Raw Code, Specification, and Proof of the Avalon Queue Example

Chun Gong and Jeannette M. Wing

11 August 1989

CMU-CS-89-172

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

This technical report contains the unedited code, specification, and proofs of properties of the Avalon/C++ queue example. The code compiles and runs. We used the Larch Checker to process the specifications and then used these specifications as input to the Larch Prover. We then proved the representation invariants and key correctness condition for the queue example, proving various sets of helping lemmas in the process. The companion technical report [5] gives a high-level description of this specification and verification exercise, including a performance analysis.

This research was sponsored by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 4976, monitored by the Air Force Avionics Laboratory Under Contract No. F33615-87-C-1499. Additional support was provided in part by the National Science Foundation under grant CCR-8620027.

Contents

Raw Code, Specification, and Proof of the Avalon Queue Example

Chun Gong and Jeannette M. Wing

August 10, 1989

1. Introduction

This document contains the code, specification, and proof of the well-worn Avalon/C++ queue example [1, 2]. The code compiles and runs. We successfully ran the Larch Shared Language (LSL) [3] specifications through the Larch Checker (LC) which checked for syntactic, type, and static semantic errors. Some minor edits were made to these specifications (e.g., adding some signature information to disambiguate some operators) to make them acceptable input to the Larch Prover (LP) [4]. We give the LP input versions of the specifications here too. Finally, we proved the representation invariants and the type-specific correctness condition (the so-called "prefix" property [2]) from these specifications using LP. The companion paper [5] gives a high-level description of this specification and verification exercise, including detailed statistics on the time and space usage of LP. Hence, what follows is unedited text that represents the raw code, specification, and proof transcripts.

2. The Avalon Code

2.1. The Representation

```
struct enq_rec {
  int item;                          // Item enqueued.
  trans_id enqr;                     // Who enqueued it.
  enq_rec(int i, trans_id& en)       // Constructor.
    {item = i; enqr = en;}
};

struct deq_rec {
  int item;                          // Item dequeued.
  trans_id enqr;                     // Who enqueued it.
  trans_id deqr;                     // Who dequeued it.
  deq_rec(int i, trans_id& en, trans_id& de);  // Constructor.
    {item = i; enqr = en; deqr = de; }
};

class atomic_queue : public subatomic {
  deq_stack deqd;                    // Stack of dequeued items.
  enq_heap enqd;                     // Heap of enqueued items.
 public:
   atomic_queue() {};                // Create empty queue.
   void enq(int item);               // Enqueue an item.
   int deq();                        // Dequeue an item.
   void commit(trans_id& t);         // Called on commit.
   void abort(trans_id& t);          // Called on abort.
};
```

2.2. The Operations

```
void atomic_queue::enq(int item) {
  trans_id tid = trans_id();
  when (deqd.is_empty() || (deqd.top()->enqr < tid))
    enqd.insert(item, tid);              // Record enqueue.
}

int atomic_queue::deq() {
  trans_id tid = trans_id();
  when ( (deqd.is_empty() || deqd.top()->deqr < tid)
        && enqd.min_exists() && (enqd.get_min()->enqr < tid)) {
    enq_rec* min_er = enqd.delete_min();
    deq_rec dr(*min_er, tid);  // Move from enqueued heap...
    deqd.push(dr);             // to dequeued stack.
    return min_er->item;
  }
}

void atomic_queue::commit(trans_id& committer) {
  when (TRUE)                    // Always ok to commit.
    if (!deqd.is_empty() && descendant(deqd.top()->deqr, committer)) {
      deqd.clear();              // Discard all dequeue records.
    }
}

void atomic_queue::abort(trans_id& aborter) {
  when (TRUE) {                  // Always ok to abort.
    while (!deqd.is_empty()      // Undo aborted dequeue by...
      && descendant(deqd.top()->deqr, aborter)) { // aborting transaction.
      deq_rec* d = deqd.pop(); // Undo aborted dequeue.
      enqd.insert(d->item, d->enqr);  // Put it back.
    }
    enqd.discard(aborter);       // Undo aborted enqueues.
```

```
    }
  }
```

3. Larch Specifications

3.1. Some Basics

```
Set (EL, C): trait
  introduces
    emptyset: -> C
    insert: C, EL -> C
    in: EL, C-> Bool
    notin: EL, C -> Bool
    U: C, C -> C
    insect: C, C -> C
    __-__: C, C -> C
    delete: C, EL -> C
    subseteq: C, C -> Bool
    isEmpty: C -> Bool
  asserts C generated by (emptyset, insert)
          C partitioned by (in)
    for all (y, y1: C, x, x1: EL)
      ~(in(x, emptyset)),
      in(x, insert(y, x1)) == (x = x1) | in(x, y),
      notin(x, y) == ~(in(x, y)),
      in(x, U(y, y1)) == in(x, y) | in(x, y1),
      in(x, insect(y, y1)) == in(x, y) & in(x, y1),
      in(x, (y - y1)) == in(x, y) & notin(x, y1),
      in(x, delete(y, x1)) == (x \= x1) & in(x, y),
      subseteq(emptyset, y1),
      subseteq(insert(y, x), y1) == subseteq(y, y1) & in(x, y1),
      isEmpty(emptyset),
      ~isEmpty(insert(y, x))
  end

Stack (EL, C): trait
  introduces
    new: -> C
    push: C, EL -> C
    top: C -> EL
    pop: C -> C
    isNew: C -> Bool
  asserts
    C generated by (new, push)
    for all (x: C, y: EL)
      top(push(x, y)) == y,
      pop(push(x, y)) == x,
      isNew(new),
      ~ isNew(push(x, y))
  end

Pair (T1, T2, T): trait
  introduces
    pair: T1, T2 -> T
    first: T -> T1
    second: T -> T2
  asserts
    T generated by (pair)
    T partitioned by (first, second)
    for all (x: T1, y: T2)
      first(pair(x,y)) == x,
      second(pair(x,y)) == y
  end

Triple (T1, T2, T3, T): trait
  introduces
    trip: T1, T2, T3 -> T
    first: T -> T1
    second: T -> T2
    third: T -> T3
```

```
  asserts
    T generated by (trip)
    T partitioned by (first, second, third)
    for all (x: T1, y: T2, z: T3)
      first(trip(x,y,z)) == x,
      second(trip(x,y,z)) == y,
      third(trip(x,y,z)) == z
  end
```

3.2. Queue Representation

```
TransID(Tid): trait
  introduces
    __<__: Tid, Tid -> Bool
    c_xt: -> Tid
  asserts for all(xt, xt1, xt2: Tid)
    ((xt < xt1) & (xt1 < xt2)) => (xt < xt2),
    ((xt < xt1) & (xt1 < xt)) => (xt = xt1)
  end

Enq_Rec(EL, enq_rec): trait
  includes TransID, Pair(EL, Tid, enq_rec, element for first, enqt for second)
  introduces
    e_before: enq_rec, enq_rec -> Bool
  asserts enq_rec partitioned by (element)
  for all(x, x1: enq_rec)
    e_before(x, x1) == enqt(x) < enqt(x1)
  end

Deq_Rec(EL, deq_rec): trait
  includes TransID, Enq_Rec,
           Triple(EL, Tid, Tid, deq_rec, what for first,
                  enqr for second, deqr for third)
  introduces
    d_before: deq_rec, deq_rec -> Bool
    convert: deq_rec -> enq_rec
  asserts for all (x, x1: deq_rec)
    d_before(x, x1) == deqr(x) < deqr(x1),
    convert(x) == pair(what(x), enqr(x))
  end

Enq_Heap(enq_heap): trait
  includes Enq_Rec, Set(enq_rec, enq_heap)
  introduces
    in_heap: enq_rec, enq_heap -> Bool
    e_in_heap: EL, enq_heap -> Bool
    least: enq_rec, enq_heap -> Bool
    is_top: enq_rec, enq_heap -> Bool
  asserts for all (xp: enq_heap, y, y1:  enq_rec, xt: Tid, xe: EL)
    in_heap(y, xp) == in(y, xp),
    e_in_heap(xe, emptyset) == false,
    e_in_heap(xe, insert(xp, y)) == (element(y)=xe) | e_in_heap(xe, xp),
    least(y, emptyset) == true,
    least(y, insert(xp, y1)) == (enqt(y)<enqt(y1)) & least(y, xp),
    is_top(y, xp) == in_heap(y,xp) & least(y, xp)
  end

Deq_Stack(deq_stack): trait
  includes Deq_Rec, Stack(deq_rec, deq_stack)
  introduces
    deq_before: deq_rec, deq_rec, deq_stack -> Bool
    in_stack: deq_rec, deq_stack -> Bool
    e_in_stack: EL, deq_stack -> Bool
  asserts for all (xk: deq_stack, y, y1, y2: deq_rec, xt: Tid, xe: EL)
    deq_before(y, y1, new) == false,
    deq_before(y, y1, push(xk, y2)) == ((y1=y2) & (in_stack(y, xk))) |
```

```
                                            deq_before(y, y1, xk),
  in_stack(y, new) == false,
  in_stack(y, push(xk, y1)) == if y = y1
                               then true
                               else in_stack(y, xk),
  e_in_stack(xe, new) == false,
  e_in_stack(xe, push(xk, y)) == (what(y)=xe) | e_in_stack(xe, xk)
end
```

3.3. LP Input of Basics and Queue Representation

```
% Last modified on Fri May 19 11:37:29 PDT 1989 by horning
%      modified on Mon Jun 27 15:10:41 1988 by saxe

set name bool
declare
  true:->bool
  false:->bool
  &:bool,bool->bool
  |:bool,bool->bool
  <=>:bool,bool->bool
  =>:bool,bool->bool
  not:bool->bool
  b::bool
  b1::bool
  b2::bool
..

op ac <=> & |
op prec <=> &
op prec <=> |

add
  true & b -> b
  false & b -> false
  b & b -> b
  not(b) -> false <=> b
  true <=> b -> b
  not(b) & b -> false
  true | b -> true
  false | b -> b
  b | b -> b
  not(b) | b -> true
  b => b1 -> not(b) | b1
  (b | b1) & b -> b
  % not(b) & not(b1) -> not(b | b1)
  not(b | b1) -> not(b) & not(b1)
  % not(b) | not(b1) -> not(b & b1)
  not(b & b1) -> not(b) | not(b1)
  b & (not(b) | b1) -> b & b1
  (b | b1) & not(b) & not(b1) -> false
  (b | b1) & (b | not(b1)) -> b
  (b & b1) | not(b1) -> b | not(b1)
  (b & b1) | (b & not(b1)) -> b
  b | (not(b) & b1) -> b | b1
  b | (b & b1) -> b
% Jorgen's additions
%   b | (b1 & b2) -> (b | b1) & (b | b2)
  (b <=> b1) | (b1 <=> b2) | (b <=> b2) -> true
..

add-ded
  when (b <=> false) == false
  yield b -> true
  when b <=> b1 == b <=> b2
  yield b1 == b2
  when if(b, b1, b2) == true
  yield b => b1
        b | b2
  when if(b, b1, b2) == false
  yield b1 => not(b)
        b2 => b

..

set name TransID
declare
```

```
xt,xt1,xt2::Tid
..
add
   ((xt < xt1) & (xt1 < xt2)) => (xt < xt2)
   ((xt < xt1) & (xt1 < xt)) => (xt = xt1)
..

set name Pair
declare
xn,xn1::enq_rec
xe::EL
..
add-generators
  pair : EL, Tid -> enq_rec
..
add-deduction-rules
  when
    element(xn) == element(xn1)
    enqt(xn) == enqt(xn1)
  yield xn == xn1
..
add
  element(pair(xe, xt)) == xe
  enqt(pair(xe, xt)) == xt
..

set name Enq_Rec
add-deduction-rules
  when
    element(xn) == element(xn1)
  yield xn == xn1
..
add
  e_before(x, x1) == (enqt(x) < enqt(x1))
..

set name Triple
add-generators
  trip : EL, Tid, Tid -> deq_rec
..
add-deduction-rules
  when
    what(y) == what(z)
    enqr(y) == enqr(z)
    deqr(y) == deqr(z)
  yield y == z
..
add
  what(trip(x, y, z)) == x
  enqr(trip(x, y, z)) == y
  deqr(trip(x, y, z)) == z
..

set name Deq_Rec
add
  d_before(x, x1) == (deqr(x) < deqr(x1))
  convert(x) == pair(what(x), enqr(x))
..

set name Stack
add-generators
  new : -> deq_stack
  push : deq_stack, deq_rec -> deq_stack
..
add
  top(push(x, y)) == y
  pop(push(x, y)) == x
  isNew(new)
  not(isNew(push(x, y)))
```

```
..

set name Deq_Stack
declare xk::deq_stack
add
  deq_before(y, y1, new) == false
  deq_before(y, y1, push(xk, y2)) == ((y1 = y2) & in_stack(y, xk)) |
                                          deq_before(y, y1, xk)
  in_stack(y, new) == false
  in_stack(y, push(xk, y1)) == if(y = y1, true, in_stack(y, xk))
  e_in_stack(xe, new) == false
  e_in_stack(xe, push(xk, y)) == (what(y) = xe) | e_in_stack(xe, xk)
..

set name Set
add-generators
  emptyset : -> enq_heap
  insert : enq_heap, enq_rec -> enq_heap
..
add-deduction-rules
  when
    in(x_1_1, y) == in(x_1_1, z)
  yield y == z
..
add
  not(in(x, emptyset))
  in(x, insert(y, x1)) == (x = x1) | in(x, y)
  notin(x, y) == not(in(x, y))
  in(x, U(y, y1)) == in(x, y) | in(x, y1)
  in(x, insect(y, y1)) == in(x, y) & in(x, y1)
  in(x, (y - y1)) == in(x, y) & notin(x, y1)
  in(x, delete(y, x1)) == not(x = x1) & in(x, y)
  subseteq(emptyset, y1)
  subseteq(insert(y, x), y1) == subseteq(y, y1) & in(x, y1)
  isEmpty(emptyset)
  not(isEmpty(insert(y, x)))
..

set name Enq_Heap
declare xp::enq_heap
add
  in_heap(y, xp) == in(y, xp)
  e_in_heap(xe, emptyset) == false
  e_in_heap(xe, insert(xp, y)) == (element(y) = xe) | e_in_heap(xe, xp)
  least(y, emptyset) == true
  least(y, insert(xp, y1)) == (enqt(y) < enqt(y1)) & least(y, xp)
  is_top(y, xp) == in_heap(y, xp) & least(y, xp)
..

set name State
declare
xst::St
..
add-generators
  init : -> St
  deq : St, Tid, enq_rec -> St
  enq : St, Tid, EL -> St
  commit : St, Tid -> St
  abort : St, Tid -> St
..

add-deduction-rules
  when
    deqd(y) == deqd(z)
    enqd(y) == enqd(z)
  yield y == z
..
add
  deqd(init) == new
```

```
enqd(init) == emptyset
when_enq(xst, z, w, xt, xe) ==(((deqd(xst)=new)|(enqr(top(deqd(xst)))<xt)) &
                              not(in_heap(z, enqd(xst)) & (element(z) = xe))) &
                              not(in_stack(w, deqd(xst)) & (what(w) = xe))
deqd(enq(xst, xt, xe)) == deqd(xst)
enqd(enq(xst, xt, xe)) == insert(enqd(xst), pair(xe, xt))
when_deq(xst, x,  xt, xn) == ((((deqd(xst)=new)|((deqr(top(deqd(xst)))<xt) &
                                (enqr(top(deqd(xst))) < enqt(xn)))) &
                                is_top(xn, enqd(xst))) & (enqt(xn) < xt)) &
                           not(in_stack(x, deqd(xst)) & (what(x) = element(xn)))
deqd(deq(xst, xt, xn)) == push(deqd(xst), trip(element(xn), enqt(xn), xt))
enqd(deq(xst, xt, xn)) == delete(enqd(xst), xn)
deqd(commit(xst, xt)) ==  if(not(deqd(xst) = new)&(deqr(top(deqd(xst)))<xt),
                                new, deqd(xst))
enqd(commit(xst, xt)) == enqd(xst)
in_stack(x, deqd(abort(xst, xt))) == in_stack(x, deqd(xst))&not(deqr(x) = xt)
deq_before(x, y, deqd(abort(xst, xt))) => deq_before(x, y, deqd(xst))
in_heap(x1, enqd(abort(xst, xt))) => (not(enqt(x1) = xt) &
                                       (in_heap(x1, enqd(xst)) |
                           (in_stack(trip(element(x1),enqt(x1), xt),deqd(xst))&
                             not(in_stack(x,deqd(abort(xst,xt)))&
                                  (what(x)=element(x1))))))
```

3.4. Histories and Abstraction Function

```
Sequence (EL, Seq): trait
  introduces
    null: -> Seq
    cons: Seq, EL -> Seq
    append: Seq, Seq -> Seq
    prefix: Seq, Seq -> Bool
    sub: Seq, Seq -> Seq
  asserts Seq generated by (null, cons)
  for all (xs, xs1: Seq, xe, xe1: EL)
    cons(xs,xe)=cons(xs1,xe1) == (xs=xs1) & (xe=xe1),
    append(xs, null) == xs,
    append(null, xs) == xs,
    append(xs, cons(xs1, xe)) == cons(append(xs, xs1), xe),
    prefix(null, xs1) == true,
    prefix(cons(xs, xe), null) == false,
    prefix(cons(xs, xe), cons(xs1, xe1)) == ((xe=xe1) & (xs=xs1)) | prefix(cons(xs, xe), xs1),
    sub(null, xs) == null,
    sub(xs, null) == xs,
    sub(cons(xs, xe), cons(xs1, xe1)) == if ((xs=xs1) & (xe=xe1))
                                          then null
                                          else cons(sub(xs, cons(xs1,xe1)), xe),
    ~(null = cons(xs,xe))
  end

Event (Ev): trait
  includes Enq_Rec, Deq_Rec,
  introduces
    E: enq_rec -> Ev
    D: deq_rec -> Ev
  asserts Ev generated by (E, D)
          enq_rec partitioned by (E)
          deq_rec partitioned by (D)
  for all (x,x1: enq_rec, y,y1: deq_rec)
    (x=x1)=>(E(x)=E(x1)),
    (y=y1)=>(D(y)=D(y1)),
    ~(E(x)=D(y))
  end

History (H): trait
  includes Event, Sequence, Sequence(Ev, H)
  introduces
    c_h1: -> H
    c_h2: -> H
    DEQ: H -> Seq
    ENQ: H -> Seq
    max: Tid, H -> Bool
    min: Tid, H -> Bool
    ordered: H -> Bool
    discard: Tid, H -> H
  asserts for all (xh: H, u:enq_rec, v:deq_rec, xt:Tid)
    ENQ(null) == null,
    ENQ(cons(xh, E(u))) == cons(ENQ(xh),element(u)),
    ENQ(cons(xh, D(v))) == ENQ(xh),
    DEQ(null) == null,
    DEQ(cons(xh, E(u))) == DEQ(xh),
    DEQ(cons(xh, D(v))) == cons(DEQ(xh), what(v)),
    max(xt,null),
    max(xt,cons(xh,E(u))) == max(xt,xh) & (~(enqt(u)<xt)),
    max(xt,cons(xh,D(v))) == max(xt,xh) & (~(deqr(v)<xt)),
    min(xt,null),
    min(xt,cons(xh,E(u))) == min(xt,xh) & (~(xt<enqt(u))),
    min(xt,cons(xh,D(v))) == min(xt,xh) & (~(xt<deqr(v))),
    ordered(null),
    ordered(cons(xh,E(u))) == ordered(xh) & min(enqt(u), xh),
    ordered(cons(xh,D(v))) == ordered(xh) & min(deqr(v), xh),
    discard(xt, null) == null,
```

```
  discard(xt, cons(xh,E(u))) == if enqt(u)=xt
                                then discard(xt,xh)
                                else cons(discard(xt,xh),E(u)),
  discard(xt, cons(xh,D(v))) == if deqr(v)=xt
                                then discard(xt,xh)
                                else cons(discard(xt,xh),D(v))
end
```

3.5. LP Input of Histories and Abstraction Function

```
set name Event
add-generators
  E : enq_rec -> Ev
  D : deq_rec -> Ev
..
add-deduction-rules
  when E(xu::enq_rec) == E(xv::enq_rec)
  yield xu::enq_rec == xv::enq_rec
..
add-deduction-rules
  when D(yu::deq_rec) == D(yv::deq_rec)
  yield yu::deq_rec == yv::deq_rec
..
add
  (x=x1)=>(E(x)=E(x1))
  (yu::deq_rec=yu1::deq_rec)=>(D(yu::deq_rec)=D(yu1::deq_rec))
  not(E(x)=D(yu::deq_rec))
..
set name Sequence
declare xs,xs1::Seq
declare xe,xe1::EL
add-generators
  null : -> Seq
  cons : Seq, EL -> Seq
..
add
  cons(xs,xe)=cons(xs1,xe1) == (xs=xs1) & (xe=xe1)
  append(xs, null) == xs
  append(null, xs) == xs
  append(xs, cons(xs1, xe)) == cons(append(xs, xs1),  xe)
  prefix(null, xs1) == true
  prefix(cons(xs, xe), null) == false
  prefix(cons(xs, xe), cons(xs1, xe1)) == ((xe = xe1) & (xs = xs1)) |
                                          prefix(cons(xs, xe), xs1)
  sub(null, xs) == null
  sub(xs, null) == xs
  sub(cons(xs, xe), cons(xs1, xe1)) == if((xs = xs1) & (xe = xe1), null,
                                          cons(sub(xs, cons(xs1, xe1)), xe))
  not(null = cons(xs,xe))
..

set name Sequence
declare xh::H
declare xev,xev1::Ev
add-generators
  null : -> H
  cons : H, Ev -> H
..
add
  cons:H,Ev->H(xh,xev)=cons:H,Ev->H(xh1,xev1) == (xh=xh1) & (xev=xev1)
  append(xh, null:->H) == xh
  append(null:->H, xh) == xh
  append(xh, cons(xh1, xev)) == cons(append(xh, xh1), xev)
  prefix(null:->H, xh1) == true
  prefix(cons(xh, xev), null:->H) == false
  prefix(cons(xh, xev), cons(xh1, xev1)) == ((xev = xev1) & (xh = xh1)) |
                                            prefix(cons(xh, xev), xh1)
  sub(null:->H, xh) == null:->H
  sub(xh, null:->H) == xh
  sub(cons(xh, xev), cons(xh1, xev1)) == if((xh = xh1)&(xev = xev1), null:->H,
                                            cons(sub(xh, cons(xh1, xev1)), xev))
  not(null:->H = cons(xh,xev))
..

set name History
declare ue::enq_rec
```

```
declare vd::deq_rec
declare xt::Tid
declare c_h1,c_h2:->H
declare c_ue:->enq_rec
add
  ENQ(null:->H) == null:->Seq
  ENQ(cons(xh, E(ue))) == cons:Seq,EL->Seq(ENQ(xh), element(ue))
  ENQ(cons(xh, D(vd))) == ENQ(xh)
  DEQ(null:->H) == null:->Seq
  DEQ(cons(xh, E(ue))) == DEQ(xh)
  DEQ(cons(xh, D(vd))) == cons:Seq,EL->Seq(DEQ(xh), what(vd))
  max(xt, null:->H)
  max(xt, cons(xh, E(ue))) == max(xt, xh) & not(enqt(ue)<xt)
  max(xt, cons(xh, D(vd))) == max(xt, xh) & not(deqr(vd)<xt)
  min(xt, null:->H)
  min(xt, cons(xh, E(ue))) == min(xt, xh) & not(xt<enqt(ue))
  min(xt, cons(xh, D(vd))) == min(xt, xh) & not(xt<deqr(vd))
  ordered(null:->H)
  ordered(cons(xh,E(ue))) == ordered(xh) & min(enqt(ue), xh)
  ordered(cons(xh,D(vd))) == ordered(xh) & min(deqr(vd), xh)
  discard(xt, null:->H) == null:->H
  discard(xt, cons(xh, E(ue))) == if(enqt(ue) = xt, xh,
                                    cons(xh, E(ue)))
  discard(xt, cons(xh, D(vd))) == if(deqr(vd) = xt, xh,
                                    cons(xh, D(vd)))
..

set name Set
declare ya,ya1,za::A
add-generators
  emptyset : -> A
  insert : A, H -> A
..

add-deduction-rules
  when  in(xh, ya) == in(xh, za)
  yield ya == za
..
add
  not(in(xh, emptyset:->A))
  in(xh, insert(ya, xh1)) == (xh = xh1) | in(xh, ya)
  notin(xh, ya) == not(in(xh, ya))
  in(xh, U(ya, ya1)) == in(xh, ya) | in(xh, ya1)
  in(xh, insect(ya, ya1)) == in(xh, ya) & in(xh, ya1)
  in(xh, (ya - ya1)) == in(xh, ya) & notin(xh, ya1)
  in(xh, delete(ya, xh1)) == not(xh = xh1) & in(xh, ya)
  subseteq(emptyset:->A, ya1)
  subseteq(insert(ya, xh), ya1) == subseteq(ya, ya1) & in(xh, ya1)
  isEmpty(emptyset:->A)
  not(isEmpty(insert(ya, xh)))
..

set name Abstraction
declare xst::St
declare c_xt:->Tid
add
  in_state(null:->H, xst)
  in_state(cons(xh, E(ue)), xst) => (in_state(xh,xst)&(in_heap(ue,enqd(xst)) |
                      in_stack(trip(element(ue),enqt(ue),c_xt),deqd(xst))))
  in_state(cons(xh, D(vd)), xst) => (in_state(xh,xst)&in_stack(vd,deqd(xst)))
  in_state(xh,xst) => not(DEQ(xh) = cons:Seq,EL->Seq(ENQ(xh),xe))
  in(xh, af(xst)) => (ordered(xh) & in_state(xh, xst))
  in(xh, af(enq(xst, xt, xe))) => (in(append(c_h1, c_h2), af(xst)) &
                                  (xh=append(cons(c_h1, E(pair(xe,xt))),c_h2)))
  in(xh, af(deq(xst, xt, xn))) => (in(append(c_h1, c_h2), af(xst)) &
                                  (xh = append(cons(c_h1,
                                  D(trip(element(xn),enqt(xn),xt))),c_h2)) &
                                  (DEQ(c_h2)=null:->Seq))
  in(xh, af(commit(xst, xt))) => (DEQ(xh) = null:->Seq)
```

```
in(xh, af(abort(xst, xt))) => (in(c_h1, af(xst)) & (xh = discard(xt, c_h1)))
(prefix(DEQ(append(xh1,xh2)),ENQ(append(xh1,xh2)))&in(append(xh1,xh2),af(xst))&
 not(prefix(DEQ(append(xh1,xh2)),ENQ(append(cons(xh1,E(pair(xe,xt))),xh2))))&
 ordered(append(cons(xh1,E(pair(xe,xt))),xh2))) =>
                                  not(enqr(top(deqd(xst)))<xt)
(in(xh,af(xst))&prefix(DEQ(xh),ENQ(xh))&in(xn,enqd(xst))&least(xn,enqd(xst)))
              => prefix(cons:Seq,EL->Seq(DEQ(xh),element(xn)),ENQ(xh))
```

