc-d, vc-x)))
      =   car (v&c-apply$ ('k,
                         list (cons (car (vc-a), 0),
                             cons (car (vc-b), 0),
                             cons (car (vc-c), 0),
                             cons (car (vc-d), 0),
                             cons (iplus (car (vc-x),
                                        idifference (car (vc-d),
                                                  itimes (car (vc-b),
                                                         iplus (-1,
                                                               car (vc-c))))),
                                   0)))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; The next two events give a more general version
; of Part 6  of the Main Theorem.
```

THEOREM: k_x-iff-k_x+d_c-1_b-when-0<b&1<c&0<d&a>=x-version-3

```
(vc-a
  ∧   vc-b
  ∧   vc-c
  ∧   vc-d
  ∧   vc-x
  ∧   ilessp (0, car (vc-b))
  ∧   ilessp (1, car (vc-c))
  ∧   ilessp (0, car (vc-d))
  ∧   (¬ ilessp (car (vc-a), car (vc-x))))
 →   (v&c-apply$ ('k, list (vc-a, vc-b, vc-c, vc-d, vc-x))
      ↔   v&c-apply$ ('k,
                     list (vc-a,
                         vc-b,
                         vc-c,
                         vc-d,
                         cons (iplus (car (vc-x),
                                    idifference (car (vc-d),
                                         itimes (car (vc-b),
                                                 iplus (-1,
                                                        car (vc-c))))),
                              cost))))
```

THEOREM: k_x=k_x+d_c-1_b-when-0<b&1<c&0<d&a>=x-version-3

```
 (vc-a
  ∧   vc-b
  ∧   vc-c
  ∧   vc-d
  ∧   vc-x
  ∧   ilessp (0, car (vc-b))
  ∧   ilessp (1, car (vc-c))
  ∧   ilessp (0, car (vc-d))
  ∧   (¬ ilessp (car (vc-a), car (vc-x))))
 →    (car (v&c-apply$ ('k, list (vc-a, vc-b, vc-c, vc-d, vc-x)))
        =   car (v&c-apply$ ('k,
                             list (vc-a,
                                   vc-b,
                                   vc-c,
                                   vc-d,
                                   cons (iplus (car (vc-x),
                                                idifference (car (vc-d),
                                                             itimes (car (vc-b),
                                                                     iplus (-1,
                                                                            car (vc-c))))),
                                         cost)))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Lemma 16. The parameter c can be replaced by 1 if
;           the parameter d is modified in a suitable
;           way, when certain restrictions are placed
;           on the parameters.

;  Assume  0 < b, 1 < c, 0 < d, and  (c-1)b < d.

;  Then

;  IterK( a,b,c,d,1,x ) exists
;   iff
;  IterK( a,b,1,d-(c-1)b,1,x ) exists

;  and
;  IterK( a,b,c,d,1,x )
;   =
;  IterK( a,b,1,d-(c-1)b,1,x ).

; Proof.  By induction on the value given by

;          if  x > a  then  0
;                     else  1 + a - x.

;          Also use Lemma 15.
```

THEOREM: k-measure_x+d-_c-1_b<k-measure_x-when-_c-1_b<d&a>=x

```
 (ilessp (itimes (b, c), iplus (b, d)) ∧ (¬ ilessp (a, x)))
 →    (k-measure (a, iplus (b, iplus (d, iplus (x, ineg (itimes (b, c))))))
         <    k-measure (a, x))
```

DEFINITION:

```
induct-hint-1-k-measure (a, b, c, d, x)
=    if ¬ ilessp (itimes (b, iplus (-1, c)), d)  then t
     elseif ilessp (a, x)  then t
     else induct-hint-1-k-measure (a,
                                   b,
                                   c,
                                   d,
                                   iplus (x,
                                          idifference (d,
                                                       itimes (b,
                                                               iplus (-1,
                                                                      c))))) endif
```

THEOREM: iterk_c&d-iff-iterk_1&d-c-1_b-when-0<b&1<c&0<d&b_c-1_<d

```
 (ilessp (0, b)
  ∧   ilessp (1, c)
  ∧   ilessp (0, d)
  ∧   ilessp (itimes (b, iplus (-1, c)), d))
 →    (v&c-apply$ ('iterk,
                   list (cons (a, 0),
                         cons (b, 0),
                         cons (c, 0),
                         cons (d, 0),
                         '(1 . 0),
                         cons (x, 0)))
         ↔    v&c-apply$ ('iterk,
                          list (cons (a, 0),
                              cons (b, 0),
                              '(1 . 0),
                              cons (idifference (d, itimes (b, iplus (-1, c))), 0),
                              '(1 . 0),
                              cons (x, 0))))
```

THEOREM: iterk_c&d=iterk_1&d-c-1_b-when-0<b&1<c&0<d&b_c-1_<d

```
 (ilessp (0, b)
  ∧   ilessp (1, c)
  ∧   ilessp (0, d)
  ∧   ilessp (itimes (b, iplus (-1, c)), d))
 →    (car (v&c-apply$ ('iterk,
                        list (cons (a, 0),
                              cons (b, 0),
                              cons (c, 0),
                              cons (d, 0),
                              '(1 . 0),
                              cons (x, 0))))
        =    car (v&c-apply$ ('iterk,
                              list (cons (a, 0),
                                    cons (b, 0),
                                    '(1 . 0),
                                    cons (idifference (d, itimes (b, iplus (-1, c))),
                                          0),
                                    '(1 . 0),
                                    cons (x, 0)))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; The next two events give a version of Part 7  of
; the Main Theorem given above in the introduction.

;  7. If  d > 0, c > 1, b > 0, and (c-1)b < d, then
;     K( a,b,c,d,x )  exists if and only if
;     K( a,b,1,d-(c-1)b,x ) exists and
;     K( a,b,c,d,x ) = K( a,b,1,d-(c-1)b,x ).

;     Proof. By Lemma 16.
```

THEOREM: k_c&d-iff-k_1&d-c-1_b-when-0<b&1<c&0<d&b_c-1_<d

```
 (ilessp (0, b)
  ∧   ilessp (1, c)
  ∧   ilessp (0, d)
  ∧   ilessp (itimes (b, iplus (-1, c)), d))
 →    (v&c-apply$ ('k,
                   list (cons (a, 0), cons (b, 0), cons (c, 0), cons (d, 0), cons (x, 0)))
     ↔    v&c-apply$ ('k,
                      list (cons (a, 0),
                            cons (b, 0),
                            '(1 . 0),
                            cons (idifference (d, itimes (b, iplus (-1, c))), 0),
                            cons (x, 0))))
```

THEOREM: k_c&d=k_1&d-c-1_b-when-0<b&1<c&0<d&b_c-1_<d

```
 (ilessp (0, b)
  ∧    ilessp (1, c)
  ∧    ilessp (0, d)
  ∧    ilessp (itimes (b, iplus (-1, c)), d))
 →     (car (v&c-apply$ ('k,
                        list (cons (a, 0),
                              cons (b, 0),
                              cons (c, 0),
                              cons (d, 0),
                              cons (x, 0))))
        =    car (v&c-apply$ ('k,
                             list (cons (a, 0),
                                   cons (b, 0),
                                   '(1 . 0),
                                   cons (idifference (d, itimes (b, iplus (-1, c))),
                                         0),
                                   cons (x, 0)))))
```

THEOREM: k_c&d-iff-k_1&d-c-1_b-when-0<b&1<c&0<d&b_c-1_<d-version-2

```
 (vc-a
  ∧    vc-b
  ∧    vc-c
  ∧    vc-d
  ∧    vc-x
  ∧    ilessp (0, car (vc-b))
  ∧    ilessp (1, car (vc-c))
  ∧    ilessp (0, car (vc-d))
  ∧    ilessp (itimes (car (vc-b), iplus (-1, car (vc-c))), car (vc-d)))
 →     (v&c-apply$ ('k, list (vc-a, vc-b, vc-c, vc-d, vc-x))
        ↔    v&c-apply$ ('k,
                        list (cons (car (vc-a), 0),
                              cons (car (vc-b), 0),
                              '(1 . 0),
                              cons (idifference (car (vc-d),
                                                 itimes (car (vc-b),
                                        iplus (-1, car (vc-c)))),
                   0),
               cons (car (vc-x), 0))))
```

THEOREM: k_c&d=k_1&d-c-1_b-when-0<b&1<c&0<d&b_c-1_<d-version-2

```
 (vc-a
  ∧   vc-b
  ∧   vc-c
  ∧   vc-d
  ∧   vc-x
  ∧   ilessp (0, car (vc-b))
  ∧   ilessp (1, car (vc-c))
  ∧   ilessp (0, car (vc-d))
  ∧   ilessp (itimes (car (vc-b), iplus (-1, car (vc-c))), car (vc-d)))
 →    (car (v&c-apply$ ('k, list (vc-a, vc-b, vc-c, vc-d, vc-x)))
        =    car (v&c-apply$ ('k,
                              list (cons (car (vc-a), 0),
                                    cons (car (vc-b), 0),
                                    '(1 . 0),
                                    cons (idifference (car (vc-d),
                                                       itimes (car (vc-b),
                                                               iplus (-1,
                                                                      car (vc-c)))),
                                          0),
                                    cons (car (vc-x), 0)))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; The next two events give a more general version
; of Part 7  of the Main Theorem.
```

THEOREM: k_c&d-iff-k_1&d-c-1_b-when-0<b&1<c&0<d&b_c-1_<d-version-3