4. Proof of Representation Invariants

4.1. Statement of Representation Invariants

```
set name Invariant
add
Inv1(xst, x, y) == (in_stack(x,deqd(xst))&in_heap(y,enqd(xst)))=>
                         not(what(x)=element(y))
Inv2(xst, x, x1) == deq_before(x,x1,deqd(xst))=>
                   (enqr(x)<enqr(x1))&(deqr(x)<deqr(x1))
Inv3(xst, x) == in_stack(x, deqd(xst))=>(enqr(x)<deqr(x))
..
```

4.2. LP Proof Session of Invariant 1

```
-> thaw Inv

System thawed from 'Inv.frz'.

-> set name thml

The name prefix is now 'thml'.

-> prove Invl(xst,x,y) by induction xst St

The basis step in an inductive proof of Conjecture thml.1
    Invl(xst, x, y) -> true
involves proving the following lemma(s):

thml.1.1: Invl(init, x, y) -> true
          [] Proved by normalization

The induction step in an inductive proof of Conjecture thml.1
    Invl(xst, x, y) -> true
uses the following equation(s) for the induction hypothesis:

Induct.2: Invl(c_xst, x, y) -> true

The system now contains 1 equation, 78 rewrite rules, and 9 deduction rules.

Ordered equation Induct.2 into the rewrite rule:
  ((element(y) = what(x)) <=> false)
   | (false <=> in(y, enqd(c_xst)))
   | (false <=> in_stack(x, deqd(c_xst)))
  -> true

The system now contains 79 rewrite rules and 9 deduction rules.

The induction step involves proving the following lemma(s):

thml.1.2: Invl(deq(c_xst, vil, vi2), x, y) -> true
              which reduces to the equation
              (((trip(element(vi2), enqt(vi2), vil) = x) <=> false)
                & (false <=> in_stack(x, deqd(c_xst))))
               | ((element(y) = what(x)) <=> false)
               | (false <=> in(y, enqd(c_xst)))
               | (vi2 = y)
              -> true
thml.1.3: Invl(enq(c_xst, vil, vi2), x, y) -> true
              which reduces to the equation
              (((pair(vi2, vil) = y) <=> false)
                & (false <=> in(y, enqd(c_xst))))
               | ((element(y) = what(x)) <=> false)
               | (false <=> in_stack(x, deqd(c_xst)))
              -> true
thml.1.4: Invl(commit(c_xst, vil), x, y) -> true
          [] Proved by normalization
thml.1.5: Invl(abort(c_xst, vil), x, y) -> true
              which reduces to the equation
              ((element(y) = what(x)) <=> false)
               | (false <=> in(y, enqd(abort(c_xst, vil))))
               | (false <=> in_stack(x, deqd(c_xst)))
               | (deqr(x) = vil)
              -> true

Proof of Lemma thml.1.5 suspended.

-> resume by case in_stack(x,deqd(c_xst))

Case.4.1
    in_stack(c_x, deqd(c_xst)) == true
```

```
involves proving Lemma thml.1.5.1
    Invl(abort(c_xst, vil), c_x, y) -> true

The case system now contains 1 equation.

Ordered equation Case.4.1 into the rewrite rule:
  in_stack(c_x, deqd(c_xst)) -> true

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 79 rewrite rules, and 9 deduction rules.

Ordered equation Case.4.1 into the rewrite rule:
  in_stack(c_x, deqd(c_xst)) -> true

The system now contains 80 rewrite rules and 9 deduction rules.

Lemma thml.1.5.1 in the proof by cases of Lemma thml.1.5
    Invl(abort(c_xst, vil), c_x, y) -> true
    Case.4.1: in_stack(c_x, deqd(c_xst))
is NOT provable using the current partially completed system.  It reduces to
the equation
     ((element(y) = what(c_x)) <=> false)
      | (false <=> in(y, enqd(abort(c_xst, vil))))
      | (deqr(c_x) = vil)
    -> true

Proof of Lemma thml.1.5.1 suspended.

-> resume by case in(y,enqd(abort(c_xst,vil)))

Case.5.1
    in(c_y, enqd(abort(c_xst, c_vil))) == true
involves proving Lemma thml.1.5.1.1
    Invl(abort(c_xst, c_vil), c_x, c_y) -> true

The case system now contains 1 equation.

Ordered equation Case.5.1 into the rewrite rule:
  in(c_y, enqd(abort(c_xst, c_vil))) -> true

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 80 rewrite rules, and 9 deduction rules.

Ordered equation Case.5.1 into the rewrite rule:
  in(c_y, enqd(abort(c_xst, c_vil))) -> true

The system now contains 81 rewrite rules and 9 deduction rules.

Lemma thml.1.5.1.1 in the proof by cases of Lemma thml.1.5.1
    Invl(abort(c_xst, c_vil), c_x, c_y) -> true
    Case.5.1: in(c_y, enqd(abort(c_xst, c_vil)))
is NOT provable using the current partially completed system.  It reduces to
the equation
     ((element(c_y) = what(c_x)) <=> false) | (c_vil = deqr(c_x)) -> true

Proof of Lemma thml.1.5.1.1 suspended.

-> crit case with State.14

Critical pairs between rule Case.4.1:
  in_stack(c_x, deqd(c_xst)) -> true
and rule State.14:
  (((enqt(x1) = xt) <=> false)
    & ((in_stack(trip(element(x1), enqt(x1), xt), deqd(xst))
         & (((element(x1) = what(x)) <=> false)
              | (false <=> in_stack(x, deqd(xst)))
              | (deqr(x) = xt)))
```

```
        | in(x1, enqd(xst))))

   | (false <=> in(x1, enqd(abort(xst, xt))))
  -> true
  are as follows:
    (((enqt(x1) = xt) <=> false)
     & ((in_stack(trip(element(x1), enqt(x1), xt), deqd(c_xst))
          & (((element(x1) = what(c_x)) <=> false) | (deqr(c_x) = xt)))
         | in(x1, enqd(c_xst))))

    | (false <=> in(x1, enqd(abort(c_xst, xt))))
    == true

The system now contains 1 equation, 81 rewrite rules, and 9 deduction rules.

Ordered equation thm1.2 into the rewrite rule:
  (((enqt(x1) = xt) <=> false)
   & ((in_stack(trip(element(x1), enqt(x1), xt), deqd(c_xst))
        & (((element(x1) = what(c_x)) <=> false) | (deqr(c_x) = xt)))
       | in(x1, enqd(c_xst))))

   | (false <=> in(x1, enqd(abort(c_xst, xt))))
  -> true

The system now contains 82 rewrite rules and 9 deduction rules.

Critical pairs between rule Case.5.1:
  in(c_y, enqd(abort(c_xst, c_vi1))) -> true
and rule State.14:
  (((enqt(x1) = xt) <=> false)
   & ((in_stack(trip(element(x1), enqt(x1), xt), deqd(xst))
        & (((element(x1) = what(x)) <=> false)
            | (false <=> in_stack(x, deqd(xst)))
            | (deqr(x) = xt)))

       | in(x1, enqd(xst))))

   | (false <=> in(x1, enqd(abort(xst, xt))))
  -> true
  are as follows:
    ((enqt(c_y) = xt) <=> false)
     | (false <=> in(c_y, enqd(abort(abort(c_xst, c_vi1), xt))))
    == true
    ((c_vi1 = enqt(c_y)) <=> false)
     & ((in_stack(trip(element(c_y), enqt(c_y), c_vi1), deqd(c_xst))
         & (((element(c_y) = what(x)) <=> false)
             | (false <=> in_stack(x, deqd(c_xst)))
             | (c_vi1 = deqr(x))))

        | in(c_y, enqd(c_xst)))

    == true

The system now contains 1 equation, 82 rewrite rules, and 9 deduction rules.

Ordered equation thm1.3 into the rewrite rule:
  ((enqt(c_y) = xt) <=> false)
   | (false <=> in(c_y, enqd(abort(abort(c_xst, c_vi1), xt))))
  -> true

The system now contains 83 rewrite rules and 9 deduction rules.

The system now contains 1 equation, 83 rewrite rules, and 9 deduction rules.

Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
```

```
has been applied to equation thml.4:
  ((c_vil = enqt(c_y)) <=> false)
   & ((in_stack(trip(element(c_y), enqt(c_y), c_vil), deqd(c_xst))
        & (((element(c_y) = what(x)) <=> false)
            | (false <=> in_stack(x, deqd(c_xst)))
            | (c_vil = deqr(x))))

       | in(c_y, enqd(c_xst)))

  == true
to yield the following equations:
  thml.4.1: (c_vil = enqt(c_y)) <=> false == true
  thml.4.2: (in_stack(trip(element(c_y), enqt(c_y), c_vil), deqd(c_xst))
              & (((element(c_y) = what(x)) <=> false)
                  | (false <=> in_stack(x, deqd(c_xst)))
                  | (c_vil = deqr(x))))

              | in(c_y, enqd(c_xst))
             == true

Ordered equation thml.4.2 into the rewrite rule:
  (in_stack(trip(element(c_y), enqt(c_y), c_vil), deqd(c_xst))
    & (((element(c_y) = what(x)) <=> false)
        | (false <=> in_stack(x, deqd(c_xst)))
        | (c_vil = deqr(x))))

   | in(c_y, enqd(c_xst))
  -> true

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation thml.4.1:
  (c_vil = enqt(c_y)) <=> false == true
to yield the following equations:
  thml.4.1.1: c_vil = enqt(c_y) == false

Ordered equation thml.4.1.1 into the rewrite rule:
  c_vil = enqt(c_y) -> false

The system now contains 85 rewrite rules and 9 deduction rules.

Computed 3 new critical pairs.  Added 3 of them to the system.

-> resume by case in(c_y,enqd(c_xst))

Case.6.1
    in(c_y, enqd(c_xst)) == true
involves proving Lemma thml.1.5.1.1.1
    Invl(abort(c_xst, c_vil), c_x, c_y) -> true

The case system now contains 1 equation.

Ordered equation Case.6.1 into the rewrite rule:
  in(c_y, enqd(c_xst)) -> true

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 85 rewrite rules, and 9 deduction rules.

Ordered equation Case.6.1 into the rewrite rule:
  in(c_y, enqd(c_xst)) -> true

    Left-hand side reduced:
    (in_stack(trip(element(c_y), enqt(c_y), c_vil), deqd(c_xst))
      & (((element(c_y) = what(x)) <=> false)
          | (false <=> in_stack(x, deqd(c_xst)))
          | (c_vil = deqr(x))))
```

```
      | in(c_y, enqd(c_xst))
     -> true
      became equation thm1.4.2:
       (in_stack(trip(element(c_y), enqt(c_y), c_vil), deqd(c_xst))
         & (((element(c_y) = what(x)) <=> false)
              | (false <=> in_stack(x, deqd(c_xst)))
              | (c_vil = deqr(x))))

        | true
       == true

The system now contains 85 rewrite rules and 9 deduction rules.

Lemma thm1.1.5.1.1.1 in the proof by cases of Lemma thm1.1.5.1.1
    Inv1(abort(c_xst, c_vil), c_x, c_y) -> true
    Case.6.1: in(c_y, enqd(c_xst))
is NOT provable using the current partially completed system.  It reduces to
the equation
    ((element(c_y) = what(c_x)) <=> false) | (c_vil = deqr(c_x)) -> true

Proof of Lemma thm1.1.5.1.1.1 suspended.

-> crit case with induct

Critical pairs between rule Case.4.1:
  in_stack(c_x, deqd(c_xst)) -> true
and rule Induct.2:
  ((element(y) = what(x)) <=> false)
   | (false <=> in(y, enqd(c_xst)))
   | (false <=> in_stack(x, deqd(c_xst)))
  -> true
  are as follows:
    ((element(y) = what(c_x)) <=> false) | (false <=> in(y, enqd(c_xst)))
    == true

The system now contains 1 equation, 85 rewrite rules, and 9 deduction rules.

Ordered equation thm1.5 into the rewrite rule:
  ((element(y) = what(c_x)) <=> false) | (false <=> in(y, enqd(c_xst))) -> true

The system now contains 86 rewrite rules and 9 deduction rules.

Critical pairs between rule Case.6.1:
  in(c_y, enqd(c_xst)) -> true
and rule Induct.2:
  ((element(y) = what(x)) <=> false)
   | (false <=> in(y, enqd(c_xst)))
   | (false <=> in_stack(x, deqd(c_xst)))
  -> true
  are as follows:
    ((element(c_y) = what(x)) <=> false) | (false <=> in_stack(x, deqd(c_xst)))
    == true

The system now contains 1 equation, 86 rewrite rules, and 9 deduction rules.

Ordered equation thm1.6 into the rewrite rule:
  ((element(c_y) = what(x)) <=> false) | (false <=> in_stack(x, deqd(c_xst)))
  -> true

The system now contains 87 rewrite rules and 9 deduction rules.

Computed 2 new critical pairs.  Added 2 of them to the system.

-> crit case with thm1

Critical pairs between rule Case.4.1:
  in_stack(c_x, deqd(c_xst)) -> true
and rule thm1.6:
  ((element(c_y) = what(x)) <=> false) | (false <=> in_stack(x, deqd(c_xst)))
```

```
  -> true
  are as follows:
    (element(c_y) = what(c_x)) <=> false == true

The system now contains 1 equation, 87 rewrite rules, and 9 deduction rules.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation thml.7:
  (element(c_y) = what(c_x)) <=> false == true
to yield the following equations:
  thml.7.1: element(c_y) = what(c_x) == false

Ordered equation thml.7.1 into the rewrite rule:
  element(c_y) = what(c_x) -> false

The system now contains 88 rewrite rules and 9 deduction rules.

Lemma thml.1.5.1.1.1 in the proof by cases of Lemma thml.1.5.1.1
    Invl(abort(c_xst, c_vil), c_x, c_y) -> true
    Case.6.1: in(c_y, enqd(c_xst))
[] Proved by rewriting.

Case.6.2
    not(in(c_y, enqd(c_xst))) == true
involves proving Lemma thml.1.5.1.1.2
    Invl(abort(c_xst, c_vil), c_x, c_y) -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.6.2:
  false <=> in(c_y, enqd(c_xst)) == true
to yield the following equations:
  Case.6.2.1: false == in(c_y, enqd(c_xst))

Ordered equation Case.6.2.1 into the rewrite rule:
  in(c_y, enqd(c_xst)) -> false

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 85 rewrite rules, and 9 deduction rules.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.6.2:
  false <=> in(c_y, enqd(c_xst)) == true
to yield the following equations:
  Case.6.2.2: false == in(c_y, enqd(c_xst))

Ordered equation Case.6.2.2 into the rewrite rule:
  in(c_y, enqd(c_xst)) -> false

    Left-hand side reduced:
    (in_stack(trip(element(c_y), enqt(c_y), c_vil), deqd(c_xst))
      & (((element(c_y) = what(x)) <=> false)
          | (false <=> in_stack(x, deqd(c_xst)))
          | (c_vil = deqr(x))))

     | in(c_y, enqd(c_xst))
    -> true
      became equation thml.4.2:
      (in_stack(trip(element(c_y), enqt(c_y), c_vil), deqd(c_xst))
        & (((element(c_y) = what(x)) <=> false)
            | (false <=> in_stack(x, deqd(c_xst)))
```

```
            | (c_vil = deqr(x))))

        | false
       == true

Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation thml.4.2:
  in_stack(trip(element(c_y), enqt(c_y), c_vil), deqd(c_xst))
   & (((element(c_y) = what(x)) <=> false)
       | (false <=> in_stack(x, deqd(c_xst)))
       | (c_vil = deqr(x)))

  == true
to yield the following equations:
  thml.4.2.1: in_stack(trip(element(c_y), enqt(c_y), c_vil), deqd(c_xst))
              == true
  thml.4.2.2: ((element(c_y) = what(x)) <=> false)
               | (false <=> in_stack(x, deqd(c_xst)))
               | (c_vil = deqr(x))
              == true

Ordered equation thml.4.2.2 into the rewrite rule:
  ((element(c_y) = what(x)) <=> false)
   | (false <=> in_stack(x, deqd(c_xst)))
   | (c_vil = deqr(x))
  -> true

Ordered equation thml.4.2.1 into the rewrite rule:
  in_stack(trip(element(c_y), enqt(c_y), c_vil), deqd(c_xst)) -> true

The system now contains 87 rewrite rules and 9 deduction rules.

Lemma thml.1.5.1.1.2 in the proof by cases of Lemma thml.1.5.1.1
    Invl(abort(c_xst, c_vil), c_x, c_y) -> true
    Case.6.2: not(in(c_y, enqd(c_xst)))
is NOT provable using the current partially completed system.  It reduces to
the equation
    ((element(c_y) = what(c_x)) <=> false) | (c_vil = deqr(c_x)) -> true

Proof of Lemma thml.1.5.1.1.2 suspended.

Critical-pair computation abandoned because a theorem has been proved.

Computed 1 new critical pair.  Added 1 of them to the system.

-> resume by case in_stack(x,deqd(c_xst))

Case.7.1
    in_stack(c_xl, deqd(c_xst)) == true
involves proving Lemma thml.1.5.1.1.2.1
    Invl(abort(c_xst, c_vil), c_x, c_y) -> true

The case system now contains 1 equation.

Ordered equation Case.7.1 into the rewrite rule:
  in_stack(c_xl, deqd(c_xst)) -> true

The case system now contains 1 rewrite rule.

Lemma thml.1.5.1.1.2.1 in the proof by cases of Lemma thml.1.5.1.1.2
    Invl(abort(c_xst, c_vil), c_x, c_y) -> true
    Case.7.1: in_stack(c_xl, deqd(c_xst))
[] Proved by rewriting (with unreduced rules).

Case.7.2
    not(in_stack(c_xl, deqd(c_xst))) == true
```

```
involves proving Lemma thm1.1.5.1.1.2.2
    Inv1(abort(c_xst, c_vil), c_x, c_y) -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.7.2:
  false <=> in_stack(c_x1, deqd(c_xst)) == true
to yield the following equations:
  Case.7.2.1: false == in_stack(c_x1, deqd(c_xst))

Ordered equation Case.7.2.1 into the rewrite rule:
  in_stack(c_x1, deqd(c_xst)) -> false

The case system now contains 1 rewrite rule.

Lemma thm1.1.5.1.1.2.2 in the proof by cases of Lemma thm1.1.5.1.1.2
    Inv1(abort(c_xst, c_vil), c_x, c_y) -> true
    Case.7.2: not(in_stack(c_x1, deqd(c_xst)))
[] Proved by rewriting (with unreduced rules).

Lemma thm1.1.5.1.1.2 in the proof by cases of Lemma thm1.1.5.1.1
    Inv1(abort(c_xst, c_vil), c_x, c_y) -> true
    Case.6.2: not(in(c_y, enqd(c_xst)))
[] Proved by cases
    in_stack(x, deqd(c_xst)) | not(in_stack(x, deqd(c_xst)))

Lemma thm1.1.5.1.1 in the proof by cases of Lemma thm1.1.5.1
    Inv1(abort(c_xst, c_vil), c_x, c_y) -> true
    Case.5.1: in(c_y, enqd(abort(c_xst, c_vil)))
[] Proved by cases
    in(c_y, enqd(c_xst)) | not(in(c_y, enqd(c_xst)))

Case.5.2
    not(in(c_y, enqd(abort(c_xst, c_vil)))) == true
involves proving Lemma thm1.1.5.1.2
    Inv1(abort(c_xst, c_vil), c_x, c_y) -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.5.2:
  false <=> in(c_y, enqd(abort(c_xst, c_vil))) == true
to yield the following equations:
  Case.5.2.1: false == in(c_y, enqd(abort(c_xst, c_vil)))

Ordered equation Case.5.2.1 into the rewrite rule:
  in(c_y, enqd(abort(c_xst, c_vil))) -> false

The case system now contains 1 rewrite rule.

Lemma thm1.1.5.1.2 in the proof by cases of Lemma thm1.1.5.1
    Inv1(abort(c_xst, c_vil), c_x, c_y) -> true
    Case.5.2: not(in(c_y, enqd(abort(c_xst, c_vil))))
[] Proved by rewriting (with unreduced rules).

Lemma thm1.1.5.1 in the proof by cases of Lemma thm1.1.5
    Inv1(abort(c_xst, vil), c_x, y) -> true
    Case.4.1: in_stack(c_x, deqd(c_xst))
[] Proved by cases
    in(y, enqd(abort(c_xst, vil))) | not(in(y, enqd(abort(c_xst, vil))))

Case.4.2
    not(in_stack(c_x, deqd(c_xst))) == true
involves proving Lemma thm1.1.5.2
```

```
    Invl(abort(c_xst, vi1), c_x, y) -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.4.2:
  false <=> in_stack(c_x, deqd(c_xst)) == true
to yield the following equations:
  Case.4.2.1: false == in_stack(c_x, deqd(c_xst))

Ordered equation Case.4.2.1 into the rewrite rule:
  in_stack(c_x, deqd(c_xst)) -> false

The case system now contains 1 rewrite rule.

Lemma thml.1.5.2 in the proof by cases of Lemma thml.1.5
    Invl(abort(c_xst, vi1), c_x, y) -> true
    Case.4.2: not(in_stack(c_x, deqd(c_xst)))
[] Proved by rewriting (with unreduced rules).

Lemma thml.1.5 for the induction step in the proof of Conjecture thml.1
    Invl(abort(c_xst, vi1), x, y) -> true
[] Proved by cases
    in_stack(x, deqd(c_xst)) | not(in_stack(x, deqd(c_xst)))

Lemma thml.1.3 for the induction step in the proof of Conjecture thml.1
    Invl(enq(c_xst, vi1, vi2), x, y) -> true
is NOT provable using the current partially completed system.  It reduces to
the equation
     (((pair(vi2, vi1) = y) <=> false) & (false <=> in(y, enqd(c_xst))))
      | ((element(y) = what(x)) <=> false)
      | (false <=> in_stack(x, deqd(c_xst)))
     -> true

Proof of Lemma thml.1.3 suspended.

-> resume by case in(y,enqd(c_xst))

Case.8.1
    in(c_y, enqd(c_xst)) == true
involves proving Lemma thml.1.3.1
    Invl(enq(c_xst, vi1, vi2), x, c_y) -> true

The case system now contains 1 equation.

Ordered equation Case.8.1 into the rewrite rule:
  in(c_y, enqd(c_xst)) -> true

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 79 rewrite rules, and 9 deduction rules.

Ordered equation Case.8.1 into the rewrite rule:
  in(c_y, enqd(c_xst)) -> true

The system now contains 80 rewrite rules and 9 deduction rules.

Lemma thml.1.3.1 in the proof by cases of Lemma thml.1.3
    Invl(enq(c_xst, vi1, vi2), x, c_y) -> true
    Case.8.1: in(c_y, enqd(c_xst))
is NOT provable using the current partially completed system.  It reduces to
the equation
    ((element(c_y) = what(x)) <=> false) | (false <=> in_stack(x, deqd(c_xst)))
    -> true

Proof of Lemma thml.1.3.1 suspended.
```

```
-> resume by case in_stack(x,deqd(c_xst))

Case.9.1
    in_stack(c_x, deqd(c_xst)) == true
involves proving Lemma thm1.1.3.1.1
    Inv1(enq(c_xst, vi1, vi2), c_x, c_y) -> true

The case system now contains 1 equation.

Ordered equation Case.9.1 into the rewrite rule:
  in_stack(c_x, deqd(c_xst)) -> true

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 80 rewrite rules, and 9 deduction rules.

Ordered equation Case.9.1 into the rewrite rule:
  in_stack(c_x, deqd(c_xst)) -> true

The system now contains 81 rewrite rules and 9 deduction rules.

Lemma thm1.1.3.1.1 in the proof by cases of Lemma thm1.1.3.1
    Inv1(enq(c_xst, vi1, vi2), c_x, c_y) -> true
    Case.9.1: in_stack(c_x, deqd(c_xst))
is NOT provable using the current partially completed system.  It reduces to
the equation
    (element(c_y) = what(c_x)) <=> false -> true

Proof of Lemma thm1.1.3.1.1 suspended.

-> crit case with induct

Critical pairs between rule Case.8.1:
  in(c_y, enqd(c_xst)) -> true
and rule Induct.2:
  ((element(y) = what(x)) <=> false)
   | (false <=> in(y, enqd(c_xst)))
   | (false <=> in_stack(x, deqd(c_xst)))
  -> true
  are as follows:
    ((element(c_y) = what(x)) <=> false) | (false <=> in_stack(x, deqd(c_xst)))
    == true

The system now contains 1 equation, 81 rewrite rules, and 9 deduction rules.

Ordered equation thm1.8 into the rewrite rule:
  ((element(c_y) = what(x)) <=> false) | (false <=> in_stack(x, deqd(c_xst)))
  -> true

The system now contains 82 rewrite rules and 9 deduction rules.

Critical pairs between rule Case.9.1:
  in_stack(c_x, deqd(c_xst)) -> true
and rule Induct.2:
  ((element(y) = what(x)) <=> false)
   | (false <=> in(y, enqd(c_xst)))
   | (false <=> in_stack(x, deqd(c_xst)))
  -> true
  are as follows:
    ((element(y) = what(c_x)) <=> false) | (false <=> in(y, enqd(c_xst)))
    == true

The system now contains 1 equation, 82 rewrite rules, and 9 deduction rules.

Ordered equation thm1.9 into the rewrite rule:
  ((element(y) = what(c_x)) <=> false) | (false <=> in(y, enqd(c_xst))) -> true

The system now contains 83 rewrite rules and 9 deduction rules.
```

```
Computed 2 new critical pairs.  Added 2 of them to the system.

-> crit case with thm1

Critical pairs between rule Case.8.1:
  in(c_y, enqd(c_xst)) -> true
and rule thm1.9:
  ((element(y) = what(c_x)) <=> false) | (false <=> in(y, enqd(c_xst))) -> true
  are as follows:
    (element(c_y) = what(c_x)) <=> false == true

The system now contains 1 equation, 83 rewrite rules, and 9 deduction rules.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation thm1.10:
  (element(c_y) = what(c_x)) <=> false == true
to yield the following equations:
  thm1.10.1: element(c_y) = what(c_x) == false

Ordered equation thm1.10.1 into the rewrite rule:
  element(c_y) = what(c_x) -> false

The system now contains 84 rewrite rules and 9 deduction rules.

Lemma thm1.1.3.1.1 in the proof by cases of Lemma thm1.1.3.1
    Inv1(enq(c_xst, vi1, vi2), c_x, c_y) -> true
    Case.9.1: in_stack(c_x, deqd(c_xst))
[] Proved by rewriting.

Case.9.2
    not(in_stack(c_x, deqd(c_xst))) == true
involves proving Lemma thm1.1.3.1.2
    Inv1(enq(c_xst, vi1, vi2), c_x, c_y) -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.9.2:
  false <=> in_stack(c_x, deqd(c_xst)) == true
to yield the following equations:
  Case.9.2.1: false == in_stack(c_x, deqd(c_xst))

Ordered equation Case.9.2.1 into the rewrite rule:
  in_stack(c_x, deqd(c_xst)) -> false

The case system now contains 1 rewrite rule.