```
 (vc-a
  ∧   vc-b
  ∧   vc-c
  ∧   vc-d
  ∧   vc-x
  ∧   ilessp (0, car (vc-b))
  ∧   ilessp (1, car (vc-c))
  ∧   ilessp (0, car (vc-d))
  ∧   ilessp (itimes (car (vc-b), iplus (-1, car (vc-c))), car (vc-d)))
 →    (v&c-apply$ ('k, list (vc-a, vc-b, vc-c, vc-d, vc-x))
        ↔    v&c-apply$ ('k,
                         list (vc-a,
                               vc-b,
                               cons (1, cost1),
                               cons (idifference (car (vc-d),
                                                  itimes (car (vc-b),
                                                          iplus (-1, car (vc-c)))),
                                     cost2),
                               vc-x)))
```

THEOREM: k_c&d=k_1&d-c-1_b-when-0<b&1<c&0<d&b_c-1_<d-version-3

```
(vc-a
 ∧   vc-b
 ∧   vc-c
 ∧   vc-d
 ∧   vc-x
 ∧   ilessp (0, car (vc-b))
 ∧   ilessp (1, car (vc-c))
 ∧   ilessp (0, car (vc-d))
 ∧   ilessp (itimes (car (vc-b), iplus (-1, car (vc-c))), car (vc-d)))
→    (car (v&c-apply$ ('k, list (vc-a, vc-b, vc-c, vc-d, vc-x)))
        =  car (v&c-apply$ ('k,
                            list (vc-a,
                                  vc-b,
                                  cons (1, cost1),
                                  cons (idifference (car (vc-d),
                                                     itimes (car (vc-b),
                                                             iplus (-1,
                                                                    car (vc-c)))),
                                        cost2),
                                  vc-x))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Lemma 17. Essentially the if-part of Knuth's
;           theorem characterizing the parameters
;           when K is total.

;  Assume  0 < b, 1 < c, 0 < d, and  (c-1)b < d.

;  Then IterK( a,b,c,d,1,x ) exists.

; Proof.  By Lemma 4 and Lemma 16.
```

THEOREM: iterk-exists-when-0<b&1<c&0<d&b_c-1_<d

```
(ilessp (0, b)
 ∧   ilessp (1, c)
 ∧   ilessp (0, d)
 ∧   ilessp (itimes (b, iplus (-1, c)), d))
→    v&c-apply$ ('iterk,
                list (cons (a, 0),
                      cons (b, 0),
                      cons (c, 0),
                      cons (d, 0),
                      '(1 . 0),
                      cons (x, 0)))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; The next two events are versions of Part 8  of the
; Main Theorem given above in the introduction.

;  8. If  d > 0, c > 1, b > 0, and (c-1)b < d, then
;     K( a,b,c,d,x ) exists.

;     Proof. By Lemma 17.
```

Theorem: k-exists-when-0<b&1<c&0<d&b_c-1_<d

```
(ilessp (0, b)
 ∧   ilessp (1, c)
 ∧   ilessp (0, d)
 ∧   ilessp (itimes (b, iplus (-1, c)), d))
→    v&c-apply$ ('k,
                list (cons (a, 0), cons (b, 0), cons (c, 0), cons (d, 0), cons (x, 0)))
```

Theorem: k-halts-when-0<b&1<c&0<d&b_c-1_<d

```
(vc-a
 ∧   vc-b
 ∧   vc-c
 ∧   vc-d
 ∧   vc-x
 ∧   ilessp (0, car (vc-b))
 ∧   ilessp (1, car (vc-c))
 ∧   ilessp (0, car (vc-d))
 ∧   ilessp (itimes (car (vc-b), iplus (-1, car (vc-c))), car (vc-d)))
→    v&c-apply$ ('k, list (vc-a, vc-b, vc-c, vc-d, vc-x))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Lemma 18. The cost of computing K is lowered when d is
;           added to x.

;  Assume  1 < c, a >= x, and IterK( a,b,c,d,1,x ) exists.
;  Then IterK( a,b,c,d,1,x+d ) exists

;   and

;  cost[ IterK( a,b,c,d,1,x+d ) ]
;   <
;  cost[ IterK( a,b,c,d,1,x ) ]

; Proof. By the definition of IterK and Lemma 2.
```

Theorem: iterk-x+d-exists-when-iterk-exists&1<c&a>=x

```
 (ilessp (1, c)
  ∧   (¬ ilessp (a, x))
  ∧   v&c-apply$ ('iterk,
                   list (cons (a, 0),
                         cons (b, 0),
                         cons (c, 0),
                         cons (d, 0),
                         '(1 . 0),
                         cons (x, 0))))
 →   v&c-apply$ ('iterk,
                   list (cons (a, 0),
                         cons (b, 0),
                         cons (c, 0),
                         cons (d, 0),
                         '(1 . 0),
                         cons (iplus (d, x), 0)))
```

Theorem: iterk-x+d-cost<cost-iterk-when-iterk-exists&1<c&a>=x

```
 (ilessp (1, c)
  ∧   (¬ ilessp (a, x))
  ∧   v&c-apply$ ('iterk,
                   list (cons (a, 0),
                         cons (b, 0),
                         cons (c, 0),
                         cons (d, 0),
                         '(1 . 0),
                         cons (x, 0))))
 →   (cdr (v&c-apply$ ('iterk,
                         list (cons (a, 0),
                               cons (b, 0),
                               cons (c, 0),
                               cons (d, 0),
                           '(1 . 0),
                           cons (iplus (d, x), 0))))
         <    cdr (v&c-apply$ ('iterk,
                           list (cons (a, 0),
                                 cons (b, 0),
                                 cons (c, 0),
                                 cons (d, 0),
                                 '(1 . 0),
                                 cons (x, 0)))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Lemma 19. The cost of computing K is lowered
;           when K is iterated c times and
;           "enough" d's are added to x.

;  Assume  1 < c, a >= x, 0 < d, and
;  IterK( a,b,c,d,1,x ) exists.

;  Then IterK( a,b,c,d,c,x+N( a,d,x )d ) exists

;   and

;  cost[ IterK( a,b,c,d,c,x+N( a,d,x )d ) ]
;   <
;  cost[ IterK( a,b,c,d,1,x ) ]

; Proof. Hold a fixed and induct on the
;        value given by

;        if  x > a  then  0
;                   else  1 + a - x.
```

Theorem: n=1-when-x+d>a&a>=x&d>0

$(\text{ilessp}\,(0,\ d) \land (\neg\ \text{ilessp}\,(a,\ x)) \land \text{ilessp}\,(a,\ \text{iplus}\,(d,\ x)))$
$\rightarrow\quad (\text{n}\,(a,\ d,\ x) = 1)$

Definition:

```
induct-hint-2-k-measure (a, d, x)
=   if ¬ ilessp (0, d)  then t
    elseif ilessp (a, x)  then t
    elseif ilessp (a, iplus (d, x))  then t
    else induct-hint-2-k-measure (a, d, iplus (d, x)) endif
```

Theorem: iterk-x+nd-exists-when-iterk-exists&1<c&a>=x&0<d

```
(ilessp (1, c)
 ∧   ilessp (0, d)
 ∧   (¬ ilessp (a, x))
 ∧   v&c-apply$ ('iterk,
                  list (cons (a, 0),
                        cons (b, 0),
                        cons (c, 0),
                        cons (d, 0),
                        '(1 . 0),
                        cons (x, 0))))
→    v&c-apply$ ('iterk,
                  list (cons (a, 0),
                        cons (b, 0),
                        cons (c, 0),
                        cons (d, 0),
                        cons (c, 0),
                        cons (iplus (x, itimes (d, n (a, d, x))), 0)))
```

THEOREM: iterk-x+nd-cost<cost-iterk-when-iterk-exists&1<c&a>=x&0<d

```
(ilessp (1, c)
 ∧   ilessp (0, d)
 ∧   (¬ ilessp (a, x))
 ∧   v&c-apply$ ('iterk,
                  list (cons (a, 0),
                        cons (b, 0),
                        cons (c, 0),
                        cons (d, 0),
                        '(1 . 0),
                        cons (x, 0))))
→    (cdr (v&c-apply$ ('iterk,
                        list (cons (a, 0),
                              cons (b, 0),
                              cons (c, 0),
                              cons (d, 0),
                              cons (c, 0),
                              cons (iplus (x, itimes (d, n (a, d, x))), 0))))
      <    cdr (v&c-apply$ ('iterk,
                             list (cons (a, 0),
                                   cons (b, 0),
                                   cons (c, 0),
                                   cons (d, 0),
                                   '(1 . 0),
                                   cons (x, 0)))))
```

;;

```
; Lemma 20. The number of iterates of K can be
;           reduced to 1 if "enough" b's are
;           sutracted from x.