Lemma thm1.1.3.1.2 in the proof by cases of Lemma thm1.1.3.1
    Inv1(enq(c_xst, vi1, vi2), c_x, c_y) -> true
    Case.9.2: not(in_stack(c_x, deqd(c_xst)))
[] Proved by rewriting (with unreduced rules).

Lemma thm1.1.3.1 in the proof by cases of Lemma thm1.1.3
    Inv1(enq(c_xst, vi1, vi2), x, c_y) -> true
    Case.8.1: in(c_y, enqd(c_xst))
[] Proved by cases
    in_stack(x, deqd(c_xst)) | not(in_stack(x, deqd(c_xst)))

Case.8.2
    not(in(c_y, enqd(c_xst))) == true
involves proving Lemma thm1.1.3.2
    Inv1(enq(c_xst, vi1, vi2), x, c_y) -> true

The case system now contains 1 equation.
```

```
Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.8.2:
  false <=> in(c_y, enqd(c_xst)) == true
to yield the following equations:
  Case.8.2.1: false == in(c_y, enqd(c_xst))

Ordered equation Case.8.2.1 into the rewrite rule:
  in(c_y, enqd(c_xst)) -> false

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 79 rewrite rules, and 9 deduction rules.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.8.2:
  false <=> in(c_y, enqd(c_xst)) == true
to yield the following equations:
  Case.8.2.2: false == in(c_y, enqd(c_xst))

Ordered equation Case.8.2.2 into the rewrite rule:
  in(c_y, enqd(c_xst)) -> false

The system now contains 80 rewrite rules and 9 deduction rules.

Lemma thm1.1.3.2 in the proof by cases of Lemma thm1.1.3
    Inv1(enq(c_xst, vi1, vi2), x, c_y) -> true
    Case.8.2: not(in(c_y, enqd(c_xst)))
is NOT provable using the current partially completed system.  It reduces to
the equation
    ((c_y = pair(vi2, vi1)) <=> false)
     | ((element(c_y) = what(x)) <=> false)
     | (false <=> in_stack(x, deqd(c_xst)))
    -> true

Proof of Lemma thm1.1.3.2 suspended.

Critical-pair computation abandoned because a theorem has been proved.

Computed 1 new critical pair.  Added 1 of them to the system.

-> add when_enq(c_xst, z,w,vi1,vi2::EL)

Added 1 equation to the system.

Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation thm1.11:
  ((enqr(top(deqd(c_xst))) < vi1) | (deqd(c_xst) = new))
   & (((element(z) = vi2) <=> false) | (false <=> in(z, enqd(c_xst))))
   & (((what(w) = vi2) <=> false) | (false <=> in_stack(w, deqd(c_xst))))
  -> true
to yield the following equations:
  thm1.11.1: (enqr(top(deqd(c_xst))) < vi1) | (deqd(c_xst) = new) == true
  thm1.11.2: ((element(z) = vi2) <=> false) | (false <=> in(z, enqd(c_xst)))
             == true
  thm1.11.3: ((what(w) = vi2) <=> false) | (false <=> in_stack(w, deqd(c_xst)))
             == true

Ordered equation thm1.11.3 into the rewrite rule:
  ((what(w) = vi2) <=> false) | (false <=> in_stack(w, deqd(c_xst))) -> true

    Left-hand side reduced:
    ((element(y) = what(x)) <=> false)
```

```
     | (false <=> in(y, enqd(c_xst)))
     | (false <=> in_stack(x, deqd(c_xst)))
    -> true
      became equation Induct.2:
      (false <=> in(y, enqd(c_xst))) | true -> true

Ordered equation thm1.11.2 into the rewrite rule:
  ((element(z) = vi2) <=> false) | (false <=> in(z, enqd(c_xst))) -> true

Ordered equation thm1.11.1 into the rewrite rule:
  (enqr(top(deqd(c_xst))) < vi1) | (deqd(c_xst) = new) -> true

The system now contains 82 rewrite rules and 9 deduction rules.

Lemma thm1.1.3.2 in the proof by cases of Lemma thm1.1.3
    Inv1(enq(c_xst, vi1, vi2), x, c_y) -> true
    Case.8.2: not(in(c_y, enqd(c_xst)))
[] Proved by rewriting.

Lemma thm1.1.3 for the induction step in the proof of Conjecture thm1.1
    Inv1(enq(c_xst, vi1, vi2), x, y) -> true
[] Proved by cases
    in(y, enqd(c_xst)) | not(in(y, enqd(c_xst)))

Lemma thm1.1.2 for the induction step in the proof of Conjecture thm1.1
    Inv1(deq(c_xst, vi1, vi2), x, y) -> true
is NOT provable using the current partially completed system.  It reduces to
the equation
    (((trip(element(vi2), enqt(vi2), vi1) = x) <=> false)
      & (false <=> in_stack(x, deqd(c_xst))))
     | ((element(y) = what(x)) <=> false)
     | (false <=> in(y, enqd(c_xst)))
     | (vi2 = y)
    -> true

Proof of Lemma thm1.1.2 suspended.

->  resume by case in_stack(x,deqd(c_xst))

Case.16.1
    in_stack(c_x, deqd(c_xst)) == true
involves proving Lemma thm1.1.2.3
    Inv1(deq(c_xst, vi1, vi2), c_x, y) -> true

The case system now contains 1 equation.

Ordered equation Case.16.1 into the rewrite rule:
  in_stack(c_x, deqd(c_xst)) -> true

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 79 rewrite rules, and 9 deduction rules.

Ordered equation Case.16.1 into the rewrite rule:
  in_stack(c_x, deqd(c_xst)) -> true

The system now contains 80 rewrite rules and 9 deduction rules.

Lemma thm1.1.2.3 in the proof by cases of Lemma thm1.1.2
    Inv1(deq(c_xst, vi1, vi2), c_x, y) -> true
    Case.16.1: in_stack(c_x, deqd(c_xst))
is NOT provable using the current partially completed system.  It reduces to
the equation
    ((element(y) = what(c_x)) <=> false)
     | (false <=> in(y, enqd(c_xst)))
     | (vi2 = y)
    -> true

Proof of Lemma thm1.1.2.3 suspended.
```

```
-> crit case with induct

Critical pairs between rule Case.16.1:
  in_stack(c_x, deqd(c_xst)) -> true
and rule Induct.2:
  ((element(y) = what(x)) <=> false)
   | (false <=> in(y, enqd(c_xst)))
   | (false <=> in_stack(x, deqd(c_xst)))
  -> true
  are as follows:
    ((element(y) = what(c_x)) <=> false) | (false <=> in(y, enqd(c_xst)))
    == true

The system now contains 1 equation, 80 rewrite rules, and 9 deduction rules.

Ordered equation thm1.18 into the rewrite rule:
  ((element(y) = what(c_x)) <=> false) | (false <=> in(y, enqd(c_xst))) -> true

The system now contains 81 rewrite rules and 9 deduction rules.

Lemma thm1.1.2.3 in the proof by cases of Lemma thm1.1.2
    Inv1(deq(c_xst, vi1, vi2), c_x, y) -> true
    Case.16.1: in_stack(c_x, deqd(c_xst))
[] Proved by rewriting.

Case.16.2
    not(in_stack(c_x, deqd(c_xst))) == true
involves proving Lemma thm1.1.2.4
    Inv1(deq(c_xst, vi1, vi2), c_x, y) -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.16.2:
  false <=> in_stack(c_x, deqd(c_xst)) == true
to yield the following equations:
  Case.16.2.1: false == in_stack(c_x, deqd(c_xst))

Ordered equation Case.16.2.1 into the rewrite rule:
  in_stack(c_x, deqd(c_xst)) -> false

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 79 rewrite rules, and 9 deduction rules.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.16.2:
  false <=> in_stack(c_x, deqd(c_xst)) == true
to yield the following equations:
  Case.16.2.2: false == in_stack(c_x, deqd(c_xst))

Ordered equation Case.16.2.2 into the rewrite rule:
  in_stack(c_x, deqd(c_xst)) -> false

The system now contains 80 rewrite rules and 9 deduction rules.

Lemma thm1.1.2.4 in the proof by cases of Lemma thm1.1.2
    Inv1(deq(c_xst, vi1, vi2), c_x, y) -> true
    Case.16.2: not(in_stack(c_x, deqd(c_xst)))
is NOT provable using the current partially completed system.  It reduces to
the equation
    ((c_x = trip(element(vi2), enqt(vi2), vi1)) <=> false)
     | ((element(y) = what(c_x)) <=> false)
     | (false <=> in(y, enqd(c_xst)))
     | (vi2 = y)
```

```
    -> true

Proof of Lemma thml.1.2.4 suspended.

Critical-pair computation abandoned because a theorem has been proved.

Computed 1 new critical pair.  Added 1 of them to the system.

->  resume by case c_x=trip(element(vi2::enq_rec),enqt(vi2::enq_rec),vi1)

Case.17.1
    c_x = trip(element(c_vi2), enqt(c_vi2), c_vi1) == true
involves proving Lemma thml.1.2.4.1
    Inv1(deq(c_xst, c_vi1, c_vi2), c_x, y) -> true

The case system now contains 1 equation.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.17.1:
  c_x = trip(element(c_vi2), enqt(c_vi2), c_vi1) == true
to yield the following equations:
  Case.17.1.1: c_x == trip(element(c_vi2), enqt(c_vi2), c_vi1)

Ordered equation Case.17.1.1 into the rewrite rule:
  c_x -> trip(element(c_vi2), enqt(c_vi2), c_vi1)

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 80 rewrite rules, and 9 deduction rules.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.17.1:
  c_x = trip(element(c_vi2), enqt(c_vi2), c_vi1) == true
to yield the following equations:
  Case.17.1.2: c_x == trip(element(c_vi2), enqt(c_vi2), c_vi1)

Ordered equation Case.17.1.2 into the rewrite rule:
  c_x -> trip(element(c_vi2), enqt(c_vi2), c_vi1)

    Left-hand side reduced:
    in_stack(c_x, deqd(c_xst)) -> false
      became equation Case.16.2.2:
      in_stack(trip(element(c_vi2), enqt(c_vi2), c_vi1), deqd(c_xst)) == false

Ordered equation Case.16.2.2 into the rewrite rule:
  in_stack(trip(element(c_vi2), enqt(c_vi2), c_vi1), deqd(c_xst)) -> false

The system now contains 81 rewrite rules and 9 deduction rules.

Lemma thml.1.2.4.1 in the proof by cases of Lemma thml.1.2.4
    Inv1(deq(c_xst, c_vi1, c_vi2), c_x, y) -> true
    Case.17.1: c_x = trip(element(c_vi2), enqt(c_vi2), c_vi1)
is NOT provable using the current partially completed system.  It reduces to
the equation
    ((element(c_vi2) = element(y)) <=> false)
     | (false <=> in(y, enqd(c_xst)))
     | (c_vi2 = y)
    -> true

Proof of Lemma thml.1.2.4.1 suspended.

-> resume by case in(y,enqd(c_xst))

Case.18.1
    in(c_y, enqd(c_xst)) == true
```

```
involves proving Lemma thm1.1.2.4.1.1
    Invl(deq(c_xst, c_vi1, c_vi2), c_x, c_y) -> true

The case system now contains 1 equation.

Ordered equation Case.18.1 into the rewrite rule:
  in(c_y, enqd(c_xst)) -> true

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 81 rewrite rules, and 9 deduction rules.

Ordered equation Case.18.1 into the rewrite rule:
  in(c_y, enqd(c_xst)) -> true

The system now contains 82 rewrite rules and 9 deduction rules.

Lemma thm1.1.2.4.1.1 in the proof by cases of Lemma thm1.1.2.4.1
    Invl(deq(c_xst, c_vi1, c_vi2), c_x, c_y) -> true
    Case.18.1: in(c_y, enqd(c_xst))
is NOT provable using the current partially completed system.  It reduces to
the equation
     ((element(c_vi2) = element(c_y)) <=> false) | (c_vi2 = c_y) -> true

Proof of Lemma thm1.1.2.4.1.1 suspended.

-> crit case with Induct

Critical pairs between rule Case.18.1:
  in(c_y, enqd(c_xst)) -> true
and rule Induct.2:
  ((element(y) = what(x)) <=> false)
   | (false <=> in(y, enqd(c_xst)))
   | (false <=> in_stack(x, deqd(c_xst)))
  -> true
  are as follows:
    ((element(c_y) = what(x)) <=> false) | (false <=> in_stack(x, deqd(c_xst)))
    == true

The system now contains 1 equation, 82 rewrite rules, and 9 deduction rules.

Ordered equation thm1.19 into the rewrite rule:
  ((element(c_y) = what(x)) <=> false) | (false <=> in_stack(x, deqd(c_xst)))
  -> true

The system now contains 83 rewrite rules and 9 deduction rules.

Computed 2 new critical pairs, 1 of which reduced to an identity.  Added 1 of
them to the system.

-> add when_deq(c_xst,c_x,c_vi1,c_vi2::enq_rec)

Added 1 equation to the system.

Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation thm1.20:
  (enqt(c_vi2) < c_vi1)
   & in(c_vi2, enqd(c_xst))
   & least(c_vi2, enqd(c_xst))
   & (((deqr(top(deqd(c_xst))) < c_vi1)
        & (enqr(top(deqd(c_xst))) < enqt(c_vi2)))
       | (deqd(c_xst) = new))

  -> true
to yield the following equations:
  thm1.20.1: enqt(c_vi2) < c_vi1 == true
```

```
  thml.20.2: in(c_vi2, enqd(c_xst)) == true
  thml.20.3: least(c_vi2, enqd(c_xst)) == true
  thml.20.4: ((deqr(top(deqd(c_xst))) < c_vil)
               & (enqr(top(deqd(c_xst))) < enqt(c_vi2)))
              | (deqd(c_xst) = new)
             == true

Ordered equation thml.20.4 into the rewrite rule:
  ((deqr(top(deqd(c_xst))) < c_vil) & (enqr(top(deqd(c_xst))) < enqt(c_vi2)))
   | (deqd(c_xst) = new)
  -> true

Ordered equation thml.20.3 into the rewrite rule:
  least(c_vi2, enqd(c_xst)) -> true

Ordered equation thml.20.2 into the rewrite rule:
  in(c_vi2, enqd(c_xst)) -> true

Ordered equation thml.20.1 into the rewrite rule:
  enqt(c_vi2) < c_vil -> true

The system now contains 87 rewrite rules and 9 deduction rules.

->  crit case with thml

Computed 1 new critical pair, which reduced to an identity.  Added 0 of them to
the system.

-> resume by case c_vi2=c_y

Case.19.1
    c_vi2 = c_y == true
involves proving Lemma thml.1.2.4.1.1.1
    Invl(deq(c_xst, c_vil, c_vi2), c_x, c_y) -> true

The case system now contains 1 equation.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.19.1:
  c_vi2 = c_y == true
to yield the following equations:
  Case.19.1.1: c_vi2 == c_y

Ordered equation Case.19.1.1 into the rewrite rule:
  c_vi2 -> c_y

The case system now contains 1 rewrite rule.

Lemma thml.1.2.4.1.1.1 in the proof by cases of Lemma thml.1.2.4.1.1
    Invl(deq(c_xst, c_vil, c_vi2), c_x, c_y) -> true
    Case.19.1: c_vi2 = c_y
[] Proved by rewriting (with unreduced rules).

Case.19.2
    not(c_vi2 = c_y) == true
involves proving Lemma thml.1.2.4.1.1.2
    Invl(deq(c_xst, c_vil, c_vi2), c_x, c_y) -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.19.2:
  (c_vi2 = c_y) <=> false == true
to yield the following equations:
  Case.19.2.1: c_vi2 = c_y == false
```

```
Ordered equation Case.19.2.1 into the rewrite rule:
  c_vi2 = c_y -> false

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 87 rewrite rules, and 9 deduction rules.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.19.2:
  (c_vi2 = c_y) <=> false == true
to yield the following equations:
  Case.19.2.2: c_vi2 = c_y == false

Ordered equation Case.19.2.2 into the rewrite rule:
  c_vi2 = c_y -> false

The system now contains 88 rewrite rules and 9 deduction rules.

Lemma thm1.1.2.4.1.1.2 in the proof by cases of Lemma thm1.1.2.4.1.1
    Invl(deq(c_xst, c_vi1, c_vi2), c_x, c_y) -> true
    Case.19.2: not(c_vi2 = c_y)
is NOT provable using the current partially completed system.  It reduces to
the equation
    (element(c_vi2) = element(c_y)) <=> false -> true

Proof of Lemma thm1.1.2.4.1.1.2 suspended.

->  prove not(element(x)=element(y))=>not(x=y)

Conjecture thm1.21
    not(element(x) = element(y)) => not(x = y) -> true
is NOT provable using the current partially completed system.  It reduces to
the equation
    ((x = y) <=> false) | (element(x) = element(y)) -> true

Proof of Conjecture thm1.21 suspended.

-> resume by case x=y

Case.20.1
    c_x1 = c_y1 == true
involves proving Lemma thm1.21.1
    not(element(c_x1) = element(c_y1)) => not(c_x1 = c_y1) -> true

The case system now contains 1 equation.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.20.1:
  c_x1 = c_y1 == true
to yield the following equations:
  Case.20.1.1: c_x1 == c_y1

Ordered equation Case.20.1.1 into the rewrite rule:
  c_x1 -> c_y1

The case system now contains 1 rewrite rule.

Lemma thm1.21.1 in the proof by cases of Conjecture thm1.21
    not(element(c_x1) = element(c_y1)) => not(c_x1 = c_y1) -> true
    Case.20.1: c_x1 = c_y1
[] Proved by rewriting (with unreduced rules).

Case.20.2
    not(c_x1 = c_y1) == true
```

```
involves proving Lemma thm1.21.2
    not(element(c_x1) = element(c_y1)) => not(c_x1 = c_y1) -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.20.2:
  (c_x1 = c_y1) <=> false == true
to yield the following equations:
  Case.20.2.1: c_x1 = c_y1 == false

Ordered equation Case.20.2.1 into the rewrite rule:
  c_x1 = c_y1 -> false

The case system now contains 1 rewrite rule.

Lemma thm1.21.2 in the proof by cases of Conjecture thm1.21
    not(element(c_x1) = element(c_y1)) => not(c_x1 = c_y1) -> true
    Case.20.2: not(c_x1 = c_y1)
[] Proved by rewriting (with unreduced rules).

Conjecture thm1.21
    not(element(x) = element(y)) => not(x = y) -> true
[] Proved by cases
     (x = y) | not(x = y)

The system now contains 1 equation, 88 rewrite rules, and 9 deduction rules.

Ordered equation thm1.21 into the rewrite rule:
  ((x = y) <=> false) | (element(x) = element(y)) -> true

The system now contains 89 rewrite rules and 9 deduction rules.

Lemma thm1.1.2.4.1.1.2 in the proof by cases of Lemma thm1.1.2.4.1.1
    Inv1(deq(c_xst, c_vi1, c_vi2), c_x, c_y) -> true
    Case.19.2: not(c_vi2 = c_y)
is NOT provable using the current partially completed system.  It reduces to
the equation
     (element(c_vi2) = element(c_y)) <=> false -> true

Proof of Lemma thm1.1.2.4.1.1.2 suspended.

-> resume by case element(c_vi2)=element(c_y)

Case.21.1
    element(c_vi2) = element(c_y) == true
involves proving Lemma thm1.1.2.4.1.1.2.1
    Inv1(deq(c_xst, c_vi1, c_vi2), c_x, c_y) -> true

The case system now contains 1 equation.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.21.1:
  element(c_vi2) = element(c_y) == true
to yield the following equations:
  Case.21.1.1: element(c_vi2) == element(c_y)

Ordered equation Case.21.1.1 into the rewrite rule:
  element(c_vi2) -> element(c_y)

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 89 rewrite rules, and 9 deduction rules.

Deduction rule equality.4:
```

```
  when  x = y == true
  yield x == y
has been applied to equation Case.21.1:
  element(c_vi2) = element(c_y) == true
to yield the following equations:
  Case.21.1.2: element(c_vi2) == element(c_y)

Deduction rule Enq_Rec.1:
  when  element(xn) == element(xn1)
  yield xn == xn1
has been applied to equation Case.21.1.2:
  element(c_vi2) == element(c_y)
to yield the following equations, which imply the original equation:
  Case.21.1.2.1: c_vi2 == c_y

Ordered equation Case.21.1.2 into the rewrite rule:
  element(c_vi2) -> element(c_y)

    Left-hand side reduced:
    in_stack(trip(element(c_vi2), enqt(c_vi2), c_vi1), deqd(c_xst)) -> false
      became equation Case.16.2.2:
      in_stack(trip(element(c_y), enqt(c_vi2), c_vi1), deqd(c_xst)) == false

Ordered equation Case.21.1.2.1 into the rewrite rule:
  c_vi2 -> c_y

    Following 6 left-hand sides reduced:
    ((deqr(top(deqd(c_xst))) < c_vi1) & (enqr(top(deqd(c_xst))) < enqt(c_vi2)))
     | (deqd(c_xst) = new)
    -> true
      became equation thm1.20.4:
      ((deqr(top(deqd(c_xst))) < c_vi1) & (enqr(top(deqd(c_xst))) < enqt(c_y)))
       | (deqd(c_xst) = new)
      == true
    least(c_vi2, enqd(c_xst)) -> true
      became equation thm1.20.3:
      least(c_y, enqd(c_xst)) == true
    in(c_vi2, enqd(c_xst)) -> true
      became equation thm1.20.2:
      in(c_y, enqd(c_xst)) == true
    enqt(c_vi2) < c_vi1 -> true
      became equation thm1.20.1:
      enqt(c_y) < c_vi1 == true
    c_vi2 = c_y -> false
      became equation Case.19.2.2:
      c_y = c_y == false
    element(c_vi2) -> element(c_y)
      became identity Case.21.1.2:
      element(c_y) == element(c_y)

Ordered equation Case.16.2.2 into the rewrite rule:
  in_stack(trip(element(c_y), enqt(c_y), c_vi1), deqd(c_xst)) -> false

Ordered equation thm1.20.4 into the rewrite rule:
  ((deqr(top(deqd(c_xst))) < c_vi1) & (enqr(top(deqd(c_xst))) < enqt(c_y)))
   | (deqd(c_xst) = new)
  -> true

Ordered equation thm1.20.3 into the rewrite rule:
  least(c_y, enqd(c_xst)) -> true

Ordered equation thm1.20.1 into the rewrite rule:
  enqt(c_y) < c_vi1 -> true

Equation Case.19.2.2
    true == false
is inconsistent.

Lemma thm1.1.2.4.1.1.2.1 in the proof by cases of Lemma thm1.1.2.4.1.1.2
```

```
    Invl(deq(c_xst, c_vil, c_vi2), c_x, c_y) -> true
    Case.21.1: element(c_vi2) = element(c_y)
[] Proved by impossible case.

Case.21.2
    not(element(c_vi2) = element(c_y)) == true
involves proving Lemma thml.1.2.4.1.1.2.2
    Invl(deq(c_xst, c_vil, c_vi2), c_x, c_y) -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.21.2:
  (element(c_vi2) = element(c_y)) <=> false == true
to yield the following equations:
  Case.21.2.1: element(c_vi2) = element(c_y) == false

Ordered equation Case.21.2.1 into the rewrite rule:
  element(c_vi2) = element(c_y) -> false

The case system now contains 1 rewrite rule.

Lemma thml.1.2.4.1.1.2.2 in the proof by cases of Lemma thml.1.2.4.1.1.2
    Invl(deq(c_xst, c_vil, c_vi2), c_x, c_y) -> true
    Case.21.2: not(element(c_vi2) = element(c_y))
[] Proved by rewriting (with unreduced rules).

Lemma thml.1.2.4.1.1.2 in the proof by cases of Lemma thml.1.2.4.1.1
    Invl(deq(c_xst, c_vil, c_vi2), c_x, c_y) -> true
    Case.19.2: not(c_vi2 = c_y)
[] Proved by cases
    (element(c_vi2) = element(c_y)) | not(element(c_vi2) = element(c_y))

Lemma thml.1.2.4.1.1 in the proof by cases of Lemma thml.1.2.4.1
    Invl(deq(c_xst, c_vil, c_vi2), c_x, c_y) -> true
    Case.18.1: in(c_y, enqd(c_xst))
[] Proved by cases
    (c_vi2 = c_y) | not(c_vi2 = c_y)

Case.18.2
    not(in(c_y, enqd(c_xst))) == true
involves proving Lemma thml.1.2.4.1.2
    Invl(deq(c_xst, c_vil, c_vi2), c_x, c_y) -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.18.2:
  false <=> in(c_y, enqd(c_xst)) == true
to yield the following equations:
  Case.18.2.1: false == in(c_y, enqd(c_xst))

Ordered equation Case.18.2.1 into the rewrite rule:
  in(c_y, enqd(c_xst)) -> false

The case system now contains 1 rewrite rule.

Lemma thml.1.2.4.1.2 in the proof by cases of Lemma thml.1.2.4.1
    Invl(deq(c_xst, c_vil, c_vi2), c_x, c_y) -> true
    Case.18.2: not(in(c_y, enqd(c_xst)))
[] Proved by rewriting (with unreduced rules).

Lemma thml.1.2.4.1 in the proof by cases of Lemma thml.1.2.4
    Invl(deq(c_xst, c_vil, c_vi2), c_x, y) -> true
    Case.17.1: c_x = trip(element(c_vi2), enqt(c_vi2), c_vil)
```

```
[] Proved by cases
    in(y, enqd(c_xst)) | not(in(y, enqd(c_xst)))

Case.17.2
    not(c_x = trip(element(c_vi2), enqt(c_vi2), c_vi1)) == true
involves proving Lemma thm1.1.2.4.2
    Invl(deq(c_xst, c_vi1, c_vi2), c_x, y) -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.17.2:
  (c_x = trip(element(c_vi2), enqt(c_vi2), c_vi1)) <=> false == true
to yield the following equations:
  Case.17.2.1: c_x = trip(element(c_vi2), enqt(c_vi2), c_vi1) == false

Ordered equation Case.17.2.1 into the rewrite rule:
  c_x = trip(element(c_vi2), enqt(c_vi2), c_vi1) -> false

The case system now contains 1 rewrite rule.

Lemma thm1.1.2.4.2 in the proof by cases of Lemma thm1.1.2.4
    Invl(deq(c_xst, c_vi1, c_vi2), c_x, y) -> true
    Case.17.2: not(c_x = trip(element(c_vi2), enqt(c_vi2), c_vi1))
[] Proved by rewriting (with unreduced rules).

Lemma thm1.1.2.4 in the proof by cases of Lemma thm1.1.2
    Invl(deq(c_xst, vi1, vi2), c_x, y) -> true
    Case.16.2: not(in_stack(c_x, deqd(c_xst)))
[] Proved by cases
     (c_x = trip(element(vi2), enqt(vi2), vi1))
      | not(c_x = trip(element(vi2), enqt(vi2), vi1))


Lemma thm1.1.2 for the induction step in the proof of Conjecture thm1.1
    Invl(deq(c_xst, vi1, vi2), x, y) -> true
[] Proved by cases
    in_stack(x, deqd(c_xst)) | not(in_stack(x, deqd(c_xst)))

Conjecture thm1.1
    Invl(xst, x, y) -> true
[] Proved by induction over 'xst::St' of sort 'St'.

The system now contains 1 equation, 78 rewrite rules, and 9 deduction rules.

Ordered equation thm1.1 into the rewrite rule:
  ((element(y) = what(x)) <=> false)
   | (false <=> in(y, enqd(xst)))
   | (false <=> in_stack(x, deqd(xst)))
  -> true

The system now contains 79 rewrite rules and 9 deduction rules.

-> q
```