;  Assume  0 < b, 1 < c, 0 < d, 1 < e, 0 < i < e,
;  and IterK( a,b,c,d,e,x ) exists.

;  Then IterK( a,b,c,d,1,x-ib ) exists.

; Proof. By Lemma 14 and Lemma 2.
```

THEOREM: e-i=1-when-1+i>=e&i<e

```
((¬ ilessp (iplus (1, i), e)) ∧ ilessp (i, e)) → (iplus (e, ineg (i)) = 1)
```

THEOREM: iterk_1_x-ib-exists-when-iterk_e_x-exists&0<b&1<c&0<d&1<e&0<i<e

```
(ilessp (0, b)
 ∧   ilessp (1, c)
 ∧   ilessp (0, d)
 ∧   ilessp (1, e)
 ∧   ilessp (0, i)
 ∧   ilessp (i, e)
 ∧   v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       cons (e, 0),
                       cons (x, 0))))
→   v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       '(1 . 0),
                       cons (idifference (x, itimes (b, i)), 0)))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Lemma 21. Another lemma reducing the number of
;           iterates of K and subtracting b's
;           from x.

;  Assume  0 < b, 1 < c, 0 < d, 0 < i, a+ib < x+b,
;  and IterK( a,b,c,d,i+1,x ) exists.
;  Then IterK( a,b,c,d,1,x-ib ) exists

;   and

;  cost[ IterK( a,b,c,d,1,x-ib ) ]
;   <
;  cost[ IterK( a,b,c,d,i+1,x ) ]

; Proof. By Lemma 20 (for the exists portion)
;        and induction on i (for the cost portion).
```

THEOREM: iterk_1_x-ib_cost<cost-iterk_i+1_x-base-case

```
 (ilessp (0, b)
  ∧   ilessp (1, c)
  ∧   ilessp (0, d)
  ∧   (i = 1)
  ∧   ilessp (iplus (a, itimes (b, i)), iplus (x, b))
  ∧   v&c-apply$ ('iterk,
                  list (cons (a, 0),
                        cons (b, 0),
                        cons (c, 0),
                        cons (d, 0),
                        cons (iplus (1, i), 0),
                        cons (x, 0))))
 →    (cdr (v&c-apply$ ('iterk,
                        list (cons (a, 0),
                              cons (b, 0),
                              cons (c, 0),
                              cons (d, 0),
                              '(1 . 0),
                              cons (idifference (x, itimes (b, i)), 0))))
         <    cdr (v&c-apply$ ('iterk,
                               list (cons (a, 0),
                                     cons (b, 0),
                                     cons (c, 0),
                                     cons (d, 0),
                                     cons (iplus (1, i), 0),
                                     cons (x, 0)))))
```

THEOREM: fix-int-x=x-when-0<x

```
 ilessp (0, x) → (fix-int (x) = x)
```

THEOREM: iterk_1_x-ib_cost<cost-iterk_i+1_x-step-1

```
 (ilessp (0, i)
 ∧   v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       cons (iplus (1, i), 0),
                       cons (x, 0))))
→   (cdr (v&c-apply$ ('iterk,
                      list (cons (a, 0),
                            cons (b, 0),
                            cons (c, 0),
                            cons (d, 0),
                            '(1 . 0),
                            v&c-apply$ ('iterk,
                                        list (cons (a, 0),
                                              cons (b, 0),
                                              cons (c, 0),
                                              cons (d, 0),
                                              cons (i, 0),
                                              cons (x, 0))))))
       <   cdr (v&c-apply$ ('iterk,
                            list (cons (a, 0),
                                  cons (b, 0),
                                  cons (c, 0),
                                  cons (d, 0),
                                  cons (iplus (1, i), 0),
                                  cons (x, 0)))))
```

Theorem: iterk_i_x-iff-iterk_1_x-ib-when-0<b&1<c&0<d&i>0&i<>1

```
 (ilessp (0, b) ∧ ilessp (1, c) ∧ ilessp (0, d) ∧ ilessp (0, i) ∧ (i ≠ 1))
→   (v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       cons (i, 0),
                       cons (x, 0)))
       ↔   v&c-apply$ ('iterk,
                       list (cons (a, 0),
                             cons (b, 0),
                             cons (c, 0),
                             cons (d, 0),
                             '(1 . 0),
                             cons (iplus (x, ineg (itimes (b, iplus (-1, i)))), 0))))
```

THEOREM: iterk_i_x=iterk_1_x-ib-when-0<b&1<c&0<d&i>0&i<>1

```
 (ilessp (0, b) ∧ ilessp (1, c) ∧ ilessp (0, d) ∧ ilessp (0, i) ∧ (i ≠ 1))
 →    (car (v&c-apply$ ('iterk,
                         list (cons (a, 0),
                               cons (b, 0),
                               cons (c, 0),
                               cons (d, 0),
                               cons (i, 0),
                               cons (x, 0))))
          =    car (v&c-apply$ ('iterk,
                               list (cons (a, 0),
                                     cons (b, 0),
                                     cons (c, 0),
                                     cons (d, 0),
                                     '(1 . 0),
                                     cons (iplus (x, ineg (itimes (b, iplus (-1, i)))),
                                           0)))))
```

THEOREM: iterk_1_x-ib_cost<cost-iterk_i+1_x-exists-step-2

```
 (ilessp (0, b) ∧ ilessp (1, c) ∧ ilessp (0, d) ∧ ilessp (0, i) ∧ (i ≠ 1))
 →    (v&c-apply$ ('iterk,
                    list (cons (a, 0),
                          cons (b, 0),
                          cons (c, 0),
                          cons (d, 0),
                          '(1 . 0),
                          v&c-apply$ ('iterk,
                                       list (cons (a, 0),
                                             cons (b, 0),
                                             cons (c, 0),
                                             cons (d, 0),
                                             cons (i, 0),
                                             cons (x, 0)))))
          ↔    v&c-apply$ ('iterk,
                          list (cons (a, 0),
                                cons (b, 0),
                                cons (c, 0),
                                cons (d, 0),
                                '(1 . 0),
                                v&c-apply$ ('iterk,
                                             list (cons (a, 0),
                                                   cons (b, 0),
                                                   cons (c, 0),
                                                   cons (d, 0),
                                          '(1 . 0),
                                          cons (iplus (x,
                                                     ineg (itimes (b,
                                                                iplus (-1,
                                                                      i)))),
                                               0))))))

THEOREM: iterk_1_x-ib_cost<cost-iterk_i+1_x-step-2
 (ilessp (0, b)
  ∧   ilessp (1, c)
  ∧   ilessp (0, d)
  ∧   ilessp (0, i)
  ∧   (i ≠ 1)
  ∧   v&c-apply$ ('iterk,
                  list (cons (a, 0),
                        cons (b, 0),
                        cons (c, 0),
                        cons (d, 0),
                        '(1 . 0),
                        v&c-apply$ ('iterk,
                                    list (cons (a, 0),
                                          cons (b, 0),
                                          cons (c, 0),
                                          cons (d, 0),
                                          cons (i, 0),
                                          cons (x, 0)))))
  ∧   v&c-apply$ ('iterk,
                  list (cons (a, 0),
                        cons (b, 0),
                        cons (c, 0),
                        cons (d, 0),
                        '(1 . 0),
                        v&c-apply$ ('iterk,
                                    list (cons (a, 0),
                                          cons (b, 0),
                                          cons (c, 0),
                                          cons (d, 0),
                                          '(1 . 0),
                                          cons (iplus (x,
                                                     ineg (itimes (b,
                                                                iplus (-1, i)))),
                                               0)))))
  ∧   (cdr (v&c-apply$ ('iterk,
                        list (cons (a, 0),
                                   cons (b, 0),
                                   cons (c, 0),
                                   cons (d, 0),
                                   '(1 . 0),
                                   cons (iplus (x, ineg (itimes (b, iplus (-1, i)))), 0))))
       <   cdr (v&c-apply$ ('iterk,
                                   list (cons (a, 0),
                                         cons (b, 0),
                                         cons (c, 0),
                                         cons (d, 0),
                                         cons (i, 0),
                                         cons (x, 0))))))
   →   (cdr (v&c-apply$ ('iterk,
                              list (cons (a, 0),
                                    cons (b, 0),
                                    cons (c, 0),
                                    cons (d, 0),
                                    '(1 . 0),
                                    v&c-apply$ ('iterk,
                                                list (cons (a, 0),
                                                      cons (b, 0),
                                                      cons (c, 0),
                                                      cons (d, 0),
                                                      '(1 . 0),
                                                      cons (iplus (x,
                                                                   ineg (itimes (b,
                                                                                 iplus (-1,
                                                                                        i)))),
                                                            0))))))
       <   cdr (v&c-apply$ ('iterk,
                                   list (cons (a, 0),
                                         cons (b, 0),
                                         cons (c, 0),
                                         cons (d, 0),
                                         '(1 . 0),
                                         v&c-apply$ ('iterk,
                                                     list (cons (a, 0),
                                                           cons (b, 0),
                                                           cons (c, 0),
                                                           cons (d, 0),
                                                           cons (i, 0),
                                                           cons (x, 0)))))))
```

THEOREM: iterk_i+1_exists-iff-iterk_1_iterk_i_exists