4.3. LP Proof Session of Invariant 2

```
-> set axiom o

The axiom use is now 'order-equations-into-rules'.

-> thaw Inv

System thawed from 'Inv.frz'.

-> set name thm2

The name prefix is now 'thm2'.

-> prove Inv2(xst,x,y) by induction xst St

The basis step in an inductive proof of Conjecture thm2.1
    Inv2(xst, x, y) -> true
involves proving the following lemma(s):

thm2.1.1: Inv2(init, x, y) -> true
          [] Proved by normalization

The induction step in an inductive proof of Conjecture thm2.1
    Inv2(xst, x, y) -> true
uses the following equation(s) for the induction hypothesis:

Induct.2: Inv2(c_xst, x, y) -> true

The system now contains 1 equation, 67 rewrite rules, and 5 deduction rules.

Ordered equation Induct.2 into the rewrite rule:
  ((deqr(x) < deqr(y)) & (enqr(x) < enqr(y)))
   | not(deq_before(x, y, deqd(c_xst)))
  -> true

The system now contains 68 rewrite rules and 5 deduction rules.

The induction step involves proving the following lemma(s):

thm2.1.2: Inv2(deq(c_xst, vi1, vi2), x, y) -> true
              which reduces to the equation
              ((deqr(x) < deqr(y)) & (enqr(x) < enqr(y)))
               | not(((trip(element(vi2), enqt(vi2), vi1) = y)
                      & in_stack(x, deqd(c_xst)))
                     | deq_before(x, y, deqd(c_xst)))


              -> true
thm2.1.3: Inv2(enq(c_xst, vi1, vi2), x, y) -> true
          [] Proved by normalization
thm2.1.4: Inv2(commit(c_xst, vi1), x, y) -> true
          [] Proved by normalization
thm2.1.5: Inv2(abort(c_xst, vi1), x, y) -> true
              which reduces to the equation
              ((deqr(x) < deqr(y)) & (enqr(x) < enqr(y)))
               | not(deq_before(x, y, deqd(abort(c_xst, vi1))))
              -> true

Proof of Lemma thm2.1.5 suspended.

-> resume by case deq_before(x,y,deqd(abort(c_xst,vi1)))

Case.3.1
    deq_before(c_x, c_y, deqd(abort(c_xst, c_vi1))) == true
involves proving Lemma thm2.1.5.1
    Inv2(abort(c_xst, c_vi1), c_x, c_y) -> true
```

```
The case system now contains 1 equation.

Ordered equation Case.3.1 into the rewrite rule:
  deq_before(c_x, c_y, deqd(abort(c_xst, c_vil))) -> true

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 68 rewrite rules, and 5 deduction rules.

Ordered equation Case.3.1 into the rewrite rule:
  deq_before(c_x, c_y, deqd(abort(c_xst, c_vil))) -> true

The system now contains 69 rewrite rules and 5 deduction rules.

Lemma thm2.1.5.1 in the proof by cases of Lemma thm2.1.5
    Inv2(abort(c_xst, c_vil), c_x, c_y) -> true
    Case.3.1: deq_before(c_x, c_y, deqd(abort(c_xst, c_vil)))
is NOT provable using the current partially completed system.  It reduces to
the equation
    (deqr(c_x) < deqr(c_y)) & (enqr(c_x) < enqr(c_y)) -> true

Proof of Lemma thm2.1.5.1 suspended.

-> crit case with State

Critical pairs between rule Case.3.1:
  deq_before(c_x, c_y, deqd(abort(c_xst, c_vil))) -> true
and rule State.13:
  deq_before(x, y, deqd(xst)) | not(deq_before(x, y, deqd(abort(xst, xt))))
  -> true
  are as follows:
    deq_before(c_x, c_y, deqd(c_xst)) == true

The system now contains 1 equation, 69 rewrite rules, and 5 deduction rules.

Ordered equation thm2.2 into the rewrite rule:
  deq_before(c_x, c_y, deqd(c_xst)) -> true

The system now contains 70 rewrite rules and 5 deduction rules.

Computed 2 new critical pairs, 1 of which reduced to an identity.  Added 1 of
them to the system.

-> crit thm2 with Induct

Critical pairs between rule thm2.2:
  deq_before(c_x, c_y, deqd(c_xst)) -> true
and rule Induct.2:
  ((deqr(x) < deqr(y)) & (enqr(x) < enqr(y)))
   | not(deq_before(x, y, deqd(c_xst)))
  -> true
  are as follows:
    (deqr(c_x) < deqr(c_y)) & (enqr(c_x) < enqr(c_y)) == true

The system now contains 1 equation, 70 rewrite rules, and 5 deduction rules.

Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation thm2.3:
  (deqr(c_x) < deqr(c_y)) & (enqr(c_x) < enqr(c_y)) == true
to yield the following equations:
  thm2.3.1: deqr(c_x) < deqr(c_y) == true
  thm2.3.2: enqr(c_x) < enqr(c_y) == true

Ordered equation thm2.3.2 into the rewrite rule:
  enqr(c_x) < enqr(c_y) -> true
```

```
Ordered equation thm2.3.1 into the rewrite rule:
  deqr(c_x) < deqr(c_y) -> true

The system now contains 72 rewrite rules and 5 deduction rules.

Lemma thm2.1.5.1 in the proof by cases of Lemma thm2.1.5
    Inv2(abort(c_xst, c_vi1), c_x, c_y) -> true
    Case.3.1: deq_before(c_x, c_y, deqd(abort(c_xst, c_vi1)))
[] Proved by rewriting.

Case.3.2
    not(deq_before(c_x, c_y, deqd(abort(c_xst, c_vi1)))) == true
involves proving Lemma thm2.1.5.2
    Inv2(abort(c_xst, c_vi1), c_x, c_y) -> true

The case system now contains 1 equation.

Deduction rule boolean.1:
  when  not(x) == true
  yield x == false
has been applied to equation Case.3.2:
  not(deq_before(c_x, c_y, deqd(abort(c_xst, c_vi1)))) == true
to yield the following equations:
  Case.3.2.1: deq_before(c_x, c_y, deqd(abort(c_xst, c_vi1))) == false

Ordered equation Case.3.2 into the rewrite rule:
  not(deq_before(c_x, c_y, deqd(abort(c_xst, c_vi1)))) -> true

Ordered equation Case.3.2.1 into the rewrite rule:
  deq_before(c_x, c_y, deqd(abort(c_xst, c_vi1))) -> false

    Left-hand side reduced:
    not(deq_before(c_x, c_y, deqd(abort(c_xst, c_vi1)))) -> true
      became equation Case.3.2:
      not(false) == true

Ordered equation Case.3.2 into the rewrite rule:
  not(false) -> true

The case system now contains 2 rewrite rules.

Lemma thm2.1.5.2 in the proof by cases of Lemma thm2.1.5
    Inv2(abort(c_xst, c_vi1), c_x, c_y) -> true
    Case.3.2: not(deq_before(c_x, c_y, deqd(abort(c_xst, c_vi1))))
[] Proved by rewriting (with unreduced rules).

Lemma thm2.1.5 for the induction step in the proof of Conjecture thm2.1
    Inv2(abort(c_xst, vi1), x, y) -> true
[] Proved by cases
    deq_before(x, y, deqd(abort(c_xst, vi1)))
     | not(deq_before(x, y, deqd(abort(c_xst, vi1))))


Lemma thm2.1.2 for the induction step in the proof of Conjecture thm2.1
    Inv2(deq(c_xst, vi1, vi2), x, y) -> true
is NOT provable using the current partially completed system.  It reduces to
the equation
    ((deqr(x) < deqr(y)) & (enqr(x) < enqr(y)))
     | not(((trip(element(vi2), enqt(vi2), vi1) = y)
            & in_stack(x, deqd(c_xst)))
           | deq_before(x, y, deqd(c_xst)))


    -> true

Proof of Lemma thm2.1.2 suspended.

Critical-pair computation abandoned because a theorem has been proved.
```

```
Computed 1 new critical pair.  Added 1 of them to the system.

-> resume by case deq_before(x,y,deqd(c_xst))

Case.4.1
    deq_before(c_x, c_y, deqd(c_xst)) == true
involves proving Lemma thm2.1.2.1
    Inv2(deq(c_xst, vi1, vi2), c_x, c_y) -> true

The case system now contains 1 equation.

Ordered equation Case.4.1 into the rewrite rule:
  deq_before(c_x, c_y, deqd(c_xst)) -> true

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 68 rewrite rules, and 5 deduction rules.

Ordered equation Case.4.1 into the rewrite rule:
  deq_before(c_x, c_y, deqd(c_xst)) -> true

The system now contains 69 rewrite rules and 5 deduction rules.

Lemma thm2.1.2.1 in the proof by cases of Lemma thm2.1.2
    Inv2(deq(c_xst, vi1, vi2), c_x, c_y) -> true
    Case.4.1: deq_before(c_x, c_y, deqd(c_xst))
is NOT provable using the current partially completed system.  It reduces to
the equation
    (deqr(c_x) < deqr(c_y)) & (enqr(c_x) < enqr(c_y)) -> true

Proof of Lemma thm2.1.2.1 suspended.

-> crit case with Induct

Critical pairs between rule Case.4.1:
  deq_before(c_x, c_y, deqd(c_xst)) -> true
and rule Induct.2:
  ((deqr(x) < deqr(y)) & (enqr(x) < enqr(y)))
   | not(deq_before(x, y, deqd(c_xst)))
  -> true
  are as follows:
    (deqr(c_x) < deqr(c_y)) & (enqr(c_x) < enqr(c_y)) == true

The system now contains 1 equation, 69 rewrite rules, and 5 deduction rules.

Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation thm2.4:
  (deqr(c_x) < deqr(c_y)) & (enqr(c_x) < enqr(c_y)) == true
to yield the following equations:
  thm2.4.1: deqr(c_x) < deqr(c_y) == true
  thm2.4.2: enqr(c_x) < enqr(c_y) == true

Ordered equation thm2.4.2 into the rewrite rule:
  enqr(c_x) < enqr(c_y) -> true

Ordered equation thm2.4.1 into the rewrite rule:
  deqr(c_x) < deqr(c_y) -> true

The system now contains 71 rewrite rules and 5 deduction rules.

Lemma thm2.1.2.1 in the proof by cases of Lemma thm2.1.2
    Inv2(deq(c_xst, vi1, vi2), c_x, c_y) -> true
    Case.4.1: deq_before(c_x, c_y, deqd(c_xst))
[] Proved by rewriting.

Case.4.2
```

```
    not(deq_before(c_x, c_y, deqd(c_xst))) == true
involves proving Lemma thm2.1.2.2
    Inv2(deq(c_xst, vi1, vi2), c_x, c_y) -> true

The case system now contains 1 equation.

Deduction rule boolean.1:
  when  not(x) == true
  yield x == false
has been applied to equation Case.4.2:
  not(deq_before(c_x, c_y, deqd(c_xst))) == true
to yield the following equations:
  Case.4.2.1: deq_before(c_x, c_y, deqd(c_xst)) == false

Ordered equation Case.4.2 into the rewrite rule:
  not(deq_before(c_x, c_y, deqd(c_xst))) -> true

Ordered equation Case.4.2.1 into the rewrite rule:
  deq_before(c_x, c_y, deqd(c_xst)) -> false

    Left-hand side reduced:
    not(deq_before(c_x, c_y, deqd(c_xst))) -> true
      became equation Case.4.2:
      not(false) == true

Ordered equation Case.4.2 into the rewrite rule:
  not(false) -> true

The case system now contains 2 rewrite rules.

The system now contains 1 equation, 68 rewrite rules, and 5 deduction rules.

Deduction rule boolean.1:
  when  not(x) == true
  yield x == false
has been applied to equation Case.4.2:
  not(deq_before(c_x, c_y, deqd(c_xst))) == true
to yield the following equations:
  Case.4.2.3: deq_before(c_x, c_y, deqd(c_xst)) == false

Ordered equation Case.4.2 into the rewrite rule:
  not(deq_before(c_x, c_y, deqd(c_xst))) -> true

Ordered equation Case.4.2.3 into the rewrite rule:
  deq_before(c_x, c_y, deqd(c_xst)) -> false

    Left-hand side reduced:
    not(deq_before(c_x, c_y, deqd(c_xst))) -> true
      became equation Case.4.2:
      not(false) == true

The system now contains 69 rewrite rules and 5 deduction rules.

Lemma thm2.1.2.2 in the proof by cases of Lemma thm2.1.2
    Inv2(deq(c_xst, vi1, vi2), c_x, c_y) -> true
    Case.4.2: not(deq_before(c_x, c_y, deqd(c_xst)))
is NOT provable using the current partially completed system.  It reduces to
the equation
     ((deqr(c_x) < deqr(c_y)) & (enqr(c_x) < enqr(c_y)))
      | not(c_y = trip(element(vi2), enqt(vi2), vi1))
      | not(in_stack(c_x, deqd(c_xst)))
     -> true

Proof of Lemma thm2.1.2.2 suspended.

Critical-pair computation abandoned because a theorem has been proved.

Computed 1 new critical pair.  Added 1 of them to the system.
```

```
-> resume by case c_y=trip(element(vi2::enq_rec),enqt(vi2::enq_rec),vi1)

Case.5.1
    c_y = trip(element(c_vi2), enqt(c_vi2), c_vi1) == true
involves proving Lemma thm2.1.2.2.1
    Inv2(deq(c_xst, c_vi1, c_vi2), c_x, c_y) -> true

The case system now contains 1 equation.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.5.1:
  c_y = trip(element(c_vi2), enqt(c_vi2), c_vi1) == true
to yield the following equations:
  Case.5.1.1: c_y == trip(element(c_vi2), enqt(c_vi2), c_vi1)

Ordered equation Case.5.1.1 into the rewrite rule:
  c_y -> trip(element(c_vi2), enqt(c_vi2), c_vi1)

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 69 rewrite rules, and 5 deduction rules.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.5.1:
  c_y = trip(element(c_vi2), enqt(c_vi2), c_vi1) == true
to yield the following equations:
  Case.5.1.2: c_y == trip(element(c_vi2), enqt(c_vi2), c_vi1)

Ordered equation Case.5.1.2 into the rewrite rule:
  c_y -> trip(element(c_vi2), enqt(c_vi2), c_vi1)

    Left-hand side reduced:
    deq_before(c_x, c_y, deqd(c_xst)) -> false
      became equation Case.4.2.3:
      deq_before(c_x, trip(element(c_vi2), enqt(c_vi2), c_vi1), deqd(c_xst))
      == false

Ordered equation Case.4.2.3 into the rewrite rule:
  deq_before(c_x, trip(element(c_vi2), enqt(c_vi2), c_vi1), deqd(c_xst))
  -> false

The system now contains 70 rewrite rules and 5 deduction rules.

Lemma thm2.1.2.2.1 in the proof by cases of Lemma thm2.1.2.2
    Inv2(deq(c_xst, c_vi1, c_vi2), c_x, c_y) -> true
    Case.5.1: c_y = trip(element(c_vi2), enqt(c_vi2), c_vi1)
is NOT provable using the current partially completed system.  It reduces to
the equation
    ((deqr(c_x) < c_vi1) & (enqr(c_x) < enqt(c_vi2)))
     | not(in_stack(c_x, deqd(c_xst)))
    -> true

Proof of Lemma thm2.1.2.2.1 suspended.

-> add when_deq(c_xst,c_x,c_vi1,c_vi2)

Added 1 equation to the system.

Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation thm2.5:
   (enqt(c_vi2) < c_vi1)
    & in(c_vi2, enqd(c_xst))
```

```
   & least(c_vi2, enqd(c_xst))
   & (((deqr(top(deqd(c_xst))) < c_vi1)
        & (enqr(top(deqd(c_xst))) < enqt(c_vi2)))
       | (deqd(c_xst) = new))

   & (not(element(c_vi2) = what(c_x)) | not(in_stack(c_x, deqd(c_xst))))
  -> true
to yield the following equations:
  thm2.5.1: enqt(c_vi2) < c_vi1 == true
  thm2.5.2: in(c_vi2, enqd(c_xst)) == true
  thm2.5.3: least(c_vi2, enqd(c_xst)) == true
  thm2.5.4: ((deqr(top(deqd(c_xst))) < c_vi1)
              & (enqr(top(deqd(c_xst))) < enqt(c_vi2)))
             | (deqd(c_xst) = new)
            == true
  thm2.5.5: not(element(c_vi2) = what(c_x)) | not(in_stack(c_x, deqd(c_xst)))
            == true

Ordered equation thm2.5.5 into the rewrite rule:
  not(element(c_vi2) = what(c_x)) | not(in_stack(c_x, deqd(c_xst))) -> true

Ordered equation thm2.5.4 into the rewrite rule:
  ((deqr(top(deqd(c_xst))) < c_vi1) & (enqr(top(deqd(c_xst))) < enqt(c_vi2)))
   | (deqd(c_xst) = new)
  -> true

Ordered equation thm2.5.3 into the rewrite rule:
  least(c_vi2, enqd(c_xst)) -> true

Ordered equation thm2.5.2 into the rewrite rule:
  in(c_vi2, enqd(c_xst)) -> true

Ordered equation thm2.5.1 into the rewrite rule:
  enqt(c_vi2) < c_vi1 -> true

The system now contains 75 rewrite rules and 5 deduction rules.

-> resume by case deqd(c_xst)=new

Case.6.1
    deqd(c_xst) = new == true
involves proving Lemma thm2.1.2.2.1.1
    Inv2(deq(c_xst, c_vi1, c_vi2), c_x, c_y) -> true

The case system now contains 1 equation.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.6.1:
  deqd(c_xst) = new == true
to yield the following equations:
  Case.6.1.1: deqd(c_xst) == new

Ordered equation Case.6.1.1 into the rewrite rule:
  deqd(c_xst) -> new

The case system now contains 1 rewrite rule.

Lemma thm2.1.2.2.1.1 in the proof by cases of Lemma thm2.1.2.2.1
    Inv2(deq(c_xst, c_vi1, c_vi2), c_x, c_y) -> true
    Case.6.1: deqd(c_xst) = new
[] Proved by rewriting (with unreduced rules).

Case.6.2
    not(deqd(c_xst) = new) == true
involves proving Lemma thm2.1.2.2.1.2
    Inv2(deq(c_xst, c_vi1, c_vi2), c_x, c_y) -> true
```

```
The case system now contains 1 equation.

Deduction rule boolean.1:
  when  not(x) == true
  yield x == false
has been applied to equation Case.6.2:
  not(deqd(c_xst) = new) == true
to yield the following equations:
  Case.6.2.1: deqd(c_xst) = new == false

Ordered equation Case.6.2 into the rewrite rule:
  not(deqd(c_xst) = new) -> true

Ordered equation Case.6.2.1 into the rewrite rule:
  deqd(c_xst) = new -> false

    Left-hand side reduced:
    not(deqd(c_xst) = new) -> true
      became equation Case.6.2:
      not(false) == true

Ordered equation Case.6.2 into the rewrite rule:
  not(false) -> true

The case system now contains 2 rewrite rules.

The system now contains 1 equation, 75 rewrite rules, and 5 deduction rules.

Deduction rule boolean.1:
  when  not(x) == true
  yield x == false
has been applied to equation Case.6.2:
  not(deqd(c_xst) = new) == true
to yield the following equations:
  Case.6.2.3: deqd(c_xst) = new == false

Ordered equation Case.6.2 into the rewrite rule:
  not(deqd(c_xst) = new) -> true

Ordered equation Case.6.2.3 into the rewrite rule:
  deqd(c_xst) = new -> false

    Following 2 left-hand sides reduced:
    ((deqr(top(deqd(c_xst))) < c_vi1) & (enqr(top(deqd(c_xst))) < enqt(c_vi2)))
     | (deqd(c_xst) = new)
    -> true
      became equation thm2.5.4:
      ((deqr(top(deqd(c_xst))) < c_vi1)
        & (enqr(top(deqd(c_xst))) < enqt(c_vi2)))
       | false
      == true
    not(deqd(c_xst) = new) -> true
      became equation Case.6.2:
      not(false) == true

Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation thm2.5.4:
  (deqr(top(deqd(c_xst))) < c_vi1) & (enqr(top(deqd(c_xst))) < enqt(c_vi2))
  == true
to yield the following equations:
  thm2.5.4.1: deqr(top(deqd(c_xst))) < c_vi1 == true
  thm2.5.4.2: enqr(top(deqd(c_xst))) < enqt(c_vi2) == true

Ordered equation thm2.5.4.2 into the rewrite rule:
  enqr(top(deqd(c_xst))) < enqt(c_vi2) -> true
```

```
Ordered equation thm2.5.4.1 into the rewrite rule:
  deqr(top(deqd(c_xst))) < c_vil -> true

The system now contains 77 rewrite rules and 5 deduction rules.

Lemma thm2.1.2.2.1.2 in the proof by cases of Lemma thm2.1.2.2.1
    Inv2(deq(c_xst, c_vil, c_vi2), c_x, c_y) -> true
    Case.6.2: not(deqd(c_xst) = new)
is NOT provable using the current partially completed system.  It reduces to
the equation
    ((deqr(c_x) < c_vil) & (enqr(c_x) < enqt(c_vi2)))
     | not(in_stack(c_x, deqd(c_xst)))
    -> true

Proof of Lemma thm2.1.2.2.1.2 suspended.

-> resume by case in_stack(c_x,deqd(c_xst))

Case.7.1
    in_stack(c_x, deqd(c_xst)) == true
involves proving Lemma thm2.1.2.2.1.2.1
    Inv2(deq(c_xst, c_vil, c_vi2), c_x, c_y) -> true

The case system now contains 1 equation.

Ordered equation Case.7.1 into the rewrite rule:
  in_stack(c_x, deqd(c_xst)) -> true

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 77 rewrite rules, and 5 deduction rules.

Ordered equation Case.7.1 into the rewrite rule:
  in_stack(c_x, deqd(c_xst)) -> true

    Left-hand side reduced:
    not(element(c_vi2) = what(c_x)) | not(in_stack(c_x, deqd(c_xst))) -> true
      became equation thm2.5.5:
      not(element(c_vi2) = what(c_x)) | not(true) == true

Deduction rule boolean.1:
  when  not(x) == true
  yield x == false
has been applied to equation thm2.5.5:
  not(element(c_vi2) = what(c_x)) == true
to yield the following equations:
  thm2.5.5.1: element(c_vi2) = what(c_x) == false

Ordered equation thm2.5.5 into the rewrite rule:
  not(element(c_vi2) = what(c_x)) -> true

Ordered equation thm2.5.5.1 into the rewrite rule:
  element(c_vi2) = what(c_x) -> false

    Left-hand side reduced:
    not(element(c_vi2) = what(c_x)) -> true
      became equation thm2.5.5:
      not(false) == true

The system now contains 78 rewrite rules and 5 deduction rules.

Lemma thm2.1.2.2.1.2.1 in the proof by cases of Lemma thm2.1.2.2.1.2
    Inv2(deq(c_xst, c_vil, c_vi2), c_x, c_y) -> true
    Case.7.1: in_stack(c_x, deqd(c_xst))
is NOT provable using the current partially completed system.  It reduces to
the equation
    (deqr(c_x) < c_vil) & (enqr(c_x) < enqt(c_vi2)) -> true

Proof of Lemma thm2.1.2.2.1.2.1 suspended.
```

```
-> crit case with Induct

Computed 1 new critical pair, which reduced to an identity.  Added 0 of them to
the system.

-> resume by case c_x=top(deqd(c_xst))

Case.8.1
    c_x = top(deqd(c_xst)) == true
involves proving Lemma thm2.1.2.2.1.2.1.1
    Inv2(deq(c_xst, c_vi1, c_vi2), c_x, c_y) -> true

The case system now contains 1 equation.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.8.1:
  c_x = top(deqd(c_xst)) == true
to yield the following equations:
  Case.8.1.1: c_x == top(deqd(c_xst))

Ordered equation Case.8.1.1 into the rewrite rule:
  c_x -> top(deqd(c_xst))

The case system now contains 1 rewrite rule.

Lemma thm2.1.2.2.1.2.1.1 in the proof by cases of Lemma thm2.1.2.2.1.2.1
    Inv2(deq(c_xst, c_vi1, c_vi2), c_x, c_y) -> true
    Case.8.1: c_x = top(deqd(c_xst))
[] Proved by rewriting (with unreduced rules).

Case.8.2
    not(c_x = top(deqd(c_xst))) == true
involves proving Lemma thm2.1.2.2.1.2.1.2
    Inv2(deq(c_xst, c_vi1, c_vi2), c_x, c_y) -> true

The case system now contains 1 equation.

Deduction rule boolean.1:
  when  not(x) == true
  yield x == false
has been applied to equation Case.8.2:
  not(c_x = top(deqd(c_xst))) == true
to yield the following equations:
  Case.8.2.1: c_x = top(deqd(c_xst)) == false

Ordered equation Case.8.2 into the rewrite rule:
  not(c_x = top(deqd(c_xst))) -> true

Ordered equation Case.8.2.1 into the rewrite rule:
  c_x = top(deqd(c_xst)) -> false

    Left-hand side reduced:
    not(c_x = top(deqd(c_xst))) -> true
      became equation Case.8.2:
      not(false) == true

Ordered equation Case.8.2 into the rewrite rule:
  not(false) -> true

The case system now contains 2 rewrite rules.

The system now contains 1 equation, 78 rewrite rules, and 5 deduction rules.

Deduction rule boolean.1:
  when  not(x) == true
  yield x == false
```

```
has been applied to equation Case.8.2:
  not(c_x = top(deqd(c_xst))) == true
to yield the following equations:
  Case.8.2.3: c_x = top(deqd(c_xst)) == false

Ordered equation Case.8.2 into the rewrite rule:
  not(c_x = top(deqd(c_xst))) -> true

Ordered equation Case.8.2.3 into the rewrite rule:
  c_x = top(deqd(c_xst)) -> false

    Left-hand side reduced:
    not(c_x = top(deqd(c_xst))) -> true
      became equation Case.8.2:
      not(false) == true

The system now contains 79 rewrite rules and 5 deduction rules.

Lemma thm2.1.2.2.1.2.1.2 in the proof by cases of Lemma thm2.1.2.2.1.2.1
    Inv2(deq(c_xst, c_vil, c_vi2), c_x, c_y) -> true
    Case.8.2: not(c_x = top(deqd(c_xst)))
is NOT provable using the current partially completed system.  It reduces to
the equation
    (deqr(c_x) < c_vil) & (enqr(c_x) < enqt(c_vi2)) -> true

Proof of Lemma thm2.1.2.2.1.2.1.2 suspended.

-> crit case with lemma

Critical pairs between rule Case.7.1:
  in_stack(c_x, deqd(c_xst)) -> true
and rule lemma.3:
  (top(y) = x) | deq_before(x, top(y), y) | not(in_stack(x, y)) -> true
  are as follows:
    deq_before(c_x, top(deqd(c_xst)), deqd(c_xst)) == true

The system now contains 1 equation, 79 rewrite rules, and 5 deduction rules.

Ordered equation thm2.6 into the rewrite rule:
  deq_before(c_x, top(deqd(c_xst)), deqd(c_xst)) -> true

The system now contains 80 rewrite rules and 5 deduction rules.

Computed 2 new critical pairs, 1 of which reduced to an identity.  Added 1 of
them to the system.


-> prove  (deqr(c_x)<deqr(top(deqd(c_xst))))&(enqr(c_x)<enqr(top(deqd(c_xst))))

Conjecture thm2.7
    (deqr(c_x) < deqr(top(deqd(c_xst)))) & (enqr(c_x) < enqr(top(deqd(c_xst))))
    -> true
is NOT provable using the current partially completed system.

Proof of Conjecture thm2.7 suspended.

-> crit thm2 with Induct

Critical pairs between rule thm2.6:
  deq_before(c_x, top(deqd(c_xst)), deqd(c_xst)) -> true
and rule Induct.2:
  ((deqr(x) < deqr(y)) & (enqr(x) < enqr(y)))
   | not(deq_before(x, y, deqd(c_xst)))
  -> true
  are as follows:
    (deqr(c_x) < deqr(top(deqd(c_xst)))) & (enqr(c_x) < enqr(top(deqd(c_xst))))
    == true

The system now contains 1 equation, 80 rewrite rules, and 5 deduction rules.
```

```
Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation thm2.8:
  (deqr(c_x) < deqr(top(deqd(c_xst)))) & (enqr(c_x) < enqr(top(deqd(c_xst))))
  == true
to yield the following equations:
  thm2.8.1: deqr(c_x) < deqr(top(deqd(c_xst))) == true
  thm2.8.2: enqr(c_x) < enqr(top(deqd(c_xst))) == true

Ordered equation thm2.8.2 into the rewrite rule:
  enqr(c_x) < enqr(top(deqd(c_xst))) -> true

Ordered equation thm2.8.1 into the rewrite rule:
  deqr(c_x) < deqr(top(deqd(c_xst))) -> true

The system now contains 82 rewrite rules and 5 deduction rules.

Conjecture thm2.7
    (deqr(c_x) < deqr(top(deqd(c_xst)))) & (enqr(c_x) < enqr(top(deqd(c_xst))))
    -> true
[] Proved by rewriting.

Lemma thm2.1.2.2.1.2.1.2 in the proof by cases of Lemma thm2.1.2.2.1.2.1
    Inv2(deq(c_xst, c_vi1, c_vi2), c_x, c_y) -> true
    Case.8.2: not(c_x = top(deqd(c_xst)))
is NOT provable using the current partially completed system.  It reduces to
the equation
    (deqr(c_x) < c_vi1) & (enqr(c_x) < enqt(c_vi2)) -> true

Proof of Lemma thm2.1.2.2.1.2.1.2 suspended.

Critical-pair computation abandoned because a theorem has been proved.

Computed 1 new critical pair.  Added 1 of them to the system.


-> instantiate xt by deqr(c_x),xt1 by deqr(top(deqd(c_xst))),xt2 by c_vi1 in TransID.1


Equation TransID.1:
  (xt < xt2) | not(xt < xt1) | not(xt1 < xt2) -> true
has been instantiated to equation TransID.1.1:
  deqr(c_x) < c_vi1 -> true

Added 1 equation to the system.

Ordered equation TransID.1.1 into the rewrite rule:
  deqr(c_x) < c_vi1 -> true

The system now contains 83 rewrite rules and 5 deduction rules.

Lemma thm2.1.2.2.1.2.1.2 in the proof by cases of Lemma thm2.1.2.2.1.2.1
    Inv2(deq(c_xst, c_vi1, c_vi2), c_x, c_y) -> true
    Case.8.2: not(c_x = top(deqd(c_xst)))
is NOT provable using the current partially completed system.  It reduces to
the equation
    enqr(c_x) < enqt(c_vi2) -> true

Proof of Lemma thm2.1.2.2.1.2.1.2 suspended.

-> instantiate xt by enqr(c_x),xt1 by enqr(top(deqd(c_xst))),xt2 by enqt(c_vi2) in TransID.1


Equation TransID.1:
  (xt < xt2) | not(xt < xt1) | not(xt1 < xt2) -> true
has been instantiated to equation TransID.1.2:
```

```
  enqr(c_x) < enqt(c_vi2) -> true

Added 1 equation to the system.

Ordered equation TransID.1.2 into the rewrite rule:
  enqr(c_x) < enqt(c_vi2) -> true

The system now contains 84 rewrite rules and 5 deduction rules.

Lemma thm2.1.2.2.1.2.1.2 in the proof by cases of Lemma thm2.1.2.2.1.2.1
    Inv2(deq(c_xst, c_vi1, c_vi2), c_x, c_y) -> true
    Case.8.2: not(c_x = top(deqd(c_xst)))
[] Proved by rewriting.

Lemma thm2.1.2.2.1.2.1 in the proof by cases of Lemma thm2.1.2.2.1.2
    Inv2(deq(c_xst, c_vi1, c_vi2), c_x, c_y) -> true
    Case.7.1: in_stack(c_x, deqd(c_xst))
[] Proved by cases
    (c_x = top(deqd(c_xst))) | not(c_x = top(deqd(c_xst)))

Case.7.2
    not(in_stack(c_x, deqd(c_xst))) == true
involves proving Lemma thm2.1.2.2.1.2.2
    Inv2(deq(c_xst, c_vi1, c_vi2), c_x, c_y) -> true

The case system now contains 1 equation.

Deduction rule boolean.1:
  when  not(x) == true
  yield x == false
has been applied to equation Case.7.2:
  not(in_stack(c_x, deqd(c_xst))) == true
to yield the following equations:
  Case.7.2.1: in_stack(c_x, deqd(c_xst)) == false

Ordered equation Case.7.2 into the rewrite rule:
  not(in_stack(c_x, deqd(c_xst))) -> true

Ordered equation Case.7.2.1 into the rewrite rule:
  in_stack(c_x, deqd(c_xst)) -> false

    Left-hand side reduced:
    not(in_stack(c_x, deqd(c_xst))) -> true
      became equation Case.7.2:
      not(false) == true

Ordered equation Case.7.2 into the rewrite rule:
  not(false) -> true

The case system now contains 2 rewrite rules.

Lemma thm2.1.2.2.1.2.2 in the proof by cases of Lemma thm2.1.2.2.1.2
    Inv2(deq(c_xst, c_vi1, c_vi2), c_x, c_y) -> true
    Case.7.2: not(in_stack(c_x, deqd(c_xst)))
[] Proved by rewriting (with unreduced rules).

Lemma thm2.1.2.2.1.2 in the proof by cases of Lemma thm2.1.2.2.1
    Inv2(deq(c_xst, c_vi1, c_vi2), c_x, c_y) -> true
    Case.6.2: not(deqd(c_xst) = new)
[] Proved by cases
    in_stack(c_x, deqd(c_xst)) | not(in_stack(c_x, deqd(c_xst)))

Lemma thm2.1.2.2.1 in the proof by cases of Lemma thm2.1.2.2
    Inv2(deq(c_xst, c_vi1, c_vi2), c_x, c_y) -> true
    Case.5.1: c_y = trip(element(c_vi2), enqt(c_vi2), c_vi1)
[] Proved by cases
    (deqd(c_xst) = new) | not(deqd(c_xst) = new)

Case.5.2
```

```
    not(c_y = trip(element(c_vi2), enqt(c_vi2), c_vi1)) == true
involves proving Lemma thm2.1.2.2.2
    Inv2(deq(c_xst, c_vi1, c_vi2), c_x, c_y) -> true

The case system now contains 1 equation.

Deduction rule boolean.1:
  when  not(x) == true
  yield x == false
has been applied to equation Case.5.2:
  not(c_y = trip(element(c_vi2), enqt(c_vi2), c_vi1)) == true
to yield the following equations:
  Case.5.2.1: c_y = trip(element(c_vi2), enqt(c_vi2), c_vi1) == false

Ordered equation Case.5.2 into the rewrite rule:
  not(c_y = trip(element(c_vi2), enqt(c_vi2), c_vi1)) -> true

Ordered equation Case.5.2.1 into the rewrite rule:
  c_y = trip(element(c_vi2), enqt(c_vi2), c_vi1) -> false

    Left-hand side reduced:
    not(c_y = trip(element(c_vi2), enqt(c_vi2), c_vi1)) -> true
      became equation Case.5.2:
      not(false) == true

Ordered equation Case.5.2 into the rewrite rule:
  not(false) -> true

The case system now contains 2 rewrite rules.

Lemma thm2.1.2.2.2 in the proof by cases of Lemma thm2.1.2.2
    Inv2(deq(c_xst, c_vi1, c_vi2), c_x, c_y) -> true
    Case.5.2: not(c_y = trip(element(c_vi2), enqt(c_vi2), c_vi1))
[] Proved by rewriting (with unreduced rules).

Lemma thm2.1.2.2 in the proof by cases of Lemma thm2.1.2
    Inv2(deq(c_xst, vi1, vi2), c_x, c_y) -> true
    Case.4.2: not(deq_before(c_x, c_y, deqd(c_xst)))
[] Proved by cases
     (c_y = trip(element(vi2), enqt(vi2), vi1))
      | not(c_y = trip(element(vi2), enqt(vi2), vi1))


Lemma thm2.1.2 for the induction step in the proof of Conjecture thm2.1
    Inv2(deq(c_xst, vi1, vi2), x, y) -> true
[] Proved by cases
    deq_before(x, y, deqd(c_xst)) | not(deq_before(x, y, deqd(c_xst)))

Conjecture thm2.1
    Inv2(xst, x, y) -> true
[] Proved by induction over 'xst::St' of sort 'St'.

The system now contains 1 equation, 67 rewrite rules, and 5 deduction rules.

Ordered equation thm2.1 into the rewrite rule:
  ((deqr(x) < deqr(y)) & (enqr(x) < enqr(y)))
   | not(deq_before(x, y, deqd(xst)))
  -> true

The system now contains 68 rewrite rules and 5 deduction rules.

-> qed

All conjectures have been proved.

-> q
```

4.4. LP Proof Session of Invariant 3

```
-> set axiom o

The axiom use is now 'order-equations-into-rules'.

-> thaw Inv

System thawed from 'Inv.frz'.

-> set name thm3

The name prefix is now 'thm3'.

-> prove Inv3(xst,x) by induction xst St

The basis step in an inductive proof of Conjecture thm3.1
    Inv3(xst, x) -> true
involves proving the following lemma(s):

thm3.1.1: Inv3(init, x) -> true
          [] Proved by normalization

The induction step in an inductive proof of Conjecture thm3.1
    Inv3(xst, x) -> true
uses the following equation(s) for the induction hypothesis:

Induct.2: Inv3(c_xst, x) -> true

The system now contains 1 equation, 67 rewrite rules, and 5 deduction rules.

Ordered equation Induct.2 into the rewrite rule:
  (enqr(x) < deqr(x)) | not(in_stack(x, deqd(c_xst))) -> true

The system now contains 68 rewrite rules and 5 deduction rules.

The induction step involves proving the following lemma(s):

thm3.1.2: Inv3(deq(c_xst, vi1, vi2), x) -> true
              which reduces to the equation
              (enqr(x) < deqr(x))
               | not((trip(element(vi2), enqt(vi2), vi1) = x)
                     | in_stack(x, deqd(c_xst)))


              -> true
thm3.1.3: Inv3(enq(c_xst, vi1, vi2), x) -> true
          [] Proved by normalization
thm3.1.4: Inv3(commit(c_xst, vi1), x) -> true
          [] Proved by normalization
thm3.1.5: Inv3(abort(c_xst, vi1), x) -> true
          [] Proved by normalization

Proof of Lemma thm3.1.2 suspended.

-> resume by case in_stack(x,deqd(c_xst))

Case.3.1
    in_stack(c_x, deqd(c_xst)) == true
involves proving Lemma thm3.1.2.1
    Inv3(deq(c_xst, vi1, vi2), c_x) -> true

The case system now contains 1 equation.

Ordered equation Case.3.1 into the rewrite rule:
  in_stack(c_x, deqd(c_xst)) -> true

The case system now contains 1 rewrite rule.
```

```
The system now contains 1 equation, 68 rewrite rules, and 5 deduction rules.

Ordered equation Case.3.1 into the rewrite rule:
  in_stack(c_x, deqd(c_xst)) -> true

The system now contains 69 rewrite rules and 5 deduction rules.

Lemma thm3.1.2.1 in the proof by cases of Lemma thm3.1.2
    Inv3(deq(c_xst, vi1, vi2), c_x) -> true
    Case.3.1: in_stack(c_x, deqd(c_xst))
is NOT provable using the current partially completed system.  It reduces to
the equation
    enqr(c_x) < deqr(c_x) -> true

Proof of Lemma thm3.1.2.1 suspended.

-> crit case with Induct

Critical pairs between rule Case.3.1:
  in_stack(c_x, deqd(c_xst)) -> true
and rule Induct.2:
  (enqr(x) < deqr(x)) | not(in_stack(x, deqd(c_xst))) -> true
  are as follows:
    enqr(c_x) < deqr(c_x) == true

The system now contains 1 equation, 69 rewrite rules, and 5 deduction rules.

Ordered equation thm3.2 into the rewrite rule:
  enqr(c_x) < deqr(c_x) -> true

The system now contains 70 rewrite rules and 5 deduction rules.

Lemma thm3.1.2.1 in the proof by cases of Lemma thm3.1.2
    Inv3(deq(c_xst, vi1, vi2), c_x) -> true
    Case.3.1: in_stack(c_x, deqd(c_xst))
[] Proved by rewriting.

Case.3.2
    not(in_stack(c_x, deqd(c_xst))) == true
involves proving Lemma thm3.1.2.2
    Inv3(deq(c_xst, vi1, vi2), c_x) -> true

The case system now contains 1 equation.

Deduction rule boolean.1:
  when  not(x) == true
  yield x == false
has been applied to equation Case.3.2:
  not(in_stack(c_x, deqd(c_xst))) == true
to yield the following equations:
  Case.3.2.1: in_stack(c_x, deqd(c_xst)) == false

Ordered equation Case.3.2 into the rewrite rule:
  not(in_stack(c_x, deqd(c_xst))) -> true

Ordered equation Case.3.2.1 into the rewrite rule:
  in_stack(c_x, deqd(c_xst)) -> false

    Left-hand side reduced:
    not(in_stack(c_x, deqd(c_xst))) -> true
      became equation Case.3.2:
      not(false) == true

Ordered equation Case.3.2 into the rewrite rule:
  not(false) -> true

The case system now contains 2 rewrite rules.
```

```
The system now contains 1 equation, 68 rewrite rules, and 5 deduction rules.

Deduction rule boolean.1:
  when  not(x) == true
  yield x == false
has been applied to equation Case.3.2:
  not(in_stack(c_x, deqd(c_xst))) == true
to yield the following equations:
  Case.3.2.3: in_stack(c_x, deqd(c_xst)) == false

Ordered equation Case.3.2 into the rewrite rule:
  not(in_stack(c_x, deqd(c_xst))) -> true

Ordered equation Case.3.2.3 into the rewrite rule:
  in_stack(c_x, deqd(c_xst)) -> false

    Left-hand side reduced:
    not(in_stack(c_x, deqd(c_xst))) -> true
      became equation Case.3.2:
      not(false) == true

The system now contains 69 rewrite rules and 5 deduction rules.

Lemma thm3.1.2.2 in the proof by cases of Lemma thm3.1.2
    Inv3(deq(c_xst, vi1, vi2), c_x) -> true
    Case.3.2: not(in_stack(c_x, deqd(c_xst)))
is NOT provable using the current partially completed system.  It reduces to
the equation
    (enqr(c_x) < deqr(c_x)) | not(c_x = trip(element(vi2), enqt(vi2), vi1))
    -> true

Proof of Lemma thm3.1.2.2 suspended.

Critical-pair computation abandoned because a theorem has been proved.

Computed 1 new critical pair.  Added 1 of them to the system.

-> resume by case c_x=trip(element(vi2::enq_rec),enqt(vi2::enq_rec),vi1)

Case.4.1
    c_x = trip(element(c_vi2), enqt(c_vi2), c_vi1) == true
involves proving Lemma thm3.1.2.2.1
    Inv3(deq(c_xst, c_vi1, c_vi2), c_x) -> true

The case system now contains 1 equation.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.4.1:
  c_x = trip(element(c_vi2), enqt(c_vi2), c_vi1) == true
to yield the following equations:
  Case.4.1.1: c_x == trip(element(c_vi2), enqt(c_vi2), c_vi1)

Ordered equation Case.4.1.1 into the rewrite rule:
  c_x -> trip(element(c_vi2), enqt(c_vi2), c_vi1)

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 69 rewrite rules, and 5 deduction rules.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.4.1:
  c_x = trip(element(c_vi2), enqt(c_vi2), c_vi1) == true
to yield the following equations:
  Case.4.1.2: c_x == trip(element(c_vi2), enqt(c_vi2), c_vi1)
```

```
Ordered equation Case.4.1.2 into the rewrite rule:
  c_x -> trip(element(c_vi2), enqt(c_vi2), c_vi1)

    Left-hand side reduced:
    in_stack(c_x, deqd(c_xst)) -> false
      became equation Case.3.2.3:
      in_stack(trip(element(c_vi2), enqt(c_vi2), c_vi1), deqd(c_xst)) == false

Ordered equation Case.3.2.3 into the rewrite rule:
  in_stack(trip(element(c_vi2), enqt(c_vi2), c_vi1), deqd(c_xst)) -> false

The system now contains 70 rewrite rules and 5 deduction rules.

Lemma thm3.1.2.2.1 in the proof by cases of Lemma thm3.1.2.2
    Inv3(deq(c_xst, c_vi1, c_vi2), c_x) -> true
    Case.4.1: c_x = trip(element(c_vi2), enqt(c_vi2), c_vi1)
is NOT provable using the current partially completed system.  It reduces to
the equation
    enqt(c_vi2) < c_vi1 -> true

Proof of Lemma thm3.1.2.2.1 suspended.

-> add when_deq(c_xst,c_x,c_vi1,c_vi2)

Added 1 equation to the system.

Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation thm3.3:
  (enqt(c_vi2) < c_vi1)
   & in(c_vi2, enqd(c_xst))
   & least(c_vi2, enqd(c_xst))
   & (((deqr(top(deqd(c_xst))) < c_vi1)
        & (enqr(top(deqd(c_xst))) < enqt(c_vi2)))
      | (deqd(c_xst) = new))

  -> true
to yield the following equations:
  thm3.3.1: enqt(c_vi2) < c_vi1 == true
  thm3.3.2: in(c_vi2, enqd(c_xst)) == true
  thm3.3.3: least(c_vi2, enqd(c_xst)) == true
  thm3.3.4: ((deqr(top(deqd(c_xst))) < c_vi1)
              & (enqr(top(deqd(c_xst))) < enqt(c_vi2)))
             | (deqd(c_xst) = new)
            == true

Ordered equation thm3.3.4 into the rewrite rule:
  ((deqr(top(deqd(c_xst))) < c_vi1) & (enqr(top(deqd(c_xst))) < enqt(c_vi2)))
   | (deqd(c_xst) = new)
  -> true

Ordered equation thm3.3.3 into the rewrite rule:
  least(c_vi2, enqd(c_xst)) -> true

Ordered equation thm3.3.2 into the rewrite rule:
  in(c_vi2, enqd(c_xst)) -> true

Ordered equation thm3.3.1 into the rewrite rule:
  enqt(c_vi2) < c_vi1 -> true

The system now contains 74 rewrite rules and 5 deduction rules.

Lemma thm3.1.2.2.1 in the proof by cases of Lemma thm3.1.2.2
    Inv3(deq(c_xst, c_vi1, c_vi2), c_x) -> true
    Case.4.1: c_x = trip(element(c_vi2), enqt(c_vi2), c_vi1)
[] Proved by rewriting.
```

```
Case.4.2
    not(c_x = trip(element(c_vi2), enqt(c_vi2), c_vi1)) == true
involves proving Lemma thm3.1.2.2.2
    Inv3(deq(c_xst, c_vi1, c_vi2), c_x) -> true

The case system now contains 1 equation.

Deduction rule boolean.1:
  when  not(x) == true
  yield x == false
has been applied to equation Case.4.2:
  not(c_x = trip(element(c_vi2), enqt(c_vi2), c_vi1)) == true
to yield the following equations:
  Case.4.2.1: c_x = trip(element(c_vi2), enqt(c_vi2), c_vi1) == false

Ordered equation Case.4.2 into the rewrite rule:
  not(c_x = trip(element(c_vi2), enqt(c_vi2), c_vi1)) -> true

Ordered equation Case.4.2.1 into the rewrite rule:
  c_x = trip(element(c_vi2), enqt(c_vi2), c_vi1) -> false

    Left-hand side reduced:
    not(c_x = trip(element(c_vi2), enqt(c_vi2), c_vi1)) -> true
      became equation Case.4.2:
      not(false) == true

Ordered equation Case.4.2 into the rewrite rule:
  not(false) -> true

The case system now contains 2 rewrite rules.

Lemma thm3.1.2.2.2 in the proof by cases of Lemma thm3.1.2.2
    Inv3(deq(c_xst, c_vi1, c_vi2), c_x) -> true
    Case.4.2: not(c_x = trip(element(c_vi2), enqt(c_vi2), c_vi1))
[] Proved by rewriting (with unreduced rules).

Lemma thm3.1.2.2 in the proof by cases of Lemma thm3.1.2
    Inv3(deq(c_xst, vi1, vi2), c_x) -> true
    Case.3.2: not(in_stack(c_x, deqd(c_xst)))
[] Proved by cases
     (c_x = trip(element(vi2), enqt(vi2), vi1))
      | not(c_x = trip(element(vi2), enqt(vi2), vi1))


Lemma thm3.1.2 for the induction step in the proof of Conjecture thm3.1
    Inv3(deq(c_xst, vi1, vi2), x) -> true
[] Proved by cases
    in_stack(x, deqd(c_xst)) | not(in_stack(x, deqd(c_xst)))

Conjecture thm3.1
    Inv3(xst, x) -> true
[] Proved by induction over 'xst::St' of sort 'St'.

The system now contains 1 equation, 67 rewrite rules, and 5 deduction rules.

Ordered equation thm3.1 into the rewrite rule:
  (enqr(x) < deqr(x)) | not(in_stack(x, deqd(xst))) -> true

The system now contains 68 rewrite rules and 5 deduction rules.

-> qed

All conjectures have been proved.

-> q
```

5. Four Sets of Helping Lemmas

5.1. Helping Lemma Set 0

```
add
(x=pair(y,z))=>((element(x)=y)&(enqt(x)=z))
(x=trip(u,v,w))=>((what(x)=u)&(enqr(x)=v)&(deqr(x)=w))
in_stack(x,y)=>(deq_before(x,top(y),y)  |  (x=top(y)))
..
```