```
(ilessp (0, i) ∧ (i ≠ 1))
→   (v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       cons (iplus (1, i), 0),
                       cons (x, 0)))
      ↔   v&c-apply$ ('iterk,
                       list (cons (a, 0),
                             cons (b, 0),
                             cons (c, 0),
                             cons (d, 0),
                             '(1 . 0),
                             v&c-apply$ ('iterk,
                                          list (cons (a, 0),
                                                cons (b, 0),
                                                cons (c, 0),
                                                cons (d, 0),
                                                cons (i, 0),
                                                cons (x, 0))))))
```

Theorem: iterk_1_x-ib_cost<cost-iterk_i+1_x-step-1&2

```
(ilessp (0, b)
 ∧   ilessp (1, c)
 ∧   ilessp (0, d)
 ∧   ilessp (0, i)
 ∧   (i ≠ 1)
 ∧   v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       cons (iplus (1, i), 0),
                       cons (x, 0)))
 ∧   (cdr (v&c-apply$ ('iterk,
                       list (cons (a, 0),
                             cons (b, 0),
                             cons (c, 0),
                             cons (d, 0),
                             '(1 . 0),
                             cons (iplus (x, ineg (itimes (b, iplus (-1, i)))), 0))))
      <   cdr (v&c-apply$ ('iterk,
                             list (cons (a, 0),
                             cons (b, 0),
                             cons (c, 0),
                             cons (d, 0),
                             cons (i, 0),
                             cons (x, 0))))))
→    (cdr (v&c-apply$ ('iterk,
                     list (cons (a, 0),
                         cons (b, 0),
                         cons (c, 0),
                         cons (d, 0),
                         '(1 . 0),
                         v&c-apply$ ('iterk,
                                     list (cons (a, 0),
                                         cons (b, 0),
                                         cons (c, 0),
                                         cons (d, 0),
                                         '(1 . 0),
                                         cons (iplus (x,
                                                    ineg (itimes (b,
                                                               iplus (-1,
                                                                     i)))),
                                              0))))))
     <   cdr (v&c-apply$ ('iterk,
                         list (cons (a, 0),
                             cons (b, 0),
                             cons (c, 0),
                             cons (d, 0),
                             cons (iplus (1, i), 0),
                             cons (x, 0)))))
```

Theorem: iterk_1_x-ib_cost<cost-iterk_i+1_x-exists-step-3

```
(ilessp (0, b)
 ∧   ilessp (1, c)
 ∧   ilessp (0, d)
 ∧   ilessp (0, i)
 ∧   (i ≠ 1)
 ∧   ilessp (iplus (a, itimes (b, i)), iplus (x, b)))
→    (v&c-apply$ ('iterk,
                 list (cons (a, 0),
                     cons (b, 0),
                     cons (c, 0),
                     cons (d, 0),
                     '(1 . 0),
                     v&c-apply$ ('iterk,
                                list (cons (a, 0),
                                      cons (b, 0),
                                      cons (c, 0),
                                      cons (d, 0),
                                      '(1 . 0),
                                      cons (iplus (x,
                                                   ineg (itimes (b,
                                                                 iplus (-1, i)))),
                                            0)))))
↔   v&c-apply$ ('iterk,
                list (cons (a, 0),
                      cons (b, 0),
                      cons (c, 0),
                      cons (d, 0),
                      '(1 . 0),
                      cons (iplus (x, ineg (itimes (b, i))), 0))))
```

Theorem: iterk_1_x-ib_cost<cost-iterk_i+1_x-step-3

```
(ilessp (0, b)
 ∧   ilessp (1, c)
 ∧   ilessp (0, d)
 ∧   ilessp (0, i)
 ∧   (i ≠ 1)
 ∧   ilessp (iplus (a, itimes (b, i)), iplus (x, b))
 ∧   v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       '(1 . 0),
                       cons (iplus (x, ineg (itimes (b, i))), 0)))
 ∧   v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       '(1 . 0),
                       v&c-apply$ ('iterk,
                                   list (cons (a, 0),
                                         cons (b, 0),
                                         cons (c, 0),
                                         cons (d, 0),
                                         '(1 . 0),
                                         cons (iplus (x,
                                               ineg (itimes (b,
                                                           iplus (-1, i)))),
                                          0))))))
→    (cdr (v&c-apply$ ('iterk,
                     list (cons (a, 0),
                          cons (b, 0),
                          cons (c, 0),
                          cons (d, 0),
                          '(1 . 0),
                          cons (iplus (x, ineg (itimes (b, i))), 0))))
      <    cdr (v&c-apply$ ('iterk,
                         list (cons (a, 0),
                              cons (b, 0),
                              cons (c, 0),
                              cons (d, 0),
                              '(1 . 0),
                              v&c-apply$ ('iterk,
                                          list (cons (a, 0),
                                               cons (b, 0),
                                               cons (c, 0),
                                               cons (d, 0),
                                               '(1 . 0),
                                               cons (iplus (x,
                                                           ineg (itimes (b,
                                                                       iplus (-1,
                                                                             i)))),
                                                    0)))))))
```

THEOREM: iterk_i+1_x-iff-iterk_1_x-ib-when-0<b&1<c&0<d&i>0

```
 (ilessp (0, b) ∧ ilessp (1, c) ∧ ilessp (0, d) ∧ ilessp (0, i))
→    (v&c-apply$ ('iterk,
                list (cons (a, 0),
                     cons (b, 0),
                     cons (c, 0),
                     cons (d, 0),
                     cons (iplus (1, i), 0),
                     cons (x, 0)))
      ↔    v&c-apply$ ('iterk,
                     list (cons (a, 0),
                          cons (b, 0),
                          cons (c, 0),
                          cons (d, 0),
                          '(1 . 0),
                          cons (iplus (x, ineg (itimes (b, i))), 0))))
```

Theorem: iterk_1_x-ib_cost<cost-iterk_i+1_x-step-1&2&3

```
(ilessp (0, b)
 ∧   ilessp (1, c)
 ∧   ilessp (0, d)
 ∧   ilessp (0, i)
 ∧   (i ≠ 1)
 ∧   ilessp (iplus (a, itimes (b, i)), iplus (x, b))
 ∧   v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       cons (iplus (1, i), 0),
                       cons (x, 0)))
 ∧   (cdr (v&c-apply$ ('iterk,
                       list (cons (a, 0),
                             cons (b, 0),
                             cons (c, 0),
                             cons (d, 0),
                             '(1 . 0),
                             cons (iplus (x, ineg (itimes (b, iplus (-1, i)))), 0))))
       <   cdr (v&c-apply$ ('iterk,
                            list (cons (a, 0),
                                  cons (b, 0),
                                  cons (c, 0),
                                  cons (d, 0),
                                  cons (i, 0),
                                  cons (x, 0))))))
→    (cdr (v&c-apply$ ('iterk,
                       list (cons (a, 0),
                             cons (b, 0),
                             cons (c, 0),
                             cons (d, 0),
                             '(1 . 0),
                             cons (iplus (x, ineg (itimes (b, i))), 0))))
       <   cdr (v&c-apply$ ('iterk,
                            list (cons (a, 0),
                                  cons (b, 0),
                                  cons (c, 0),
                                  cons (d, 0),
                                  cons (iplus (1, i), 0),
                                  cons (x, 0)))))
```

Theorem: a+bi<x+b-when-a+bi<b+b+x&0<b

```
(ilessp (0, b) ∧ ilessp (iplus (a, itimes (b, i)), iplus (x, b)))
→   ilessp (iplus (a, itimes (b, i)), iplus (b, iplus (b, x)))
```

THEOREM: iterk-e=i-exists-when-iterk-e=i+1-exists

```
(ilessp (0, i)
 ∧   (i ≠ 1)
 ∧   v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       cons (iplus (1, i), 0),
                       cons (x, 0))))
→   v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       cons (i, 0),
                       cons (x, 0)))
```

THEOREM: iterk_1_x-ib_cost<cost-iterk_i+1_x-induction-step

```
(ilessp (0, b)
 ∧   ilessp (1, c)
 ∧   ilessp (0, d)
 ∧   ilessp (0, i)
 ∧   (i ≠ 1)
 ∧   ilessp (iplus (a, itimes (b, i)), iplus (x, b))
 ∧   v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       cons (iplus (1, i), 0),
                       cons (x, 0)))
 ∧   ((ilessp (0, b)
        ∧   ilessp (1, c)
        ∧   ilessp (0, d)
        ∧   ilessp (0, iplus (-1, i))
        ∧   ilessp (iplus (a, itimes (b, iplus (-1, i))), iplus (x, b))
        ∧   v&c-apply$ ('iterk,
                        list (cons (a, 0),
                              cons (b, 0),
                              cons (c, 0),
                                          cons (d, 0),
                                          cons (iplus (1, iplus (-1, i)), 0),
                                          cons (x, 0))))
              →    (cdr (v&c-apply$ ('iterk,
                                          list (cons (a, 0),
                                                cons (b, 0),
                                                cons (c, 0),
                                                cons (d, 0),
                                                '(1 . 0),
                                                cons (idifference (x,
                                                                   itimes (b, iplus (-1, i))),
                                                      0))))
                   <    cdr (v&c-apply$ ('iterk,
                                               list (cons (a, 0),
                                                     cons (b, 0),
                                                     cons (c, 0),
                                                     cons (d, 0),
                                                     cons (iplus (1, iplus (-1, i)), 0),
                                                     cons (x, 0)))))))
      →    (cdr (v&c-apply$ ('iterk,
                                 list (cons (a, 0),
                                       cons (b, 0),
                                       cons (c, 0),
                                       cons (d, 0),
                                       '(1 . 0),
                                       cons (idifference (x, itimes (b, i)), 0))))
              <    cdr (v&c-apply$ ('iterk,
                                       list (cons (a, 0),
                                             cons (b, 0),
                                             cons (c, 0),
                                             cons (d, 0),
                                             cons (iplus (1, i), 0),
                                             cons (x, 0)))))
```

Theorem: iterk_1_x-ib_cost<cost-iterk_i+1_x

```
 (ilessp (0, b)
   ∧    ilessp (1, c)
   ∧    ilessp (0, d)
   ∧    ilessp (0, i)
   ∧    ilessp (iplus (a, itimes (b, i)), iplus (x, b))
   ∧    v&c-apply$ ('iterk,
                     list (cons (a, 0),
                           cons (b, 0),
                           cons (c, 0),
                               cons (d, 0),
                               cons (iplus (1, i), 0),
                               cons (x, 0))))
 →   (cdr (v&c-apply$ ('iterk,
                        list (cons (a, 0),
                              cons (b, 0),
                              cons (c, 0),
                              cons (d, 0),
                              '(1 . 0),
                              cons (idifference (x, itimes (b, i)), 0))))
        <    cdr (v&c-apply$ ('iterk,
                              list (cons (a, 0),
                                    cons (b, 0),
                                    cons (c, 0),
                                    cons (d, 0),
                                    cons (iplus (1, i), 0),
                                    cons (x, 0)))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Lemma 22. A generalization of Lemma 21.

;  Assume  0 < b, 1 < c, 0 < d, 0 < i, a+ib < x+b,
;  i < e, and IterK( a,b,c,d,e,x ) exists.

;  Then IterK( a,b,c,d,1,x-ib ) exists

;   and

;  cost[ IterK( a,b,c,d,1,x-ib ) ]
;   <
;  cost[ IterK( a,b,c,d,e,x ) ]

; Proof. Lemma 2 and Lemma 21.
```

Theorem: iterk_1_x-ib_cost<cost-iterk_e_x-case-1

```
 (ilessp (0, b)
  ∧   ilessp (1, c)
  ∧   ilessp (0, d)
  ∧   ilessp (0, i)
  ∧   ilessp (iplus (1, i), e)
  ∧   ilessp (iplus (a, itimes (b, i)), iplus (x, b))
  ∧   v&c-apply$ ('iterk,
                  list (cons (a, 0),
                              cons (b, 0),
                              cons (c, 0),
                              cons (d, 0),
                              cons (e, 0),
                              cons (x, 0))))
 →    (cdr (v&c-apply$ ('iterk,
                         list (cons (a, 0),
                               cons (b, 0),
                               cons (c, 0),
                               cons (d, 0),
                               '(1 . 0),
                               cons (idifference (x, itimes (b, i)), 0))))
        <    cdr (v&c-apply$ ('iterk,
                               list (cons (a, 0),
                                     cons (b, 0),
                                     cons (c, 0),
                                     cons (d, 0),
                                     cons (e, 0),
                                     cons (x, 0)))))
```

THEOREM: iterk_1_x-ib_cost<cost-iterk_e_x-case-2

```
 (ilessp (0, b)
  ∧    ilessp (1, c)
  ∧    ilessp (0, d)
  ∧    ilessp (0, i)
  ∧    (iplus (1, i) = e)
  ∧    ilessp (iplus (a, itimes (b, i)), iplus (x, b))
  ∧    v&c-apply$ ('iterk,
                   list (cons (a, 0),
                         cons (b, 0),
                         cons (c, 0),
                         cons (d, 0),
                         cons (e, 0),
                         cons (x, 0))))
 →    (cdr (v&c-apply$ ('iterk,
                         list (cons (a, 0),
                               cons (b, 0),
                               cons (c, 0),
                               cons (d, 0),
                               '(1 . 0),
                               cons (idifference (x, itimes (b, i)), 0))))
        <    cdr (v&c-apply$ ('iterk,
                               list (cons (a, 0),
                                     cons (b, 0),
                                     cons (c, 0),
                                     cons (d, 0),
                                     cons (e, 0),
                                     cons (x, 0)))))
```

THEOREM: e=i+1-or-e>i+1-when-e>i>0

```
 (ilessp (0, i) ∧ ilessp (i, e))
 →    ((e = iplus (1, i)) ∨ ilessp (iplus (1, i), e))
```

THEOREM: iterk_1_x-ib_cost<cost-iterk_e_x

```
 (ilessp (0, b)
  ∧   ilessp (1, c)
  ∧   ilessp (0, d)
  ∧   ilessp (0, i)
  ∧   ilessp (i, e)
  ∧   ilessp (iplus (a, itimes (b, i)), iplus (x, b))
  ∧   v&c-apply$ ('iterk,
                  list (cons (a, 0),
                        cons (b, 0),
                        cons (c, 0),
                        cons (d, 0),
                        cons (e, 0),
                        cons (x, 0))))
 →    (cdr (v&c-apply$ ('iterk,
                        list (cons (a, 0),
                              cons (b, 0),
                              cons (c, 0),
                              cons (d, 0),
                              '(1 . 0),
                              cons (idifference (x, itimes (b, i)), 0))))
          <    cdr (v&c-apply$ ('iterk,
                                list (cons (a, 0),
                                      cons (b, 0),
                                      cons (c, 0),
                                      cons (d, 0),
                                      cons (e, 0),
                                      cons (x, 0)))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Define the function J( a,b,x ) recursively,
;  so that whenever b > 0, J( a,b,x ) is the smallest
;  nonnegative integer i such that x <= a + ib.
```

DEFINITION:

how-far-above-a $(a,\ x)$

$$\begin{aligned} = \quad & \textbf{if}\ \neg\ \text{ilessp}\,(a,\ x)\ \ \textbf{then}\ \texttt{0} \\ & \textbf{else}\ \text{iplus}\,(x,\ \text{ineg}\,(a))\ \textbf{endif} \end{aligned}$$

Theorem: numberp_x-a_when_x>a

$\text{ilessp}\,(a,\ x) \rightarrow (\text{iplus}\,(x,\ \text{ineg}\,(a)) \in \mathbf{N})$

Theorem: x-a-b<x-a-when-0<b&a<x&a+b<x

$(\text{ilessp}\,(\texttt{0},\ b) \land \text{ilessp}\,(a,\ x) \land \text{ilessp}\,(\text{iplus}\,(a,\ b),\ x))$

$\rightarrow \quad (\text{iplus}\,(x,\ \text{iplus}\,(\text{ineg}\,(a),\ \text{ineg}\,(b))) < \text{iplus}\,(x,\ \text{ineg}\,(a)))$

Theorem: a<>x-when-a<x

$\text{ilessp}\,(a,\ x) \rightarrow (\text{fix-int}\,(x) \neq \text{fix-int}\,(a))$

Definition:

j $(a,\ b,\ x)$

$$\begin{aligned} = \quad & \textbf{if}\ \neg\ \text{ilessp}\,(\texttt{0},\ b)\ \ \textbf{then}\ \texttt{0} \\ & \textbf{elseif}\ \neg\ \text{ilessp}\,(a,\ x)\ \ \textbf{then}\ \texttt{0} \\ & \textbf{else}\ \text{iplus}\,(\texttt{1},\ \text{j}\,(a,\ b,\ \text{iplus}\,(x,\ \text{ineg}\,(b))))\ \textbf{endif} \end{aligned}$$

Theorem: j>=0

$\neg\ \text{ilessp}\,(\text{j}\,(a,\ b,\ x),\ \texttt{0})$

Theorem: j>0-when-x>a&b>0

$(\text{ilessp}\,(\texttt{0},\ b) \land \text{ilessp}\,(a,\ x)) \rightarrow \text{ilessp}\,(\texttt{0},\ \text{j}\,(a,\ b,\ x))$

Theorem: x<=a+jb-when-0<b

$\text{ilessp}\,(\texttt{0},\ b) \rightarrow (\neg\ \text{ilessp}\,(\text{iplus}\,(a,\ \text{itimes}\,(b,\ \text{j}\,(a,\ b,\ x))),\ x))$

Theorem: a+jb<x+b-when-0<b&a<x

$(\text{ilessp}\,(\texttt{0},\ b) \land \text{ilessp}\,(a,\ x))$

$\rightarrow \quad \text{ilessp}\,(\text{iplus}\,(a,\ \text{itimes}\,(b,\ \text{j}\,(a,\ b,\ x))),\ \text{iplus}\,(b,\ x))$

Theorem: x<z-when-x<y&y<=z

$(\text{ilessp}\,(x,\ y) \land (\neg\ \text{ilessp}\,(z,\ y))) \rightarrow \text{ilessp}\,(x,\ z)$

Theorem: j<c-when-0<b&a<x&c>1&a+_c-1_b>=x

$(\text{ilessp}\,(\texttt{0},\ b)$

$\ \land \quad \text{ilessp}\,(\texttt{1},\ c)$

$\ \land \quad \text{ilessp}\,(a,\ x)$

$\ \land \quad (\neg\ \text{ilessp}\,(\text{iplus}\,(a,\ \text{itimes}\,(b,\ \text{iplus}\,(\texttt{-1},\ c))),\ x)))$

$\rightarrow \quad \text{ilessp}\,(\text{j}\,(a,\ b,\ x),\ c)$

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Lemma 23. A special case of Lemma 22.
;  Assume  0 < b, 1 < c, 0 < d, a < x, a+(c-1)b >= x,
;  and IterK( a,b,c,d,c,x ) exists.

;  Then IterK( a,b,c,d,1,x-J( a,b,x )b ) exists

;   and

;  cost[ IterK( a,b,c,d,1,x-J( a,b,x )b ) ]
;   <
;  cost[ IterK( a,b,c,d,c,x ) ]

; Proof. By Lemma 22 and the definition of J.
```