5.2. LP Proof Session of Lemma Set 0

```
-> thaw ab

System thawed from 'ab.frz'.

-> set name lemma

The name prefix is now 'lemma'.

-> set axiom o

The axiom use is now 'order-equations-into-rules'.

-> prove (x=pair(y,z))=>((element(x)=y)&(enqt(x)=z)) by case x=pair(y,z)

Case.1.1
    c_x = pair(c_y, c_z) == true
involves proving Lemma lemma.1.1
    (c_x = pair(c_y, c_z)) => ((c_y = element(c_x)) & (c_z = enqt(c_x)))
    -> true

The case system now contains 1 equation.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.1.1:
  c_x = pair(c_y, c_z) == true
to yield the following equations:
  Case.1.1.1: c_x == pair(c_y, c_z)

Ordered equation Case.1.1.1 into the rewrite rule:
  c_x -> pair(c_y, c_z)

The case system now contains 1 rewrite rule.

Lemma lemma.1.1 in the proof by cases of Conjecture lemma.1
    (c_x = pair(c_y, c_z)) => ((c_y = element(c_x)) & (c_z = enqt(c_x)))
    -> true
    Case.1.1: c_x = pair(c_y, c_z)
[] Proved by rewriting (with unreduced rules).

Case.1.2
    not(c_x = pair(c_y, c_z)) == true
involves proving Lemma lemma.1.2
    (c_x = pair(c_y, c_z)) => ((c_y = element(c_x)) & (c_z = enqt(c_x)))
    -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.1.2:
  (c_x = pair(c_y, c_z)) <=> false == true
to yield the following equations:
  Case.1.2.1: c_x = pair(c_y, c_z) == false

Ordered equation Case.1.2.1 into the rewrite rule:
  c_x = pair(c_y, c_z) -> false

The case system now contains 1 rewrite rule.

Lemma lemma.1.2 in the proof by cases of Conjecture lemma.1
    (c_x = pair(c_y, c_z)) => ((c_y = element(c_x)) & (c_z = enqt(c_x)))
    -> true
    Case.1.2: not(c_x = pair(c_y, c_z))
```

```
[] Proved by rewriting (with unreduced rules).

Conjecture lemma.1
    (pair(y, z) = x) => ((element(x) = y) & (enqt(x) = z)) -> true
[] Proved by cases
    (pair(y, z) = x) | not(pair(y, z) = x)

The system now contains 1 equation, 135 rewrite rules, and 12 deduction rules.

Ordered equation lemma.1 into the rewrite rule:
  ((element(x) = y) & (enqt(x) = z)) | ((pair(y, z) = x) <=> false) -> true

The system now contains 136 rewrite rules and 12 deduction rules.

-> prove (x=trip(u,v,w))=>((what(x)=u)&(enqr(x)=v)&(deqr(x)=w)) by case x=trip(u,v,w)

Case.2.1
    c_x = trip(c_u, c_v, c_w) == true
involves proving Lemma lemma.2.1
    (c_x = trip(c_u, c_v, c_w))
     => ((c_u = what(c_x)) & (c_v = enqr(c_x)) & (c_w = deqr(c_x)))
    -> true

The case system now contains 1 equation.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.2.1:
  c_x = trip(c_u, c_v, c_w) == true
to yield the following equations:
  Case.2.1.1: c_x == trip(c_u, c_v, c_w)

Ordered equation Case.2.1.1 into the rewrite rule:
  c_x -> trip(c_u, c_v, c_w)

The case system now contains 1 rewrite rule.

Lemma lemma.2.1 in the proof by cases of Conjecture lemma.2
    (c_x = trip(c_u, c_v, c_w))
     => ((c_u = what(c_x)) & (c_v = enqr(c_x)) & (c_w = deqr(c_x)))
    -> true
    Case.2.1: c_x = trip(c_u, c_v, c_w)
[] Proved by rewriting (with unreduced rules).

Case.2.2
    not(c_x = trip(c_u, c_v, c_w)) == true
involves proving Lemma lemma.2.2
    (c_x = trip(c_u, c_v, c_w))
     => ((c_u = what(c_x)) & (c_v = enqr(c_x)) & (c_w = deqr(c_x)))
    -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.2.2:
  (c_x = trip(c_u, c_v, c_w)) <=> false == true
to yield the following equations:
  Case.2.2.1: c_x = trip(c_u, c_v, c_w) == false

Ordered equation Case.2.2.1 into the rewrite rule:
  c_x = trip(c_u, c_v, c_w) -> false

The case system now contains 1 rewrite rule.

Lemma lemma.2.2 in the proof by cases of Conjecture lemma.2
    (c_x = trip(c_u, c_v, c_w))
```

```
     => ((c_u = what(c_x)) & (c_v = enqr(c_x)) & (c_w = deqr(c_x)))
    -> true
    Case.2.2: not(c_x = trip(c_u, c_v, c_w))
[] Proved by rewriting (with unreduced rules).

Conjecture lemma.2
    (trip(u, v, w) = x) => ((deqr(x) = w) & (enqr(x) = v) & (what(x) = u))
    -> true
[] Proved by cases
    (trip(u, v, w) = x) | not(trip(u, v, w) = x)

The system now contains 1 equation, 136 rewrite rules, and 12 deduction rules.

Ordered equation lemma.2 into the rewrite rule:
  ((deqr(x) = w) & (enqr(x) = v) & (what(x) = u))
   | ((trip(u, v, w) = x) <=> false)
  -> true

The system now contains 137 rewrite rules and 12 deduction rules.


-> prove in_stack(x,y)=>(deq_before(x,top(y),y) | (x=top(y))) by induction y deq_stack

The basis step in an inductive proof of Conjecture lemma.3
    in_stack(x, y) => ((top(y) = x) | deq_before(x, top(y), y)) -> true
involves proving the following lemma(s):

lemma.3.1: in_stack(x, new) => ((top(new) = x) | deq_before(x, top(new), new))
           -> true
           [] Proved by normalization

The induction step in an inductive proof of Conjecture lemma.3
    in_stack(x, y) => ((top(y) = x) | deq_before(x, top(y), y)) -> true
uses the following equation(s) for the induction hypothesis:

Induct.1: in_stack(x, c_y) => ((top(c_y) = x) | deq_before(x, top(c_y), c_y))
          -> true

The system now contains 1 equation, 137 rewrite rules, and 12 deduction rules.

Ordered equation Induct.1 into the rewrite rule:
  (false <=> in_stack(x, c_y)) | (top(c_y) = x) | deq_before(x, top(c_y), c_y)
  -> true

The system now contains 138 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma.3.2: in_stack(x, push(c_y, vi1))
            => ((top(push(c_y, vi1)) = x)
                 | deq_before(x, top(push(c_y, vi1)), push(c_y, vi1)))

           -> true
           [] Proved by normalization

Conjecture lemma.3
    in_stack(x, y) => ((top(y) = x) | deq_before(x, top(y), y)) -> true
[] Proved by induction over 'y' of sort 'deq_stack'.

The system now contains 1 equation, 137 rewrite rules, and 12 deduction rules.

Ordered equation lemma.3 into the rewrite rule:
  (false <=> in_stack(x, y)) | (top(y) = x) | deq_before(x, top(y), y) -> true

The system now contains 138 rewrite rules and 12 deduction rules.

-> forget undo

Undo stack cleared.
```

```
-> freeze theory

System frozen in 'theory.frz'.

-> q
```

5.3. Helping Lemma Set 1

```
add
  append(append(x, y), z) -> append(x, append(y, z))
  (append(x, sub(y, x)) = y) | not(prefix(x, y)) -> true
  (cons:Seq,EL->Seq(y, z) = x) | not(prefix(x, cons:Seq,EL->Seq(y, z))) |
                                  prefix(x, y) -> true
  append(ENQ(x), ENQ(y)) -> ENQ(append(x, y))
  append(DEQ(x), DEQ(y)) -> DEQ(append(x, y))
  ENQ(append(cons(x, E(y)), z)) ->
                          append(cons:Seq,EL->Seq(ENQ(x), element(y)), ENQ(z))
  ENQ(append(cons(x, D(y)), z)) -> ENQ(append(x, z))
  DEQ(append(cons(x, E(y)), z)) -> DEQ(append(x, z))
  DEQ(append(cons(x, D(y)), z)) ->
                            append(cons:Seq,EL->Seq(DEQ(x), what(y)), DEQ(z))
  (DEQ(x) = DEQ(y)) | not(x = y) -> true
  (ENQ(x) = ENQ(y)) | not(x = y) -> true
  (x=null:->H) | not(in_state(x, init)) -> true
  not(prefix(x,y)) | prefix(x, cons:Seq,EL->Seq(y, z)) -> true
  not(prefix(cons:Seq,EL->Seq(x, z), y)) | prefix(x, y) -> true
  in_state(xh, xst) | not(in_state(cons(xh, we::Ev), xst)) -> true
  prefix(x, append(x, y)) -> true
  (in_state(xh, xst) & prefix(DEQ(xh),ENQ(xh)))=>
          prefix(DEQ(discard(xt,xh)),ENQ(discard(xt,xh)))
..
```