THEOREM: iterk_1_x-jb-exists-when-iterk_e=c_x-exists&0<b&1<c&0<d&etc

```
 (ilessp (0, b)
  ∧    ilessp (1, c)
  ∧    ilessp (0, d)
  ∧    ilessp (a, x)
  ∧    (¬ ilessp (iplus (a, itimes (b, iplus (-1, c))), x))
  ∧    v&c-apply$ ('iterk,
                    list (cons (a, 0),
                          cons (b, 0),
                          cons (c, 0),
                          cons (d, 0),
                          cons (c, 0),
                          cons (x, 0))))
 →    v&c-apply$ ('iterk,
                    list (cons (a, 0),
                          cons (b, 0),
                          cons (c, 0),
                          cons (d, 0),
                          '(1 . 0),
                          cons (idifference (x, itimes (b, j (a, b, x))), 0)))
```

THEOREM: iterk_1_x-jb_cost<cost-iterk_e=c_x

```
 (ilessp (0, b)
  ∧    ilessp (1, c)
  ∧    ilessp (0, d)
  ∧    ilessp (a, x)
  ∧    (¬ ilessp (iplus (a, itimes (b, iplus (-1, c))), x))
  ∧    v&c-apply$ ('iterk,
                    list (cons (a, 0),
                          cons (b, 0),
                      cons (c, 0),
                      cons (d, 0),
                      cons (c, 0),
                      cons (x, 0))))
→   (cdr (v&c-apply$ ('iterk,
                      list (cons (a, 0),
                            cons (b, 0),
                            cons (c, 0),
                            cons (d, 0),
                            '(1 . 0),
                            cons (idifference (x, itimes (b, j (a, b, x))), 0))))
       <   cdr (v&c-apply$ ('iterk,
                            list (cons (a, 0),
                                  cons (b, 0),
                                  cons (c, 0),
                                  cons (d, 0),
                                  cons (c, 0),
                                  cons (x, 0)))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Define the function L( a,b,d,x ) to be the following
;  combination of  x, b, d, and the functions J and N:

;     x + N( a,d,x ) d - J( a,b,x+N( a,d,x )d ) b.
```

Definition:

```
l (a, b, d, x)
=   iplus (iplus (x, itimes (d, n (a, d, x))),
           ineg (itimes (b, j (a, b, iplus (x, itimes (d, n (a, d, x)))))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Lemma 24. Computing K with L( a,b,d,x ) is cheaper
;           than computing K with x.

;  Assume  0 < b, 1 < c, 0 < d, a >= x, (c-1)b >= d,
;  and IterK( a,b,c,d,1,x ) exists.

;  Then IterK( a,b,c,d,1,L( a,b,d,x ) ) exists

;   and

;  cost[ IterK( a,b,c,d,1,L( a,b,d,x ) ) ]
;   <
;  cost[ IterK( a,b,c,d,1,x ) ]
; Proof. By Lemma 19 and Lemma 23.
```

THEOREM: x>=z-when-x>=y&y>=z

```
((¬ ilessp (x, y)) ∧ (¬ ilessp (y, z))) → (¬ ilessp (x, z))
```

THEOREM: a+_c-1_b>=x+nd-when-0<d&a>=x&_c-1_b>=d

```
(ilessp (0, d)
 ∧   (¬ ilessp (a, x))
 ∧   (¬ ilessp (itimes (b, iplus (-1, c)), d)))
 →   (¬ ilessp (iplus (a, itimes (b, iplus (-1, c))),
                iplus (x, itimes (d, n (a, d, x)))))
```

THEOREM: iterk-l-exists-when-iterk-x-exists&0<b&1<c&0<d&etc

```
(ilessp (0, b)
 ∧   ilessp (1, c)
 ∧   ilessp (0, d)
 ∧   (¬ ilessp (a, x))
 ∧   (¬ ilessp (itimes (b, iplus (-1, c)), d))
 ∧   v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       '(1 . 0),
                       cons (x, 0))))
 →   v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                       cons (d, 0),
                       '(1 . 0),
                       cons (l (a, b, d, x), 0)))
```

THEOREM: iterk-l-cost<cost-iterk-x&0<b&1<c&0<d-etc

```
(ilessp (0, b)
 ∧   ilessp (1, c)
 ∧   ilessp (0, d)
 ∧   (¬ ilessp (a, x))
 ∧   (¬ ilessp (itimes (b, iplus (-1, c)), d))
 ∧   v&c-apply$ ('iterk,
                 list (cons (a, 0),
                       cons (b, 0),
                       cons (c, 0),
                          cons (d, 0),
                          '(1 . 0),
                          cons (x, 0))))
→    (cdr (v&c-apply$ ('iterk,
                          list (cons (a, 0),
                                cons (b, 0),
                                cons (c, 0),
                                cons (d, 0),
                                '(1 . 0),
                                cons (l (a, b, d, x), 0))))
        <    cdr (v&c-apply$ ('iterk,
                                list (cons (a, 0),
                                      cons (b, 0),
                                      cons (c, 0),
                                      cons (d, 0),
                                      '(1 . 0),
                                      cons (x, 0)))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Define the function IterL( i,a,b,d,x ) which iterates
;  the function L i times.
```

THEOREM: count_x-1<count_x-when-1<x
ilessp (1, x) → (count (iplus (-1, x)) < count (x))

DEFINITION:
iterl (i, a, b, d, x)
= **if** ilessp (1, i) **then** l (a, b, d, iterl (iplus (-1, i), a, b, d, x))
else l (a, b, d, x) **endif**

THEOREM: l<=a
ilessp (0, b) → (¬ ilessp (a, l (a, b, d, x)))

EVENT: Disable l.

THEOREM: iterl<=a
ilessp (0, b) → (¬ ilessp (a, iterl (i, a, b, d, x)))

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Lemma 25. The cost of computing K is unbounded
;           when (c-1)b >= d

;  Assume  0 < b, 1 < c, 0 < d, a >= x, (c-1)b >= d,
;  i > 0, and IterK( a,b,c,d,1,x ) exists.
;  Then IterK( a,b,c,d,1,IterL( i,a,b,d,x ) ) exists

;   and

;  cost[ IterK( a,b,c,d,1,IterL( i,a,b,d,x ) ) ] + i-1
;   <
;  cost[ IterK( a,b,c,d,1,x ) ]

; Proof. By induction on i using Lemma 24.
```

THEOREM: iterk-iterl-exists-when-iterk-x-exists&0<b&1<c&0<d&etc
(ilessp (0, b)
 ∧ ilessp (1, c)
 ∧ ilessp (0, d)
 ∧ ilessp (0, i)
 ∧ (¬ ilessp (a, x))
 ∧ (¬ ilessp (itimes (b, iplus (-1, c)), d))
 ∧ v&c-apply$ ('iterk,
  list (cons (a, 0),
   cons (b, 0),
   cons (c, 0),
   cons (d, 0),
   '(1 . 0),
   cons (x, 0))))
→ v&c-apply$ ('iterk,
  list (cons (a, 0),
   cons (b, 0),
   cons (c, 0),
   cons (d, 0),
   '(1 . 0),
   cons (iterl (i, a, b, d, x), 0)))

THEOREM: fix-int-cost=cost
fix-int (cdr (v&c-apply$ (fn, $args$))) = cdr (v&c-apply$ (fn, $args$))

THEOREM: x+i-1<z-when-x<y&-1-1+i+y<z
(($x < y$) ∧ (iplus (-1, iplus (-1, iplus (i, y))) $<$ z))
→ (iplus (-1, iplus (i, x)) $<$ z)

THEOREM: iterk-iterl-cost+i-1<cost-iterk-x&0<b&1<c&0<d-etc-induction-step
(ilessp (0, i)
 ∧ ($i \neq 1$)
 ∧ (iplus (-1,

```
              iplus (-1,
                     iplus (i,
                            cdr (v&c-apply$ ('iterk,
                                                 list (cons (a, 0),
                                                      cons (b, 0),
                                                      cons (c, 0),
                                                      cons (d, 0),
                                                      '(1 . 0),
                                                      cons (iterl (iplus (-1, i),
                                                                   a,
                                                                   b,
                                                                   d,
                                                                   x),
                                                            0)))))))
       <    cdr (v&c-apply$ ('iterk,
                                 list (cons (a, 0),
                                      cons (b, 0),
                                      cons (c, 0),
                                      cons (d, 0),
                                      '(1 . 0),
                                      cons (x, 0)))))
 ∧    ilessp (0, b)
 ∧    ilessp (1, c)
 ∧    ilessp (0, d)
 ∧    (¬ ilessp (a, x))
 ∧    (¬ ilessp (itimes (b, c), iplus (b, d)))
 ∧    v&c-apply$ ('iterk,
                     list (cons (a, 0),
                          cons (b, 0),
                          cons (c, 0),
                          cons (d, 0),
                          '(1 . 0),
                          cons (x, 0))))
→     (iplus (-1,
              iplus (i,
                     cdr (v&c-apply$ ('iterk,
                                          list (cons (a, 0),
                                               cons (b, 0),
                                               cons (c, 0),
                                               cons (d, 0),
                                               '(1 . 0),
                                               cons (l (a,
                                                        b,
                                                        d,
                                                  iterl (iplus (-1, i), a, b, d, x)),
                                                0))))))
     <    cdr (v&c-apply$ ('iterk,
                           list (cons (a, 0),
                                 cons (b, 0),
                                 cons (c, 0),
                                 cons (d, 0),
                                 '(1 . 0),
                                 cons (x, 0)))))
```

Theorem: iterk-iterl-cost+i-1<cost-iterk-x&0<b&1<c&0<d-etc

```
 (ilessp (0, b)
  ∧   ilessp (1, c)
  ∧   ilessp (0, d)
  ∧   ilessp (0, i)
  ∧   (¬ ilessp (a, x))
  ∧   (¬ ilessp (itimes (b, iplus (-1, c)), d))
  ∧   v&c-apply$ ('iterk,
                  list (cons (a, 0),
                        cons (b, 0),
                        cons (c, 0),
                        cons (d, 0),
                        '(1 . 0),
                        cons (x, 0))))
 →    (iplus (-1,
             iplus (i,
                    cdr (v&c-apply$ ('iterk,
                                     list (cons (a, 0),
                                           cons (b, 0),
                                           cons (c, 0),
                                           cons (d, 0),
                                           '(1 . 0),
                                           cons (iterl (i, a, b, d, x), 0))))))
     <    cdr (v&c-apply$ ('iterk,
                           list (cons (a, 0),
                                 cons (b, 0),
                                 cons (c, 0),
                                 cons (d, 0),
                                 '(1 . 0),
                                 cons (x, 0)))))
```

Theorem: i<y-when-0<x&-1+i+x<y

$((0 < x) \wedge (\text{iplus}\,(-1,\ \text{iplus}\,(i,\ x)) < y)) \rightarrow (i < y)$

Event: Disable iterl.

THEOREM: iterk-cost-is-unbounded-when-_c-1_b>=d&x<=a&etc

```
 (ilessp (0, b)
  ∧   ilessp (1, c)
  ∧   ilessp (0, d)
  ∧   ilessp (0, i)
  ∧   (¬ ilessp (a, x))
  ∧   (¬ ilessp (itimes (b, iplus (-1, c)), d))
  ∧   v&c-apply$ ('iterk,
                  list (cons (a, 0),
                        cons (b, 0),
                        cons (c, 0),
                        cons (d, 0),
                        '(1 . 0),
                        cons (x, 0))))
 →    (i < cdr (v&c-apply$ ('iterk,
                            list (cons (a, 0),
                                  cons (b, 0),
                                  cons (c, 0),
                                  cons (d, 0),
                                  '(1 . 0),
                                  cons (x, 0)))))
```

THEOREM: iterk-does-not-exist-when-_c-1_b>=d&x<=a&etc

```
 (ilessp (0, b)
  ∧   ilessp (1, c)
  ∧   ilessp (0, d)
  ∧   (¬ ilessp (a, x))
  ∧   (¬ ilessp (itimes (b, iplus (-1, c)), d)))
 →    (¬ v&c-apply$ ('iterk,
                     list (cons (a, 0),
                           cons (b, 0),
                           cons (c, 0),
                           cons (d, 0),
                           '(1 . 0),
                           cons (x, 0))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; The next two events are versions of Part 9 of the
; Main Theorem given above in the introduction.

;  9. If  x <= a, d > 0, c > 1, b > 0, and
;     (c-1)b >= d, then K( a,b,c,d,x ) does
;     not exist.
;     Proof. By Lemma 25 and the definition of K.
```

THEOREM: k-does-not-exist-when-_c-1_b>=d&x<=a&etc

```
 (ilessp (0, b)
  ∧   ilessp (1, c)
  ∧   ilessp (0, d)
  ∧   (¬ ilessp (a, x))
  ∧   (¬ ilessp (itimes (b, iplus (-1, c)), d)))
 →    (¬ v&c-apply$ ('k,
                     list (cons (a, 0),
                           cons (b, 0),
                           cons (c, 0),
                           cons (d, 0),
                           cons (x, 0))))
```

THEOREM: k-does-not-halt-when-_c-1_b>=d&x<=a&etc

```
 (vc-a
  ∧   vc-b
  ∧   vc-c
  ∧   vc-d
  ∧   vc-x
  ∧   ilessp (0, car (vc-b))
  ∧   ilessp (1, car (vc-c))
  ∧   ilessp (0, car (vc-d))
  ∧   (¬ ilessp (car (vc-a), car (vc-x)))
  ∧   (¬ ilessp (itimes (car (vc-b), iplus (-1, car (vc-c))), car (vc-d))))
 →    (¬ v&c-apply$ ('k, list (vc-a, vc-b, vc-c, vc-d, vc-x)))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Here a quantifier is used to define the concept that K
; is a total function of x with parameters a,b,c, and d.

; This is the only place in this file of events that
; an explicit quantifier is used.
```

DEFINITION:

```
k-is-total (vc-a, vc-b, vc-c, vc-d)
↔   ∀ vc-x (vc-x → v&c-apply$ ('k, list (vc-a, vc-b, vc-c, vc-d, vc-x)))
```

THEOREM: k-is-total-suff