5.4. LP Proof Session of Lemma Set 1

```
-> thaw theory

System thawed from `theory.frz'.

-> set axiom o

The axiom use is now `order-equations-into-rules'.

-> set name lemma1

The name prefix is now `lemma1'.

->  prove append(x,append(y,z)) = append(append(x,y),z) by induction z Seq

The basis step in an inductive proof of Conjecture lemma1.1
    append(append(x, y), z) == append(x, append(y, z))
involves proving the following lemma(s):

lemma1.1.1: append(append(x, y), null) == append(x, append(y, null))
            [] Proved by normalization

The induction step in an inductive proof of Conjecture lemma1.1
    append(append(x, y), z) == append(x, append(y, z))
uses the following equation(s) for the induction hypothesis:

Induct.2: append(append(x, y), c_z) == append(x, append(y, c_z))

The system now contains 1 equation, 138 rewrite rules, and 12 deduction rules.

Ordered equation Induct.2 into the rewrite rule:
  append(append(x, y), c_z) -> append(x, append(y, c_z))

The system now contains 139 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma1.1.2: append(append(x, y), cons(c_z, vil))
            == append(x, append(y, cons(c_z, vil)))
            [] Proved by normalization

Conjecture lemma1.1
    append(append(x, y), z) == append(x, append(y, z))
[] Proved by induction over `z' of sort `Seq'.

The system now contains 1 equation, 138 rewrite rules, and 12 deduction rules.

Ordered equation lemma1.1 into the rewrite rule:
  append(append(x, y), z) -> append(x, append(y, z))

The system now contains 139 rewrite rules and 12 deduction rules.

-> prove prefix(x,y)=>(append(x,sub(y,x))=y) by induction y Seq

The basis step in an inductive proof of Conjecture lemma1.2
    prefix(x, y) => (append(x, sub(y, x)) = y) -> true
involves proving the following lemma(s):

lemma1.2.1: prefix(x, null) => (append(x, sub(null, x)) = null) -> true
                which reduces to the equation
                (false <=> prefix(x, null)) | (null = x) -> true

Proof of Lemma lemma1.2.1 suspended.

-> resume by induction x Seq

The basis step in an inductive proof of Lemma lemma1.2.1 for the basis step in
```

```
the proof of Conjecture lemma1.2
    prefix(x, null) => (append(x, sub(null, x)) = null) -> true
involves proving the following lemma(s):

lemma1.2.1.1: prefix(null, null) => (append(null, sub(null, null)) = null)
              -> true
              [] Proved by normalization

The induction step in an inductive proof of Lemma lemma1.2.1 for the basis step
in the proof of Conjecture lemma1.2
    prefix(x, null) => (append(x, sub(null, x)) = null) -> true
uses the following equation(s) for the induction hypothesis:

Induct.3: prefix(c_x, null) => (append(c_x, sub(null, c_x)) = null) -> true

The system now contains 1 equation, 139 rewrite rules, and 12 deduction rules.

Ordered equation Induct.3 into the rewrite rule:
  (false <=> prefix(c_x, null)) | (c_x = null) -> true

The system now contains 140 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma1.2.1.2: prefix(cons(c_x, vil), null)
               => (append(cons(c_x, vil), sub(null, cons(c_x, vil))) = null)
              -> true
              [] Proved by normalization

Lemma lemma1.2.1 for the basis step in the proof of Conjecture lemma1.2
    prefix(x, null) => (append(x, sub(null, x)) = null) -> true
[] Proved by induction over 'x' of sort 'Seq'.

The induction step in an inductive proof of Conjecture lemma1.2
    prefix(x, y) => (append(x, sub(y, x)) = y) -> true
uses the following equation(s) for the induction hypothesis:

Induct.4: prefix(x, c_y) => (append(x, sub(c_y, x)) = c_y) -> true

The system now contains 1 equation, 139 rewrite rules, and 12 deduction rules.

Ordered equation Induct.4 into the rewrite rule:
  (false <=> prefix(x, c_y)) | (append(x, sub(c_y, x)) = c_y) -> true

The system now contains 140 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma1.2.2: prefix(x, cons(c_y, vil))
             => (append(x, sub(cons(c_y, vil), x)) = cons(c_y, vil))
            -> true
                which reduces to the equation
                (false <=> prefix(x, cons(c_y, vil)))
                 | (append(x, sub(cons(c_y, vil), x)) = cons(c_y, vil))
                -> true

Proof of Lemma lemma1.2.2 suspended.

-> resume by induction x Seq

The basis step in an inductive proof of Lemma lemma1.2.2 for the induction step
in the proof of Conjecture lemma1.2
    prefix(x, cons(c_y, vil))
     => (append(x, sub(cons(c_y, vil), x)) = cons(c_y, vil))
    -> true
involves proving the following lemma(s):

lemma1.2.2.1: prefix(null, cons(c_y, vil))
               => (append(null, sub(cons(c_y, vil), null)) = cons(c_y, vil))
```

```
              -> true
              [] Proved by normalization

The induction step in an inductive proof of Lemma lemma1.2.2 for the induction
step in the proof of Conjecture lemma1.2
    prefix(x, cons(c_y, vi1))
     => (append(x, sub(cons(c_y, vi1), x)) = cons(c_y, vi1))
    -> true
uses the following equation(s) for the induction hypothesis:

Induct.5: prefix(c_x, cons(c_y, vi1))
           => (append(c_x, sub(cons(c_y, vi1), c_x)) = cons(c_y, vi1))
          -> true

The system now contains 1 equation, 140 rewrite rules, and 12 deduction rules.

Ordered equation Induct.5 into the rewrite rule:
  (false <=> prefix(c_x, cons(c_y, vi1)))
   | (append(c_x, sub(cons(c_y, vi1), c_x)) = cons(c_y, vi1))
  -> true

The system now contains 141 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma1.2.2.2: prefix(cons(c_x, vi2), cons(c_y, vi1))
               => (append(cons(c_x, vi2), sub(cons(c_y, vi1), cons(c_x, vi2)))
                    = cons(c_y, vi1))

              -> true
                  which reduces to the equation
                  ((false <=> prefix(cons(c_x, vi2), c_y))
                    & (((c_x = c_y) <=> false) | ((vi1 = vi2) <=> false)))
                   | ((c_x = c_y) & (vi1 = vi2))
                   | (append(cons(c_x, vi2), sub(c_y, cons(c_x, vi2))) = c_y)
                  -> true

Proof of Lemma lemma1.2.2.2 suspended.

->  resume by case (c_x=c_y)&(vi1=vi2)

Case.3.1
    (c_vi1 = c_vi2) & (c_x = c_y) == true
involves proving Lemma lemma1.2.2.2.1
    prefix(cons(c_x, c_vi2), cons(c_y, c_vi1))
     => (append(cons(c_x, c_vi2), sub(cons(c_y, c_vi1), cons(c_x, c_vi2)))
          = cons(c_y, c_vi1))

    -> true

The case system now contains 1 equation.

Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation Case.3.1:
  (c_vi1 = c_vi2) & (c_x = c_y) == true
to yield the following equations:
  Case.3.1.1: c_vi1 = c_vi2 == true
  Case.3.1.2: c_x = c_y == true

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.3.1.2:
  c_x = c_y == true
to yield the following equations:
  Case.3.1.2.1: c_x == c_y
```

```
Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.3.1.1:
  c_vi1 = c_vi2 == true
to yield the following equations:
  Case.3.1.1.1: c_vi1 == c_vi2

Ordered equation Case.3.1.2.1 into the rewrite rule:
  c_x -> c_y

The case system now contains 1 equation and 1 rewrite rule.

Ordered equation Case.3.1.1.1 into the rewrite rule:
  c_vi1 -> c_vi2

The case system now contains 2 rewrite rules.

Lemma lemma1.2.2.2.1 in the proof by cases of Lemma lemma1.2.2.2
    prefix(cons(c_x, c_vi2), cons(c_y, c_vi1))
     => (append(cons(c_x, c_vi2), sub(cons(c_y, c_vi1), cons(c_x, c_vi2)))
          = cons(c_y, c_vi1))

    -> true
    Case.3.1: (c_vi1 = c_vi2) & (c_x = c_y)
[] Proved by rewriting (with unreduced rules).

Case.3.2
    not((c_vi1 = c_vi2) & (c_x = c_y)) == true
involves proving Lemma lemma1.2.2.2.2
    prefix(cons(c_x, c_vi2), cons(c_y, c_vi1))
     => (append(cons(c_x, c_vi2), sub(cons(c_y, c_vi1), cons(c_x, c_vi2)))
          = cons(c_y, c_vi1))

    -> true

The case system now contains 1 equation.

Ordered equation Case.3.2 into the rewrite rule:
  ((c_vi1 = c_vi2) <=> false) | ((c_x = c_y) <=> false) -> true

The case system now contains 1 rewrite rule.

Lemma lemma1.2.2.2.2 in the proof by cases of Lemma lemma1.2.2.2
    prefix(cons(c_x, c_vi2), cons(c_y, c_vi1))
     => (append(cons(c_x, c_vi2), sub(cons(c_y, c_vi1), cons(c_x, c_vi2)))
          = cons(c_y, c_vi1))

    -> true
    Case.3.2: not((c_vi1 = c_vi2) & (c_x = c_y))
[] Proved by rewriting (with unreduced rules).

Lemma lemma1.2.2.2 for the induction step in the proof of Lemma lemma1.2.2
    prefix(cons(c_x, vi2), cons(c_y, vi1))
     => (append(cons(c_x, vi2), sub(cons(c_y, vi1), cons(c_x, vi2)))
          = cons(c_y, vi1))

    -> true
[] Proved by cases
    ((c_x = c_y) & (vi1 = vi2)) | not((c_x = c_y) & (vi1 = vi2))

Lemma lemma1.2.2 for the induction step in the proof of Conjecture lemma1.2
    prefix(x, cons(c_y, vi1))
     => (append(x, sub(cons(c_y, vi1), x)) = cons(c_y, vi1))
    -> true
[] Proved by induction over `x' of sort `Seq'.

Conjecture lemma1.2
```

```
    prefix(x, y) => (append(x, sub(y, x)) = y) -> true
[] Proved by induction over 'y' of sort 'Seq'.

The system now contains 1 equation, 139 rewrite rules, and 12 deduction rules.

Ordered equation lemma1.2 into the rewrite rule:
  (false <=> prefix(x, y)) | (append(x, sub(y, x)) = y) -> true

The system now contains 140 rewrite rules and 12 deduction rules.

->  prove prefix(x,cons:Seq,EL->Seq(y,z))=>(prefix(x,y) | x=cons:Seq,EL->Seq(y,z)) by induction x Seq

The basis step in an inductive proof of Conjecture lemma1.3
    prefix(x, cons(y, z)) => ((cons(y, z) = x) | prefix(x, y)) -> true
involves proving the following lemma(s):

lemma1.3.1: prefix(null, cons(y, z)) => ((cons(y, z) = null) | prefix(null, y))
            -> true
            [] Proved by normalization

The induction step in an inductive proof of Conjecture lemma1.3
    prefix(x, cons(y, z)) => ((cons(y, z) = x) | prefix(x, y)) -> true
uses the following equation(s) for the induction hypothesis:

Induct.6: prefix(c_x, cons(y, z)) => ((c_x = cons(y, z)) | prefix(c_x, y))
          -> true

The system now contains 1 equation, 140 rewrite rules, and 12 deduction rules.

Ordered equation Induct.6 into the rewrite rule:
  (false <=> prefix(c_x, cons(y, z))) | (c_x = cons(y, z)) | prefix(c_x, y)
  -> true

The system now contains 141 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma1.3.2: prefix(cons(c_x, vil), cons(y, z))
             => ((cons(c_x, vil) = cons(y, z)) | prefix(cons(c_x, vil), y))
            -> true
            [] Proved by normalization

Conjecture lemma1.3
    prefix(x, cons(y, z)) => ((cons(y, z) = x) | prefix(x, y)) -> true
[] Proved by induction over 'x' of sort 'Seq'.

The system now contains 1 equation, 140 rewrite rules, and 12 deduction rules.

Ordered equation lemma1.3 into the rewrite rule:
  (false <=> prefix(x, cons(y, z))) | (cons(y, z) = x) | prefix(x, y) -> true

The system now contains 141 rewrite rules and 12 deduction rules.

->  prove ENQ(append(x,y))==append(ENQ(x),ENQ(y)) by induction y H

The basis step in an inductive proof of Conjecture lemma1.4
    ENQ(append(x, y)) == append(ENQ(x), ENQ(y))
involves proving the following lemma(s):

lemma1.4.1: ENQ(append(x, null)) == append(ENQ(x), ENQ(null))
            [] Proved by normalization

The induction step in an inductive proof of Conjecture lemma1.4
    ENQ(append(x, y)) == append(ENQ(x), ENQ(y))
uses the following equation(s) for the induction hypothesis:

Induct.7: ENQ(append(x, c_y)) == append(ENQ(x), ENQ(c_y))

The system now contains 1 equation, 141 rewrite rules, and 12 deduction rules.
```

```
Ordered equation Induct.7 into the rewrite rule:
  append(ENQ(x), ENQ(c_y)) -> ENQ(append(x, c_y))

The system now contains 142 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma1.4.2: ENQ(append(x, cons(c_y, vi1)))
            == append(ENQ(x), ENQ(cons(c_y, vi1)))
                which reduces to the equation
                ENQ(cons(append(x, c_y), vi1))
                == append(ENQ(x), ENQ(cons(c_y, vi1)))

Proof of Lemma lemma1.4.2 suspended.

->  resume by induction vi1 Ev

The basis step in an inductive proof of Lemma lemma1.4.2 for the induction step
in the proof of Conjecture lemma1.4
    ENQ(append(x, cons(c_y, vi1))) == append(ENQ(x), ENQ(cons(c_y, vi1)))
involves proving the following lemma(s):

lemma1.4.2.1: ENQ(append(x, cons(c_y, E(vi2))))
              == append(ENQ(x), ENQ(cons(c_y, E(vi2))))
              [] Proved by normalization
lemma1.4.2.2: ENQ(append(x, cons(c_y, D(vi2))))
              == append(ENQ(x), ENQ(cons(c_y, D(vi2))))
              [] Proved by normalization

The induction step in an inductive proof of Lemma lemma1.4.2 for the induction
step in the proof of Conjecture lemma1.4
    ENQ(append(x, cons(c_y, vi1))) == append(ENQ(x), ENQ(cons(c_y, vi1)))
is vacuous.

Lemma lemma1.4.2 for the induction step in the proof of Conjecture lemma1.4
    ENQ(append(x, cons(c_y, vi1))) == append(ENQ(x), ENQ(cons(c_y, vi1)))
[] Proved by induction over `vi1::Ev' of sort `Ev'.

Conjecture lemma1.4
    ENQ(append(x, y)) == append(ENQ(x), ENQ(y))
[] Proved by induction over `y' of sort `H'.

The system now contains 1 equation, 141 rewrite rules, and 12 deduction rules.

Ordered equation lemma1.4 into the rewrite rule:
  append(ENQ(x), ENQ(y)) -> ENQ(append(x, y))

The system now contains 142 rewrite rules and 12 deduction rules.

->  prove DEQ(append(x,y))=append(DEQ(x),DEQ(y)) by induction y H

The basis step in an inductive proof of Conjecture lemma1.5
    DEQ(append(x, y)) == append(DEQ(x), DEQ(y))
involves proving the following lemma(s):

lemma1.5.1: DEQ(append(x, null)) == append(DEQ(x), DEQ(null))
            [] Proved by normalization

The induction step in an inductive proof of Conjecture lemma1.5
    DEQ(append(x, y)) == append(DEQ(x), DEQ(y))
uses the following equation(s) for the induction hypothesis:

Induct.8: DEQ(append(x, c_y)) == append(DEQ(x), DEQ(c_y))

The system now contains 1 equation, 142 rewrite rules, and 12 deduction rules.

Ordered equation Induct.8 into the rewrite rule:
  append(DEQ(x), DEQ(c_y)) -> DEQ(append(x, c_y))
```

```
The system now contains 143 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma1.5.2: DEQ(append(x, cons(c_y, vi1)))
            == append(DEQ(x), DEQ(cons(c_y, vi1)))
               which reduces to the equation
               DEQ(cons(append(x, c_y), vi1))
               == append(DEQ(x), DEQ(cons(c_y, vi1)))

Proof of Lemma lemma1.5.2 suspended.

-> resume by induction vi1 Ev

The basis step in an inductive proof of Lemma lemma1.5.2 for the induction step
in the proof of Conjecture lemma1.5
    DEQ(append(x, cons(c_y, vi1))) == append(DEQ(x), DEQ(cons(c_y, vi1)))
involves proving the following lemma(s):

lemma1.5.2.1: DEQ(append(x, cons(c_y, E(vi2))))
              == append(DEQ(x), DEQ(cons(c_y, E(vi2))))
              [] Proved by normalization
lemma1.5.2.2: DEQ(append(x, cons(c_y, D(vi2))))
              == append(DEQ(x), DEQ(cons(c_y, D(vi2))))
              [] Proved by normalization

The induction step in an inductive proof of Lemma lemma1.5.2 for the induction
step in the proof of Conjecture lemma1.5
    DEQ(append(x, cons(c_y, vi1))) == append(DEQ(x), DEQ(cons(c_y, vi1)))
is vacuous.

Lemma lemma1.5.2 for the induction step in the proof of Conjecture lemma1.5
    DEQ(append(x, cons(c_y, vi1))) == append(DEQ(x), DEQ(cons(c_y, vi1)))
[] Proved by induction over 'vi1::Ev' of sort 'Ev'.

Conjecture lemma1.5
    DEQ(append(x, y)) == append(DEQ(x), DEQ(y))
[] Proved by induction over 'y' of sort 'H'.

The system now contains 1 equation, 142 rewrite rules, and 12 deduction rules.

Ordered equation lemma1.5 into the rewrite rule:
  append(DEQ(x), DEQ(y)) -> DEQ(append(x, y))

The system now contains 143 rewrite rules and 12 deduction rules.

->  prove ENQ(append(cons(x,E(y)),z))=append(cons:Seq,EL->Seq(ENQ(x),element(y)),ENQ(z)) by induction
z H

The basis step in an inductive proof of Conjecture lemma1.6
    ENQ(append(cons(x, E(y)), z)) == append(cons(ENQ(x), element(y)), ENQ(z))
involves proving the following lemma(s):

lemma1.6.1: ENQ(append(cons(x, E(y)), null))
            == append(cons(ENQ(x), element(y)), ENQ(null))
            [] Proved by normalization

The induction step in an inductive proof of Conjecture lemma1.6
    ENQ(append(cons(x, E(y)), z)) == append(cons(ENQ(x), element(y)), ENQ(z))
uses the following equation(s) for the induction hypothesis:

Induct.9: ENQ(append(cons(x, E(y)), c_z))
          == append(cons(ENQ(x), element(y)), ENQ(c_z))

The system now contains 1 equation, 143 rewrite rules, and 12 deduction rules.

Ordered equation Induct.9 into the rewrite rule:
  ENQ(append(cons(x, E(y)), c_z)) -> append(cons(ENQ(x), element(y)), ENQ(c_z))
```

```
The system now contains 144 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma1.6.2: ENQ(append(cons(x, E(y)), cons(c_z, vi1)))
            == append(cons(ENQ(x), element(y)), ENQ(cons(c_z, vi1)))
               which reduces to the equation
               ENQ(cons(append(cons(x, E(y)), c_z), vi1))
               == append(cons(ENQ(x), element(y)), ENQ(cons(c_z, vi1)))

Proof of Lemma lemma1.6.2 suspended.

-> resume by induction vi1 Ev

The basis step in an inductive proof of Lemma lemma1.6.2 for the induction step
in the proof of Conjecture lemma1.6
    ENQ(append(cons(x, E(y)), cons(c_z, vi1)))
    == append(cons(ENQ(x), element(y)), ENQ(cons(c_z, vi1)))
involves proving the following lemma(s):

lemma1.6.2.1: ENQ(append(cons(x, E(y)), cons(c_z, E(vi2))))
              == append(cons(ENQ(x), element(y)), ENQ(cons(c_z, E(vi2))))
              [] Proved by normalization
lemma1.6.2.2: ENQ(append(cons(x, E(y)), cons(c_z, D(vi2))))
              == append(cons(ENQ(x), element(y)), ENQ(cons(c_z, D(vi2))))
              [] Proved by normalization

The induction step in an inductive proof of Lemma lemma1.6.2 for the induction
step in the proof of Conjecture lemma1.6
    ENQ(append(cons(x, E(y)), cons(c_z, vi1)))
    == append(cons(ENQ(x), element(y)), ENQ(cons(c_z, vi1)))
is vacuous.

Lemma lemma1.6.2 for the induction step in the proof of Conjecture lemma1.6
    ENQ(append(cons(x, E(y)), cons(c_z, vi1)))
    == append(cons(ENQ(x), element(y)), ENQ(cons(c_z, vi1)))
[] Proved by induction over 'vi1::Ev' of sort 'Ev'.

Conjecture lemma1.6
    ENQ(append(cons(x, E(y)), z)) == append(cons(ENQ(x), element(y)), ENQ(z))
[] Proved by induction over 'z' of sort 'H'.

The system now contains 1 equation, 143 rewrite rules, and 12 deduction rules.

Ordered equation lemma1.6 into the rewrite rule:
  ENQ(append(cons(x, E(y)), z)) -> append(cons(ENQ(x), element(y)), ENQ(z))

The system now contains 144 rewrite rules and 12 deduction rules.

-> prove ENQ(append(cons(x,D(y)),z))=append(ENQ(x),ENQ(z)) by induction z H

The basis step in an inductive proof of Conjecture lemma1.7
    ENQ(append(cons(x, D(y)), z)) == append(ENQ(x), ENQ(z))
involves proving the following lemma(s):

lemma1.7.1: ENQ(append(cons(x, D(y)), null)) == append(ENQ(x), ENQ(null))
            [] Proved by normalization

The induction step in an inductive proof of Conjecture lemma1.7
    ENQ(append(cons(x, D(y)), z)) == append(ENQ(x), ENQ(z))
uses the following equation(s) for the induction hypothesis:

Induct.10: ENQ(append(cons(x, D(y)), c_z)) == append(ENQ(x), ENQ(c_z))

The system now contains 1 equation, 144 rewrite rules, and 12 deduction rules.

Ordered equation Induct.10 into the rewrite rule:
  ENQ(append(cons(x, D(y)), c_z)) -> ENQ(append(x, c_z))
```

```
The system now contains 145 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma1.7.2: ENQ(append(cons(x, D(y)), cons(c_z, vi1)))
            == append(ENQ(x), ENQ(cons(c_z, vi1)))
               which reduces to the equation
               ENQ(cons(append(cons(x, D(y)), c_z), vi1))
               == ENQ(cons(append(x, c_z), vi1))

Proof of Lemma lemma1.7.2 suspended.

-> resume by induction vi1 Ev

The basis step in an inductive proof of Lemma lemma1.7.2 for the induction step
in the proof of Conjecture lemma1.7
    ENQ(append(cons(x, D(y)), cons(c_z, vi1)))
    == append(ENQ(x), ENQ(cons(c_z, vi1)))
involves proving the following lemma(s):

lemma1.7.2.1: ENQ(append(cons(x, D(y)), cons(c_z, E(vi2))))
              == append(ENQ(x), ENQ(cons(c_z, E(vi2))))
              [] Proved by normalization
lemma1.7.2.2: ENQ(append(cons(x, D(y)), cons(c_z, D(vi2))))
              == append(ENQ(x), ENQ(cons(c_z, D(vi2))))
              [] Proved by normalization

The induction step in an inductive proof of Lemma lemma1.7.2 for the induction
step in the proof of Conjecture lemma1.7
    ENQ(append(cons(x, D(y)), cons(c_z, vi1)))
    == append(ENQ(x), ENQ(cons(c_z, vi1)))
is vacuous.

Lemma lemma1.7.2 for the induction step in the proof of Conjecture lemma1.7
    ENQ(append(cons(x, D(y)), cons(c_z, vi1)))
    == append(ENQ(x), ENQ(cons(c_z, vi1)))
[] Proved by induction over 'vi1::Ev' of sort 'Ev'.

Conjecture lemma1.7
    ENQ(append(cons(x, D(y)), z)) == append(ENQ(x), ENQ(z))
[] Proved by induction over 'z' of sort 'H'.

The system now contains 1 equation, 144 rewrite rules, and 12 deduction rules.

Ordered equation lemma1.7 into the rewrite rule:
  ENQ(append(cons(x, D(y)), z)) -> ENQ(append(x, z))

The system now contains 145 rewrite rules and 12 deduction rules.

->  prove DEQ(append(cons(x,E(y)),z))=append(DEQ(x),DEQ(z)) by induction z H

The basis step in an inductive proof of Conjecture lemma1.8
    DEQ(append(cons(x, E(y)), z)) == append(DEQ(x), DEQ(z))
involves proving the following lemma(s):

lemma1.8.1: DEQ(append(cons(x, E(y)), null)) == append(DEQ(x), DEQ(null))
            [] Proved by normalization

The induction step in an inductive proof of Conjecture lemma1.8
    DEQ(append(cons(x, E(y)), z)) == append(DEQ(x), DEQ(z))
uses the following equation(s) for the induction hypothesis:

Induct.11: DEQ(append(cons(x, E(y)), c_z)) == append(DEQ(x), DEQ(c_z))

The system now contains 1 equation, 145 rewrite rules, and 12 deduction rules.

Ordered equation Induct.11 into the rewrite rule:
  DEQ(append(cons(x, E(y)), c_z)) -> DEQ(append(x, c_z))
```

```
The system now contains 146 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma1.8.2: DEQ(append(cons(x, E(y)), cons(c_z, vi1)))
            == append(DEQ(x), DEQ(cons(c_z, vi1)))
                which reduces to the equation
                DEQ(cons(append(cons(x, E(y)), c_z), vi1))
                == DEQ(cons(append(x, c_z), vi1))

Proof of Lemma lemma1.8.2 suspended.

-> resume by induction vi1 Ev

The basis step in an inductive proof of Lemma lemma1.8.2 for the induction step
in the proof of Conjecture lemma1.8
    DEQ(append(cons(x, E(y)), cons(c_z, vi1)))
    == append(DEQ(x), DEQ(cons(c_z, vi1)))
involves proving the following lemma(s):

lemma1.8.2.1: DEQ(append(cons(x, E(y)), cons(c_z, E(vi2))))
              == append(DEQ(x), DEQ(cons(c_z, E(vi2))))
              [] Proved by normalization
lemma1.8.2.2: DEQ(append(cons(x, E(y)), cons(c_z, D(vi2))))
              == append(DEQ(x), DEQ(cons(c_z, D(vi2))))
              [] Proved by normalization

The induction step in an inductive proof of Lemma lemma1.8.2 for the induction
step in the proof of Conjecture lemma1.8
    DEQ(append(cons(x, E(y)), cons(c_z, vi1)))
    == append(DEQ(x), DEQ(cons(c_z, vi1)))
is vacuous.

Lemma lemma1.8.2 for the induction step in the proof of Conjecture lemma1.8
    DEQ(append(cons(x, E(y)), cons(c_z, vi1)))
    == append(DEQ(x), DEQ(cons(c_z, vi1)))
[] Proved by induction over 'vi1::Ev' of sort 'Ev'.

Conjecture lemma1.8
    DEQ(append(cons(x, E(y)), z)) == append(DEQ(x), DEQ(z))
[] Proved by induction over 'z' of sort 'H'.

The system now contains 1 equation, 145 rewrite rules, and 12 deduction rules.

Ordered equation lemma1.8 into the rewrite rule:
  DEQ(append(cons(x, E(y)), z)) -> DEQ(append(x, z))

The system now contains 146 rewrite rules and 12 deduction rules.

-> prove DEQ(append(cons(x,D(y)),z))=append(cons:Seq,EL->Seq(DEQ(x),what(y)),DEQ(z)) by induction z
H

The basis step in an inductive proof of Conjecture lemma1.9
    DEQ(append(cons(x, D(y)), z)) == append(cons(DEQ(x), what(y)), DEQ(z))
involves proving the following lemma(s):

lemma1.9.1: DEQ(append(cons(x, D(y)), null))
            == append(cons(DEQ(x), what(y)), DEQ(null))
            [] Proved by normalization

The induction step in an inductive proof of Conjecture lemma1.9
    DEQ(append(cons(x, D(y)), z)) == append(cons(DEQ(x), what(y)), DEQ(z))
uses the following equation(s) for the induction hypothesis:

Induct.12: DEQ(append(cons(x, D(y)), c_z))
           == append(cons(DEQ(x), what(y)), DEQ(c_z))

The system now contains 1 equation, 146 rewrite rules, and 12 deduction rules.
```

```
Ordered equation Induct.12 into the rewrite rule:
  DEQ(append(cons(x, D(y)), c_z)) -> append(cons(DEQ(x), what(y)), DEQ(c_z))

The system now contains 147 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma1.9.2: DEQ(append(cons(x, D(y)), cons(c_z, vi1)))
            == append(cons(DEQ(x), what(y)), DEQ(cons(c_z, vi1)))
                which reduces to the equation
                DEQ(cons(append(cons(x, D(y)), c_z), vi1))
                == append(cons(DEQ(x), what(y)), DEQ(cons(c_z, vi1)))

Proof of Lemma lemma1.9.2 suspended.

-> resume by induction vi1 Ev

The basis step in an inductive proof of Lemma lemma1.9.2 for the induction step
in the proof of Conjecture lemma1.9
    DEQ(append(cons(x, D(y)), cons(c_z, vi1)))
    == append(cons(DEQ(x), what(y)), DEQ(cons(c_z, vi1)))
involves proving the following lemma(s):

lemma1.9.2.1: DEQ(append(cons(x, D(y)), cons(c_z, E(vi2))))
              == append(cons(DEQ(x), what(y)), DEQ(cons(c_z, E(vi2))))
              [] Proved by normalization
lemma1.9.2.2: DEQ(append(cons(x, D(y)), cons(c_z, D(vi2))))
              == append(cons(DEQ(x), what(y)), DEQ(cons(c_z, D(vi2))))
              [] Proved by normalization

The induction step in an inductive proof of Lemma lemma1.9.2 for the induction
step in the proof of Conjecture lemma1.9
    DEQ(append(cons(x, D(y)), cons(c_z, vi1)))
    == append(cons(DEQ(x), what(y)), DEQ(cons(c_z, vi1)))
is vacuous.

Lemma lemma1.9.2 for the induction step in the proof of Conjecture lemma1.9
    DEQ(append(cons(x, D(y)), cons(c_z, vi1)))
    == append(cons(DEQ(x), what(y)), DEQ(cons(c_z, vi1)))
[] Proved by induction over 'vi1::Ev' of sort 'Ev'.

Conjecture lemma1.9
    DEQ(append(cons(x, D(y)), z)) == append(cons(DEQ(x), what(y)), DEQ(z))
[] Proved by induction over 'z' of sort 'H'.

The system now contains 1 equation, 146 rewrite rules, and 12 deduction rules.

Ordered equation lemma1.9 into the rewrite rule:
  DEQ(append(cons(x, D(y)), z)) -> append(cons(DEQ(x), what(y)), DEQ(z))

The system now contains 147 rewrite rules and 12 deduction rules.


->  prove (x=y)=>(DEQ(x)=DEQ(y)) by induction x H

The basis step in an inductive proof of Conjecture lemma1.10
     (x = y) => (DEQ(x) = DEQ(y)) -> true
involves proving the following lemma(s):

lemma1.10.1: (null = y) => (DEQ(null) = DEQ(y)) -> true
                 which reduces to the equation
                 ((null = y) <=> false) | (DEQ(y) = null) -> true

Proof of Lemma lemma1.10.1 suspended.

->  resume by induction y H

The basis step in an inductive proof of Lemma lemma1.10.1 for the basis step in
```

```
the proof of Conjecture lemma1.10
    (null = y) => (DEQ(null) = DEQ(y)) -> true
involves proving the following lemma(s):

lemma1.10.1.1: (null = null) => (DEQ(null) = DEQ(null)) -> true
               [] Proved by normalization

The induction step in an inductive proof of Lemma lemma1.10.1 for the basis
step in the proof of Conjecture lemma1.10
    (null = y) => (DEQ(null) = DEQ(y)) -> true
uses the following equation(s) for the induction hypothesis:

Induct.13: (c_y = null) => (DEQ(c_y) = DEQ(null)) -> true

The system now contains 1 equation, 147 rewrite rules, and 12 deduction rules.

Ordered equation Induct.13 into the rewrite rule:
  ((c_y = null) <=> false) | (DEQ(c_y) = null) -> true

The system now contains 148 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma1.10.1.2: (cons(c_y, vil) = null) => (DEQ(cons(c_y, vil)) = DEQ(null))
               -> true
               [] Proved by normalization

Lemma lemma1.10.1 for the basis step in the proof of Conjecture lemma1.10
    (null = y) => (DEQ(null) = DEQ(y)) -> true
[] Proved by induction over 'y' of sort 'H'.

The induction step in an inductive proof of Conjecture lemma1.10
    (x = y) => (DEQ(x) = DEQ(y)) -> true
uses the following equation(s) for the induction hypothesis:

Induct.14: (c_x = y) => (DEQ(c_x) = DEQ(y)) -> true

The system now contains 1 equation, 147 rewrite rules, and 12 deduction rules.

Ordered equation Induct.14 into the rewrite rule:
  ((c_x = y) <=> false) | (DEQ(c_x) = DEQ(y)) -> true

The system now contains 148 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma1.10.2: (cons(c_x, vil) = y) => (DEQ(cons(c_x, vil)) = DEQ(y)) -> true
                which reduces to the equation
                ((cons(c_x, vil) = y) <=> false)
                 | (DEQ(cons(c_x, vil)) = DEQ(y))
                -> true