```
 (vc-x (vc-a, vc-b, vc-c, vc-d)
  →   v&c-apply$ ('k, list (vc-a, vc-b, vc-c, vc-d, vc-x (vc-a, vc-b, vc-c, vc-d))))
 →    k-is-total (vc-a, vc-b, vc-c, vc-d)
```

THEOREM: k-is-total-necc

$$
\begin{array}{l}
(\neg\ (\mathit{vc\text{-}x} \rightarrow \text{v\&c-apply\$}\,(\texttt{'k},\ \text{list}\,(\mathit{vc\text{-}a},\ \mathit{vc\text{-}b},\ \mathit{vc\text{-}c},\ \mathit{vc\text{-}d},\ \mathit{vc\text{-}x}))))\\
\rightarrow \quad (\neg\ \text{k-is-total}\,(\mathit{vc\text{-}a},\ \mathit{vc\text{-}b},\ \mathit{vc\text{-}c},\ \mathit{vc\text{-}d}))
\end{array}
$$

EVENT: Disable k.

EVENT: Disable k-is-total.

THEOREM: knuth-theorem-if-part-case-1

$$
\begin{array}{ll}
(\mathit{vc\text{-}a} & \\
\ \wedge & \mathit{vc\text{-}b}\\
\ \wedge & \mathit{vc\text{-}c}\\
\ \wedge & \mathit{vc\text{-}d}\\
\ \wedge & \text{ilessp}\,(\texttt{0},\ \text{car}\,(\mathit{vc\text{-}b}))\\
\ \wedge & \text{ilessp}\,(\texttt{1},\ \text{car}\,(\mathit{vc\text{-}c}))\\
\ \wedge & \text{ilessp}\,(\texttt{0},\ \text{car}\,(\mathit{vc\text{-}d})))\\
\rightarrow & (\text{ilessp}\,(\text{itimes}\,(\text{car}\,(\mathit{vc\text{-}b}),\ \text{iplus}\,(\texttt{-1},\ \text{car}\,(\mathit{vc\text{-}c}))),\ \text{car}\,(\mathit{vc\text{-}d}))\\
& \quad \rightarrow \quad \text{k-is-total}\,(\mathit{vc\text{-}a},\ \mathit{vc\text{-}b},\ \mathit{vc\text{-}c},\ \mathit{vc\text{-}d}))
\end{array}
$$

THEOREM: knuth-theorem-if-part-case-2

$$
\begin{array}{ll}
(\mathit{vc\text{-}a} & \\
\ \wedge & \mathit{vc\text{-}b}\\
\ \wedge & \mathit{vc\text{-}c}\\
\ \wedge & \mathit{vc\text{-}d}\\
\ \wedge & \text{ilessp}\,(\texttt{0},\ \text{car}\,(\mathit{vc\text{-}b}))\\
\ \wedge & (\neg\ \text{ilessp}\,(\texttt{1},\ \text{car}\,(\mathit{vc\text{-}c})))\\
\ \wedge & \text{ilessp}\,(\texttt{0},\ \text{car}\,(\mathit{vc\text{-}d})))\\
\rightarrow & (\text{ilessp}\,(\text{itimes}\,(\text{car}\,(\mathit{vc\text{-}b}),\ \text{iplus}\,(\texttt{-1},\ \text{car}\,(\mathit{vc\text{-}c}))),\ \text{car}\,(\mathit{vc\text{-}d}))\\
& \quad \rightarrow \quad \text{k-is-total}\,(\mathit{vc\text{-}a},\ \mathit{vc\text{-}b},\ \mathit{vc\text{-}c},\ \mathit{vc\text{-}d}))
\end{array}
$$

THEOREM: knuth-theorem-if-part

$$
\begin{array}{ll}
(\mathit{vc\text{-}a} & \\
\ \wedge & \mathit{vc\text{-}b}\\
\ \wedge & \mathit{vc\text{-}c}\\
\ \wedge & \mathit{vc\text{-}d}\\
\ \wedge & \text{ilessp}\,(\texttt{0},\ \text{car}\,(\mathit{vc\text{-}b}))\\
\ \wedge & \text{ilessp}\,(\texttt{0},\ \text{car}\,(\mathit{vc\text{-}d})))\\
\rightarrow & (\text{ilessp}\,(\text{itimes}\,(\text{car}\,(\mathit{vc\text{-}b}),\ \text{iplus}\,(\texttt{-1},\ \text{car}\,(\mathit{vc\text{-}c}))),\ \text{car}\,(\mathit{vc\text{-}d}))\\
& \quad \rightarrow \quad \text{k-is-total}\,(\mathit{vc\text{-}a},\ \mathit{vc\text{-}b},\ \mathit{vc\text{-}c},\ \mathit{vc\text{-}d}))
\end{array}
$$

THEOREM: knuth-theorem-only-if-part-case-1

$$
\begin{array}{ll}
(\mathit{vc\text{-}a} & \\
\ \wedge & \mathit{vc\text{-}b}\\
\ \wedge & \mathit{vc\text{-}c}
\end{array}
$$

```
 ∧   vc-d
 ∧   ilessp (0, car (vc-b))
 ∧   ilessp (1, car (vc-c))
 ∧   ilessp (0, car (vc-d)))
→    (k-is-total (vc-a, vc-b, vc-c, vc-d)
       →   ilessp (itimes (car (vc-b), iplus (-1, car (vc-c))), car (vc-d)))
```

THEOREM: knuth-theorem-only-if-part-case-2

```
(ilessp (0, b) ∧ (¬ ilessp (1, c)) ∧ ilessp (0, d))
→   ilessp (itimes (b, iplus (-1, c)), d)
```

THEOREM: knuth-theorem-only-if-part

```
(vc-a
 ∧   vc-b
 ∧   vc-c
 ∧   vc-d
 ∧   ilessp (0, car (vc-b))
 ∧   ilessp (0, car (vc-d)))
→    (k-is-total (vc-a, vc-b, vc-c, vc-d)
       →   ilessp (itimes (car (vc-b), iplus (-1, car (vc-c))), car (vc-d)))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Here is a version of Knuth's theorem:

; Theorem.  The generalized 91 recursion with parameters ( a,b,c,d )
;           defines a total function on the integers if and only if
;           (c-1)b < d.
```

THEOREM: knuth-theorem

```
(vc-a
 ∧   vc-b
 ∧   vc-c
 ∧   vc-d
 ∧   ilessp (0, car (vc-b))
 ∧   ilessp (0, car (vc-d)))
→    (k-is-total (vc-a, vc-b, vc-c, vc-d)
       ↔   ilessp (itimes (car (vc-b), iplus (-1, car (vc-c))), car (vc-d)))
```

THEOREM: k-value-when-a<x

```
ilessp (a, x)
→   (car (v&c-apply$ ('k,
                      list (cons (a, 0),
                            cons (b, 0),
                            cons (c, 0),
                            cons (d, 0),
                            cons (x, 0))))
           =   idifference (x, b))
```

Theorem: k-value-when-a<x-version-2

```
 (vc-a ∧ vc-b ∧ vc-c ∧ vc-d ∧ vc-x ∧ ilessp (car (vc-a), car (vc-x)))
 →    (car (v&c-apply$ ('k, list (vc-a, vc-b, vc-c, vc-d, vc-x)))
          =   idifference (car (vc-x), car (vc-b)))
```

Theorem: knuth-theorem-part-2-case-1

```
 (vc-a
  ∧   vc-b
  ∧   vc-c
  ∧   vc-d
  ∧   vc-x
  ∧   ilessp (0, car (vc-b))
  ∧   ilessp (1, car (vc-c))
  ∧   ilessp (0, car (vc-d)))
 →    (car (v&c-apply$ ('k, list (vc-a, vc-b, vc-c, vc-d, vc-x)))
          =   if ilessp (car (vc-a), car (vc-x))
              then idifference (car (vc-x), car (vc-b))
              else car (v&c-apply$ ('k,
                                    list (vc-a,
                                          vc-b,
                                          vc-c,
                                          vc-d,
                                          cons (iplus (car (vc-x),
                                                       idifference (car (vc-d),
                                                                    itimes (car (vc-b),
                                                                            iplus (-1,
                                                                                   car (vc-c))))),
                                                cost)))) endif)
```

Theorem: k-value=body-value-when-a>=x&c=1

```
 (¬ ilessp (a, x))
 →    (car (v&c-apply$ ('k,
                        list (cons (a, 0),
                              cons (b, 0),
                              '(1 . 0),
                              cons (d, 0),
                              cons (x, 0))))
          =   car (v&c-apply$ ('k,
                               list (cons (a, 0),
                                     cons (b, 0),
                                     '(1 . 0),
                                     cons (d, 0),
                                     cons (iplus (x, d), 0)))))
```

THEOREM: k-value=body-value-when-a>=x&c=1-version-2

```
(vc-a
 ∧   vc-b
 ∧   vc-c
 ∧   vc-d
 ∧   vc-x
 ∧   (car (vc-c) = 1)
 ∧   (¬ ilessp (car (vc-a), car (vc-x))))
→    (car (v&c-apply$ ('k, list (vc-a, vc-b, vc-c, vc-d, vc-x)))
        =   car (v&c-apply$ ('k,
                              list (vc-a,
                                    vc-b,
                                    vc-c,
                                    vc-d,
                                    cons (iplus (car (vc-x), car (vc-d)), cost)))))
```

THEOREM: knuth-theorem-part-2-case-2

```
(vc-a ∧ vc-b ∧ vc-c ∧ vc-d ∧ vc-x ∧ (car (vc-c) = 1))
→    (car (v&c-apply$ ('k, list (vc-a, vc-b, vc-c, vc-d, vc-x)))
        =   if ilessp (car (vc-a), car (vc-x))
            then idifference (car (vc-x), car (vc-b))
            else car (v&c-apply$ ('k,
                                   list (vc-a,
                                         vc-b,
                                         vc-c,
                                         vc-d,
                                         cons (iplus (car (vc-x),
                                                      idifference (car (vc-d),
                                                                   itimes (car (vc-b),
                                                                           iplus (-1,
                                                                                  car (vc-c))))),
                                               cost)))) endif)
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Here is a version of that part of Knuth's theorem
; which gives the circumstances when K satisfies a
; simpler recurrence.

; Theorem.  In such a case the values of K( x ) also
;           satisfy the much simpler recurrence

;           K( x )  =  if  x > a  then  x - b
;                                             else  K( x+d-(c-1)b ).
```

THEOREM: knuth-theorem-part-2

```
(vc-a
  ∧   vc-b
  ∧   vc-c
  ∧   vc-d
  ∧   vc-x
  ∧   ilessp (0, car (vc-b))
  ∧   ilessp (0, car (vc-c))
  ∧   ilessp (0, car (vc-d)))
 →    (car (v&c-apply$ ('k, list (vc-a, vc-b, vc-c, vc-d, vc-x)))
       =    if ilessp (car (vc-a), car (vc-x))
            then idifference (car (vc-x), car (vc-b))
            else car (v&c-apply$ ('k,
                                     list (vc-a,
                                          vc-b,
                                          vc-c,
                                          vc-d,
                                          cons (iplus (car (vc-x),
                                                       idifference (car (vc-d),
                                                                    itimes (car (vc-b),
                                                                            iplus (-1,
                                                                                   car (vc-c))))),
                                                 cost)))) endif)
```

Index