Proof of Lemma lemma1.10.2 suspended.

->  resume by induction y H

The basis step in an inductive proof of Lemma lemma1.10.2 for the induction
step in the proof of Conjecture lemma1.10
    (cons(c_x, vil) = y) => (DEQ(cons(c_x, vil)) = DEQ(y)) -> true
involves proving the following lemma(s):

lemma1.10.2.1: (cons(c_x, vil) = null) => (DEQ(cons(c_x, vil)) = DEQ(null))
               -> true
               [] Proved by normalization

The induction step in an inductive proof of Lemma lemma1.10.2 for the induction
step in the proof of Conjecture lemma1.10
    (cons(c_x, vil) = y) => (DEQ(cons(c_x, vil)) = DEQ(y)) -> true
uses the following equation(s) for the induction hypothesis:
```

```
Induct.15: (c_y = cons(c_x, vil)) => (DEQ(c_y) = DEQ(cons(c_x, vil))) -> true

The system now contains 1 equation, 148 rewrite rules, and 12 deduction rules.

Ordered equation Induct.15 into the rewrite rule:
  ((c_y = cons(c_x, vil)) <=> false) | (DEQ(c_y) = DEQ(cons(c_x, vil))) -> true

The system now contains 149 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma1.10.2.2: (cons(c_x, vil) = cons(c_y, vi2))
                => (DEQ(cons(c_x, vil)) = DEQ(cons(c_y, vi2)))
               -> true
                   which reduces to the equation
                   ((c_x = c_y) <=> false)
                    | ((vil = vi2) <=> false)
                    | (DEQ(cons(c_x, vil)) = DEQ(cons(c_y, vi2)))
                   -> true

Proof of Lemma lemma1.10.2.2 suspended.

->  resume by induction  vil Ev

The basis step in an inductive proof of Lemma lemma1.10.2.2 for the induction
step in the proof of Lemma lemma1.10.2
    (cons(c_x, vil) = cons(c_y, vi2))
     => (DEQ(cons(c_x, vil)) = DEQ(cons(c_y, vi2)))
    -> true
involves proving the following lemma(s):

lemma1.10.2.2.1: (cons(c_x, E(vi3)) = cons(c_y, vi2))
                  => (DEQ(cons(c_x, E(vi3))) = DEQ(cons(c_y, vi2)))
                 -> true
                     which reduces to the equation
                     ((E(vi3) = vi2) <=> false)
                      | ((c_x = c_y) <=> false)
                      | (DEQ(c_x) = DEQ(cons(c_y, vi2)))
                     -> true
lemma1.10.2.2.2: (cons(c_x, D(vi3)) = cons(c_y, vi2))
                  => (DEQ(cons(c_x, D(vi3))) = DEQ(cons(c_y, vi2)))
                 -> true
                     which reduces to the equation
                     ((D(vi3) = vi2) <=> false)
                      | ((c_x = c_y) <=> false)
                      | (DEQ(cons(c_y, vi2)) = cons(DEQ(c_x), what(vi3)))
                     -> true

Proof of Lemma lemma1.10.2.2.2 suspended.

-> resume by case c_x=c_y

Case.4.1
    c_x = c_y == true
involves proving Lemma lemma1.10.2.2.2.1
    (cons(c_x, D(vi3)) = cons(c_y, vi2))
     => (DEQ(cons(c_x, D(vi3))) = DEQ(cons(c_y, vi2)))
    -> true

The case system now contains 1 equation.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.4.1:
  c_x = c_y == true
to yield the following equations:
  Case.4.1.1: c_x == c_y
```

```
Ordered equation Case.4.1.1 into the rewrite rule:
  c_x -> c_y

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 149 rewrite rules, and 12 deduction rules.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.4.1:
  c_x = c_y == true
to yield the following equations:
  Case.4.1.2: c_x == c_y

Ordered equation Case.4.1.2 into the rewrite rule:
  c_x -> c_y

    Following 2 left-hand sides reduced:
    ((c_x = y) <=> false) | (DEQ(c_x) = DEQ(y)) -> true
      became equation Induct.14:
      ((c_y = y) <=> false) | (DEQ(c_x) = DEQ(y)) -> true
    ((c_y = cons(c_x, vi1)) <=> false) | (DEQ(c_y) = DEQ(cons(c_x, vi1)))
    -> true
      became equation Induct.15:
      ((c_y = cons(c_y, vi1)) <=> false) | (DEQ(c_y) = DEQ(cons(c_x, vi1)))
      -> true

Ordered equation Induct.14 into the rewrite rule:
  ((c_y = y) <=> false) | (DEQ(c_y) = DEQ(y)) -> true

The system now contains 149 rewrite rules and 12 deduction rules.

Lemma lemma1.10.2.2.2.1 in the proof by cases of Lemma lemma1.10.2.2.2
    (cons(c_x, D(vi3)) = cons(c_y, vi2))
     => (DEQ(cons(c_x, D(vi3))) = DEQ(cons(c_y, vi2)))
    -> true
    Case.4.1: c_x = c_y
is NOT provable using the current partially completed system.  It reduces to
the equation
    ((D(vi3) = vi2) <=> false)
     | (DEQ(cons(c_y, vi2)) = cons(DEQ(c_y), what(vi3)))
    -> true

Proof of Lemma lemma1.10.2.2.2.1 suspended.

->  resume by induction vi2 Ev

The basis step in an inductive proof of Lemma lemma1.10.2.2.2.1 in the proof by
cases of Lemma lemma1.10.2.2.2
    (cons(c_x, D(vi3)) = cons(c_y, vi2))
     => (DEQ(cons(c_x, D(vi3))) = DEQ(cons(c_y, vi2)))
    -> true
    Case.4.1: c_x = c_y
involves proving the following lemma(s):

lemma1.10.2.2.2.1.1
    (cons(c_x, D(vi3)) = cons(c_y, E(vi1)))
     => (DEQ(cons(c_x, D(vi3))) = DEQ(cons(c_y, E(vi1))))
    -> true
    [] Proved by normalization
lemma1.10.2.2.2.1.2
    (cons(c_x, D(vi3)) = cons(c_y, D(vi1)))
     => (DEQ(cons(c_x, D(vi3))) = DEQ(cons(c_y, D(vi1))))
    -> true
        which reduces to the equation
        ((D(vi1) = D(vi3)) <=> false) | (what(vi1) = what(vi3)) -> true
```

```
Proof of Lemma lemma1.10.2.2.2.1.2 suspended.

-> resume by case D(vi1::deq_rec)=D(vi3::deq_rec)

Case.5.1
    D(c_vi1) = D(c_vi3) == true
involves proving Lemma lemma1.10.2.2.2.1.2.1
    (cons(c_x, D(c_vi3)) = cons(c_y, D(c_vi1)))
     => (DEQ(cons(c_x, D(c_vi3))) = DEQ(cons(c_y, D(c_vi1))))
    -> true

The case system now contains 1 equation.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.5.1:
  D(c_vi1) = D(c_vi3) == true
to yield the following equations:
  Case.5.1.1: D(c_vi1) == D(c_vi3)

Ordered equation Case.5.1.1 into the rewrite rule:
  D(c_vi1) -> D(c_vi3)

The case system now contains 1 rewrite rule.

Lemma lemma1.10.2.2.2.1.2.1 in the proof by cases of Lemma lemma1.10.2.2.2.1.2
    (cons(c_x, D(c_vi3)) = cons(c_y, D(c_vi1)))
     => (DEQ(cons(c_x, D(c_vi3))) = DEQ(cons(c_y, D(c_vi1))))
    -> true
    Case.5.1: D(c_vi1) = D(c_vi3)
[] Proved by rewriting (with unreduced rules).

Case.5.2
    not(D(c_vi1) = D(c_vi3)) == true
involves proving Lemma lemma1.10.2.2.2.1.2.2
    (cons(c_x, D(c_vi3)) = cons(c_y, D(c_vi1)))
     => (DEQ(cons(c_x, D(c_vi3))) = DEQ(cons(c_y, D(c_vi1))))
    -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.5.2:
  (D(c_vi1) = D(c_vi3)) <=> false == true
to yield the following equations:
  Case.5.2.1: D(c_vi1) = D(c_vi3) == false

Ordered equation Case.5.2.1 into the rewrite rule:
  D(c_vi1) = D(c_vi3) -> false

The case system now contains 1 rewrite rule.

Lemma lemma1.10.2.2.2.1.2.2 in the proof by cases of Lemma lemma1.10.2.2.2.1.2
    (cons(c_x, D(c_vi3)) = cons(c_y, D(c_vi1)))
     => (DEQ(cons(c_x, D(c_vi3))) = DEQ(cons(c_y, D(c_vi1))))
    -> true
    Case.5.2: not(D(c_vi1) = D(c_vi3))
[] Proved by rewriting (with unreduced rules).

Lemma lemma1.10.2.2.2.1.2 for the basis step in the proof of Lemma
lemma1.10.2.2.2.1
    (cons(c_x, D(vi3)) = cons(c_y, D(vi1)))
     => (DEQ(cons(c_x, D(vi3))) = DEQ(cons(c_y, D(vi1))))
    -> true
[] Proved by cases
    (D(vi1) = D(vi3)) | not(D(vi1) = D(vi3))
```

```
The induction step in an inductive proof of Lemma lemma1.10.2.2.2.1 in the
proof by cases of Lemma lemma1.10.2.2.2
    (cons(c_x, D(vi3)) = cons(c_y, vi2))
     => (DEQ(cons(c_x, D(vi3))) = DEQ(cons(c_y, vi2)))
    -> true
    Case.4.1: c_x = c_y
is vacuous.

Lemma lemma1.10.2.2.2.1 in the proof by cases of Lemma lemma1.10.2.2.2
    (cons(c_x, D(vi3)) = cons(c_y, vi2))
     => (DEQ(cons(c_x, D(vi3))) = DEQ(cons(c_y, vi2)))
    -> true
    Case.4.1: c_x = c_y
[] Proved by induction over 'vi2::Ev' of sort 'Ev'.

Case.4.2
    not(c_x = c_y) == true
involves proving Lemma lemma1.10.2.2.2.2
    (cons(c_x, D(vi3)) = cons(c_y, vi2))
     => (DEQ(cons(c_x, D(vi3))) = DEQ(cons(c_y, vi2)))
    -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.4.2:
  (c_x = c_y) <=> false == true
to yield the following equations:
  Case.4.2.1: c_x = c_y == false

Ordered equation Case.4.2.1 into the rewrite rule:
  c_x = c_y -> false

The case system now contains 1 rewrite rule.

Lemma lemma1.10.2.2.2.2 in the proof by cases of Lemma lemma1.10.2.2.2
    (cons(c_x, D(vi3)) = cons(c_y, vi2))
     => (DEQ(cons(c_x, D(vi3))) = DEQ(cons(c_y, vi2)))
    -> true
    Case.4.2: not(c_x = c_y)
[] Proved by rewriting (with unreduced rules).

Lemma lemma1.10.2.2.2 for the basis step in the proof of Lemma lemma1.10.2.2
    (cons(c_x, D(vi3)) = cons(c_y, vi2))
     => (DEQ(cons(c_x, D(vi3))) = DEQ(cons(c_y, vi2)))
    -> true
[] Proved by cases
    (c_x = c_y) | not(c_x = c_y)

Lemma lemma1.10.2.2.1 for the basis step in the proof of Lemma lemma1.10.2.2
    (cons(c_x, E(vi3)) = cons(c_y, vi2))
     => (DEQ(cons(c_x, E(vi3))) = DEQ(cons(c_y, vi2)))
    -> true
is NOT provable using the current partially completed system.  It reduces to
the equation
    ((E(vi3) = vi2) <=> false)
     | ((c_x = c_y) <=> false)
     | (DEQ(c_x) = DEQ(cons(c_y, vi2)))
    -> true

Proof of Lemma lemma1.10.2.2.1 suspended.

-> resume by case c_x=c_y

Case.6.1
    c_x = c_y == true
```

```
involves proving Lemma lemma1.10.2.2.1.1
     (cons(c_x, E(vi3)) = cons(c_y, vi2))
      => (DEQ(cons(c_x, E(vi3))) = DEQ(cons(c_y, vi2)))
     -> true

The case system now contains 1 equation.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.6.1:
  c_x = c_y == true
to yield the following equations:
  Case.6.1.1: c_x == c_y

Ordered equation Case.6.1.1 into the rewrite rule:
  c_x -> c_y

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 149 rewrite rules, and 12 deduction rules.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.6.1:
  c_x = c_y == true
to yield the following equations:
  Case.6.1.2: c_x == c_y

Ordered equation Case.6.1.2 into the rewrite rule:
  c_x -> c_y

    Following 2 left-hand sides reduced:
    ((c_x = y) <=> false) | (DEQ(c_x) = DEQ(y)) -> true
      became equation Induct.14:
       ((c_y = y) <=> false) | (DEQ(c_x) = DEQ(y)) -> true
    ((c_y = cons(c_x, vi1)) <=> false) | (DEQ(c_y) = DEQ(cons(c_x, vi1)))
    -> true
      became equation Induct.15:
       ((c_y = cons(c_y, vi1)) <=> false) | (DEQ(c_y) = DEQ(cons(c_x, vi1)))
       -> true

Ordered equation Induct.14 into the rewrite rule:
  ((c_y = y) <=> false) | (DEQ(c_y) = DEQ(y)) -> true

The system now contains 149 rewrite rules and 12 deduction rules.

Lemma lemma1.10.2.2.1.1 in the proof by cases of Lemma lemma1.10.2.2.1
     (cons(c_x, E(vi3)) = cons(c_y, vi2))
      => (DEQ(cons(c_x, E(vi3))) = DEQ(cons(c_y, vi2)))
     -> true
    Case.6.1: c_x = c_y
is NOT provable using the current partially completed system.  It reduces to
the equation
     ((E(vi3) = vi2) <=> false) | (DEQ(c_y) = DEQ(cons(c_y, vi2))) -> true

Proof of Lemma lemma1.10.2.2.1.1 suspended.

->  resume by induction vi2 Ev

The basis step in an inductive proof of Lemma lemma1.10.2.2.1.1 in the proof by
cases of Lemma lemma1.10.2.2.1
     (cons(c_x, E(vi3)) = cons(c_y, vi2))
      => (DEQ(cons(c_x, E(vi3))) = DEQ(cons(c_y, vi2)))
     -> true
    Case.6.1: c_x = c_y
involves proving the following lemma(s):
```

```
lemma1.10.2.2.1.1.1
    (cons(c_x, E(vi3)) = cons(c_y, E(vi1)))
     => (DEQ(cons(c_x, E(vi3))) = DEQ(cons(c_y, E(vi1))))
    -> true
    [] Proved by normalization
lemma1.10.2.2.1.1.2
    (cons(c_x, E(vi3)) = cons(c_y, D(vi1)))
     => (DEQ(cons(c_x, E(vi3))) = DEQ(cons(c_y, D(vi1))))
    -> true
    [] Proved by normalization

The induction step in an inductive proof of Lemma lemma1.10.2.2.1.1 in the
proof by cases of Lemma lemma1.10.2.2.1
    (cons(c_x, E(vi3)) = cons(c_y, vi2))
     => (DEQ(cons(c_x, E(vi3))) = DEQ(cons(c_y, vi2)))
    -> true
    Case.6.1: c_x = c_y
is vacuous.

Lemma lemma1.10.2.2.1.1 in the proof by cases of Lemma lemma1.10.2.2.1
    (cons(c_x, E(vi3)) = cons(c_y, vi2))
     => (DEQ(cons(c_x, E(vi3))) = DEQ(cons(c_y, vi2)))
    -> true
    Case.6.1: c_x = c_y
[] Proved by induction over 'vi2::Ev' of sort 'Ev'.

Case.6.2
    not(c_x = c_y) == true
involves proving Lemma lemma1.10.2.2.1.2
    (cons(c_x, E(vi3)) = cons(c_y, vi2))
     => (DEQ(cons(c_x, E(vi3))) = DEQ(cons(c_y, vi2)))
    -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.6.2:
  (c_x = c_y) <=> false == true
to yield the following equations:
  Case.6.2.1: c_x = c_y == false

Ordered equation Case.6.2.1 into the rewrite rule:
  c_x = c_y -> false

The case system now contains 1 rewrite rule.

Lemma lemma1.10.2.2.1.2 in the proof by cases of Lemma lemma1.10.2.2.1
    (cons(c_x, E(vi3)) = cons(c_y, vi2))
     => (DEQ(cons(c_x, E(vi3))) = DEQ(cons(c_y, vi2)))
    -> true
    Case.6.2: not(c_x = c_y)
[] Proved by rewriting (with unreduced rules).

Lemma lemma1.10.2.2.1 for the basis step in the proof of Lemma lemma1.10.2.2
    (cons(c_x, E(vi3)) = cons(c_y, vi2))
     => (DEQ(cons(c_x, E(vi3))) = DEQ(cons(c_y, vi2)))
    -> true
[] Proved by cases
    (c_x = c_y) | not(c_x = c_y)

The induction step in an inductive proof of Lemma lemma1.10.2.2 for the
induction step in the proof of Lemma lemma1.10.2
    (cons(c_x, vi1) = cons(c_y, vi2))
     => (DEQ(cons(c_x, vi1)) = DEQ(cons(c_y, vi2)))
    -> true
is vacuous.
```

```
Lemma lemma1.10.2.2 for the induction step in the proof of Lemma lemma1.10.2
     (cons(c_x, vi1) = cons(c_y, vi2))
      => (DEQ(cons(c_x, vi1)) = DEQ(cons(c_y, vi2)))
     -> true
[] Proved by induction over 'vi1::Ev' of sort 'Ev'.

Lemma lemma1.10.2 for the induction step in the proof of Conjecture lemma1.10
     (cons(c_x, vi1) = y) => (DEQ(cons(c_x, vi1)) = DEQ(y)) -> true
[] Proved by induction over 'y' of sort 'H'.

Conjecture lemma1.10
     (x = y) => (DEQ(x) = DEQ(y)) -> true
[] Proved by induction over 'x' of sort 'H'.

The system now contains 1 equation, 147 rewrite rules, and 12 deduction rules.

Ordered equation lemma1.10 into the rewrite rule:
   ((x = y) <=> false) | (DEQ(x) = DEQ(y)) -> true

The system now contains 148 rewrite rules and 12 deduction rules.

->  prove (x=y)=>(ENQ(x)=ENQ(y)) by induction x H

The basis step in an inductive proof of Conjecture lemma1.11
     (x = y) => (ENQ(x) = ENQ(y)) -> true
involves proving the following lemma(s):

lemma1.11.1: (null = y) => (ENQ(null) = ENQ(y)) -> true
                  which reduces to the equation
                  ((null = y) <=> false) | (ENQ(y) = null) -> true

Proof of Lemma lemma1.11.1 suspended.

-> resume by induction y H

The basis step in an inductive proof of Lemma lemma1.11.1 for the basis step in
the proof of Conjecture lemma1.11
     (null = y) => (ENQ(null) = ENQ(y)) -> true
involves proving the following lemma(s):

lemma1.11.1.1: (null = null) => (ENQ(null) = ENQ(null)) -> true
               [] Proved by normalization

The induction step in an inductive proof of Lemma lemma1.11.1 for the basis
step in the proof of Conjecture lemma1.11
     (null = y) => (ENQ(null) = ENQ(y)) -> true
uses the following equation(s) for the induction hypothesis:

Induct.16: (c_y = null) => (ENQ(c_y) = ENQ(null)) -> true

The system now contains 1 equation, 148 rewrite rules, and 12 deduction rules.

Ordered equation Induct.16 into the rewrite rule:
   ((c_y = null) <=> false) | (ENQ(c_y) = null) -> true

The system now contains 149 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma1.11.1.2: (cons(c_y, vi1) = null) => (ENQ(cons(c_y, vi1)) = ENQ(null))
               -> true
               [] Proved by normalization

Lemma lemma1.11.1 for the basis step in the proof of Conjecture lemma1.11
     (null = y) => (ENQ(null) = ENQ(y)) -> true
[] Proved by induction over 'y' of sort 'H'.

The induction step in an inductive proof of Conjecture lemma1.11
     (x = y) => (ENQ(x) = ENQ(y)) -> true
```

```
uses the following equation(s) for the induction hypothesis:

Induct.17: (c_x = y) => (ENQ(c_x) = ENQ(y)) -> true

The system now contains 1 equation, 148 rewrite rules, and 12 deduction rules.

Ordered equation Induct.17 into the rewrite rule:
  ((c_x = y) <=> false) | (ENQ(c_x) = ENQ(y)) -> true

The system now contains 149 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma1.11.2: (cons(c_x, vi1) = y) => (ENQ(cons(c_x, vi1)) = ENQ(y)) -> true
                 which reduces to the equation
                 ((cons(c_x, vi1) = y) <=> false)
                  | (ENQ(cons(c_x, vi1)) = ENQ(y))
                 -> true

Proof of Lemma lemma1.11.2 suspended.

->  resume by induction y H

The basis step in an inductive proof of Lemma lemma1.11.2 for the induction
step in the proof of Conjecture lemma1.11
    (cons(c_x, vi1) = y) => (ENQ(cons(c_x, vi1)) = ENQ(y)) -> true
involves proving the following lemma(s):

lemma1.11.2.1: (cons(c_x, vi1) = null) => (ENQ(cons(c_x, vi1)) = ENQ(null))
               -> true
               [] Proved by normalization

The induction step in an inductive proof of Lemma lemma1.11.2 for the induction
step in the proof of Conjecture lemma1.11
    (cons(c_x, vi1) = y) => (ENQ(cons(c_x, vi1)) = ENQ(y)) -> true
uses the following equation(s) for the induction hypothesis:

Induct.18: (c_y = cons(c_x, vi1)) => (ENQ(c_y) = ENQ(cons(c_x, vi1))) -> true

The system now contains 1 equation, 149 rewrite rules, and 12 deduction rules.

Ordered equation Induct.18 into the rewrite rule:
  ((c_y = cons(c_x, vi1)) <=> false) | (ENQ(c_y) = ENQ(cons(c_x, vi1))) -> true

The system now contains 150 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma1.11.2.2: (cons(c_x, vi1) = cons(c_y, vi2))
                => (ENQ(cons(c_x, vi1)) = ENQ(cons(c_y, vi2)))
               -> true
                   which reduces to the equation
                   ((c_x = c_y) <=> false)
                    | ((vi1 = vi2) <=> false)
                    | (ENQ(cons(c_x, vi1)) = ENQ(cons(c_y, vi2)))
                   -> true

Proof of Lemma lemma1.11.2.2 suspended.

-> resume by induction vi1 Ev

The basis step in an inductive proof of Lemma lemma1.11.2.2 for the induction
step in the proof of Lemma lemma1.11.2
    (cons(c_x, vi1) = cons(c_y, vi2))
     => (ENQ(cons(c_x, vi1)) = ENQ(cons(c_y, vi2)))
    -> true
involves proving the following lemma(s):

lemma1.11.2.2.1: (cons(c_x, E(vi3)) = cons(c_y, vi2))
```

```
                   => (ENQ(cons(c_x, E(vi3))) = ENQ(cons(c_y, vi2)))
                  -> true
                      which reduces to the equation
                      ((E(vi3) = vi2) <=> false)
                       | ((c_x = c_y) <=> false)
                       | (ENQ(cons(c_y, vi2)) = cons(ENQ(c_x), element(vi3)))
                      -> true
lemma1.11.2.2.2: (cons(c_x, D(vi3)) = cons(c_y, vi2))
                   => (ENQ(cons(c_x, D(vi3))) = ENQ(cons(c_y, vi2)))
                  -> true
                      which reduces to the equation
                      ((D(vi3) = vi2) <=> false)
                       | ((c_x = c_y) <=> false)
                       | (ENQ(c_x) = ENQ(cons(c_y, vi2)))
                      -> true

Proof of Lemma lemma1.11.2.2.2 suspended.

-> resume by case c_x=c_y

Case.7.1
    c_x = c_y == true
involves proving Lemma lemma1.11.2.2.2.1
    (cons(c_x, D(vi3)) = cons(c_y, vi2))
     => (ENQ(cons(c_x, D(vi3))) = ENQ(cons(c_y, vi2)))
    -> true

The case system now contains 1 equation.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.7.1:
  c_x = c_y == true
to yield the following equations:
  Case.7.1.1: c_x == c_y

Ordered equation Case.7.1.1 into the rewrite rule:
  c_x -> c_y

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 150 rewrite rules, and 12 deduction rules.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.7.1:
  c_x = c_y == true
to yield the following equations:
  Case.7.1.2: c_x == c_y

Ordered equation Case.7.1.2 into the rewrite rule:
  c_x -> c_y

    Following 2 left-hand sides reduced:
    ((c_x = y) <=> false) | (ENQ(c_x) = ENQ(y)) -> true
      became equation Induct.17:
      ((c_y = y) <=> false) | (ENQ(c_x) = ENQ(y)) -> true
    ((c_y = cons(c_x, vi1)) <=> false) | (ENQ(c_y) = ENQ(cons(c_x, vi1)))
    -> true
      became equation Induct.18:
      ((c_y = cons(c_y, vi1)) <=> false) | (ENQ(c_y) = ENQ(cons(c_x, vi1)))
      -> true

Ordered equation Induct.17 into the rewrite rule:
   ((c_y = y) <=> false) | (ENQ(c_y) = ENQ(y)) -> true

The system now contains 150 rewrite rules and 12 deduction rules.
```

```
Lemma lemma1.11.2.2.2.1 in the proof by cases of Lemma lemma1.11.2.2.2
    (cons(c_x, D(vi3)) = cons(c_y, vi2))
     => (ENQ(cons(c_x, D(vi3))) = ENQ(cons(c_y, vi2)))
    -> true
    Case.7.1: c_x = c_y
is NOT provable using the current partially completed system.  It reduces to
the equation
    ((D(vi3) = vi2) <=> false) | (ENQ(c_y) = ENQ(cons(c_y, vi2))) -> true

Proof of Lemma lemma1.11.2.2.2.1 suspended.

-> resume by induction vi2 Ev

The basis step in an inductive proof of Lemma lemma1.11.2.2.2.1 in the proof by
cases of Lemma lemma1.11.2.2.2
    (cons(c_x, D(vi3)) = cons(c_y, vi2))
     => (ENQ(cons(c_x, D(vi3))) = ENQ(cons(c_y, vi2)))
    -> true
    Case.7.1: c_x = c_y
involves proving the following lemma(s):

lemma1.11.2.2.2.1.1
    (cons(c_x, D(vi3)) = cons(c_y, E(vi1)))
     => (ENQ(cons(c_x, D(vi3))) = ENQ(cons(c_y, E(vi1))))
    -> true
    [] Proved by normalization
lemma1.11.2.2.2.1.2
    (cons(c_x, D(vi3)) = cons(c_y, D(vi1)))
     => (ENQ(cons(c_x, D(vi3))) = ENQ(cons(c_y, D(vi1))))
    -> true
    [] Proved by normalization

The induction step in an inductive proof of Lemma lemma1.11.2.2.2.1 in the
proof by cases of Lemma lemma1.11.2.2.2
    (cons(c_x, D(vi3)) = cons(c_y, vi2))
     => (ENQ(cons(c_x, D(vi3))) = ENQ(cons(c_y, vi2)))
    -> true
    Case.7.1: c_x = c_y
is vacuous.

Lemma lemma1.11.2.2.2.1 in the proof by cases of Lemma lemma1.11.2.2.2
    (cons(c_x, D(vi3)) = cons(c_y, vi2))
     => (ENQ(cons(c_x, D(vi3))) = ENQ(cons(c_y, vi2)))
    -> true
    Case.7.1: c_x = c_y
[] Proved by induction over 'vi2::Ev' of sort 'Ev'.

Case.7.2
    not(c_x = c_y) == true
involves proving Lemma lemma1.11.2.2.2.2
    (cons(c_x, D(vi3)) = cons(c_y, vi2))
     => (ENQ(cons(c_x, D(vi3))) = ENQ(cons(c_y, vi2)))
    -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.7.2:
  (c_x = c_y) <=> false == true
to yield the following equations:
  Case.7.2.1: c_x = c_y == false

Ordered equation Case.7.2.1 into the rewrite rule:
  c_x = c_y -> false

The case system now contains 1 rewrite rule.
```

```
Lemma lemma1.11.2.2.2.2 in the proof by cases of Lemma lemma1.11.2.2.2
     (cons(c_x, D(vi3)) = cons(c_y, vi2))
      => (ENQ(cons(c_x, D(vi3))) = ENQ(cons(c_y, vi2)))
     -> true
     Case.7.2: not(c_x = c_y)
[] Proved by rewriting (with unreduced rules).

Lemma lemma1.11.2.2.2 for the basis step in the proof of Lemma lemma1.11.2.2
     (cons(c_x, D(vi3)) = cons(c_y, vi2))
      => (ENQ(cons(c_x, D(vi3))) = ENQ(cons(c_y, vi2)))
     -> true
[] Proved by cases
     (c_x = c_y) | not(c_x = c_y)

Lemma lemma1.11.2.2.1 for the basis step in the proof of Lemma lemma1.11.2.2
     (cons(c_x, E(vi3)) = cons(c_y, vi2))
      => (ENQ(cons(c_x, E(vi3))) = ENQ(cons(c_y, vi2)))
     -> true
is NOT provable using the current partially completed system.  It reduces to
the equation
     ((E(vi3) = vi2) <=> false)
      | ((c_x = c_y) <=> false)
      | (ENQ(cons(c_y, vi2)) = cons(ENQ(c_x), element(vi3)))
     -> true

Proof of Lemma lemma1.11.2.2.1 suspended.

-> resume by case c_x=c_y

Case.8.1
     c_x = c_y == true
involves proving Lemma lemma1.11.2.2.1.1
     (cons(c_x, E(vi3)) = cons(c_y, vi2))
      => (ENQ(cons(c_x, E(vi3))) = ENQ(cons(c_y, vi2)))
     -> true

The case system now contains 1 equation.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.8.1:
  c_x = c_y == true
to yield the following equations:
  Case.8.1.1: c_x == c_y

Ordered equation Case.8.1.1 into the rewrite rule:
  c_x -> c_y

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 150 rewrite rules, and 12 deduction rules.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.8.1:
  c_x = c_y == true
to yield the following equations:
  Case.8.1.2: c_x == c_y

Ordered equation Case.8.1.2 into the rewrite rule:
  c_x -> c_y

     Following 2 left-hand sides reduced:
     ((c_x = y) <=> false) | (ENQ(c_x) = ENQ(y)) -> true
       became equation Induct.17:
       ((c_y = y) <=> false) | (ENQ(c_x) = ENQ(y)) -> true
```

```
    ((c_y = cons(c_x, vi1)) <=> false) | (ENQ(c_y) = ENQ(cons(c_x, vi1)))
    -> true
      became equation Induct.18:
      ((c_y = cons(c_y, vi1)) <=> false) | (ENQ(c_y) = ENQ(cons(c_x, vi1)))
      -> true

Ordered equation Induct.17 into the rewrite rule:
  ((c_y = y) <=> false) | (ENQ(c_y) = ENQ(y)) -> true

The system now contains 150 rewrite rules and 12 deduction rules.

Lemma lemma1.11.2.2.1.1 in the proof by cases of Lemma lemma1.11.2.2.1
    (cons(c_x, E(vi3)) = cons(c_y, vi2))
     => (ENQ(cons(c_x, E(vi3))) = ENQ(cons(c_y, vi2)))
    -> true
    Case.8.1: c_x = c_y
is NOT provable using the current partially completed system.  It reduces to
the equation
    ((E(vi3) = vi2) <=> false)
     | (ENQ(cons(c_y, vi2)) = cons(ENQ(c_y), element(vi3)))
    -> true

Proof of Lemma lemma1.11.2.2.1.1 suspended.

-> resume by induction vi2 Ev

The basis step in an inductive proof of Lemma lemma1.11.2.2.1.1 in the proof by
cases of Lemma lemma1.11.2.2.1
    (cons(c_x, E(vi3)) = cons(c_y, vi2))
     => (ENQ(cons(c_x, E(vi3))) = ENQ(cons(c_y, vi2)))
    -> true
    Case.8.1: c_x = c_y
involves proving the following lemma(s):

lemma1.11.2.2.1.1.1
    (cons(c_x, E(vi3)) = cons(c_y, E(vi1)))
     => (ENQ(cons(c_x, E(vi3))) = ENQ(cons(c_y, E(vi1))))
    -> true
        which reduces to the equation
        ((E(vi1) = E(vi3)) <=> false) | (element(vi1) = element(vi3)) -> true
lemma1.11.2.2.1.1.2
    (cons(c_x, E(vi3)) = cons(c_y, D(vi1)))
     => (ENQ(cons(c_x, E(vi3))) = ENQ(cons(c_y, D(vi1))))
    -> true
    [] Proved by normalization

Proof of Lemma lemma1.11.2.2.1.1.1 suspended.


-> resume by case E(vi1::enq_rec)=E(vi3::enq_rec)

Case.9.1
    E(c_vi1) = E(c_vi3) == true
involves proving Lemma lemma1.11.2.2.1.1.1.1
    (cons(c_x, E(c_vi3)) = cons(c_y, E(c_vi1)))
     => (ENQ(cons(c_x, E(c_vi3))) = ENQ(cons(c_y, E(c_vi1))))
    -> true

The case system now contains 1 equation.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.9.1:
  E(c_vi1) = E(c_vi3) == true
to yield the following equations:
  Case.9.1.1: E(c_vi1) == E(c_vi3)

Ordered equation Case.9.1.1 into the rewrite rule:
```

```
  E(c_vi1) -> E(c_vi3)

The case system now contains 1 rewrite rule.

Lemma lemma1.11.2.2.1.1.1.1 in the proof by cases of Lemma lemma1.11.2.2.1.1.1
    (cons(c_x, E(c_vi3)) = cons(c_y, E(c_vi1)))
     => (ENQ(cons(c_x, E(c_vi3))) = ENQ(cons(c_y, E(c_vi1))))
    -> true
    Case.9.1: E(c_vi1) = E(c_vi3)
[] Proved by rewriting (with unreduced rules).

Case.9.2
    not(E(c_vi1) = E(c_vi3)) == true
involves proving Lemma lemma1.11.2.2.1.1.1.2
    (cons(c_x, E(c_vi3)) = cons(c_y, E(c_vi1)))
     => (ENQ(cons(c_x, E(c_vi3))) = ENQ(cons(c_y, E(c_vi1))))
    -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.9.2:
  (E(c_vi1) = E(c_vi3)) <=> false == true
to yield the following equations:
  Case.9.2.1: E(c_vi1) = E(c_vi3) == false

Ordered equation Case.9.2.1 into the rewrite rule:
  E(c_vi1) = E(c_vi3) -> false

The case system now contains 1 rewrite rule.

Lemma lemma1.11.2.2.1.1.1.2 in the proof by cases of Lemma lemma1.11.2.2.1.1.1
    (cons(c_x, E(c_vi3)) = cons(c_y, E(c_vi1)))
     => (ENQ(cons(c_x, E(c_vi3))) = ENQ(cons(c_y, E(c_vi1))))
    -> true
    Case.9.2: not(E(c_vi1) = E(c_vi3))
[] Proved by rewriting (with unreduced rules).

Lemma lemma1.11.2.2.1.1.1 for the basis step in the proof of Lemma
lemma1.11.2.2.1.1
    (cons(c_x, E(vi3)) = cons(c_y, E(vi1)))
     => (ENQ(cons(c_x, E(vi3))) = ENQ(cons(c_y, E(vi1))))
    -> true
[] Proved by cases
    (E(vi1) = E(vi3)) | not(E(vi1) = E(vi3))

The induction step in an inductive proof of Lemma lemma1.11.2.2.1.1 in the
proof by cases of Lemma lemma1.11.2.2.1
    (cons(c_x, E(vi3)) = cons(c_y, vi2))
     => (ENQ(cons(c_x, E(vi3))) = ENQ(cons(c_y, vi2)))
    -> true
    Case.8.1: c_x = c_y
is vacuous.

Lemma lemma1.11.2.2.1.1 in the proof by cases of Lemma lemma1.11.2.2.1
    (cons(c_x, E(vi3)) = cons(c_y, vi2))
     => (ENQ(cons(c_x, E(vi3))) = ENQ(cons(c_y, vi2)))
    -> true
    Case.8.1: c_x = c_y
[] Proved by induction over `vi2::Ev' of sort `Ev'.

Case.8.2
    not(c_x = c_y) == true
involves proving Lemma lemma1.11.2.2.1.2
    (cons(c_x, E(vi3)) = cons(c_y, vi2))
     => (ENQ(cons(c_x, E(vi3))) = ENQ(cons(c_y, vi2)))
    -> true
```

```
The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.8.2:
  (c_x = c_y) <=> false == true
to yield the following equations:
  Case.8.2.1: c_x = c_y == false

Ordered equation Case.8.2.1 into the rewrite rule:
  c_x = c_y -> false

The case system now contains 1 rewrite rule.

Lemma lemma1.11.2.2.1.2 in the proof by cases of Lemma lemma1.11.2.2.1
    (cons(c_x, E(vi3)) = cons(c_y, vi2))
     => (ENQ(cons(c_x, E(vi3))) = ENQ(cons(c_y, vi2)))
    -> true
    Case.8.2: not(c_x = c_y)
[] Proved by rewriting (with unreduced rules).

Lemma lemma1.11.2.2.1 for the basis step in the proof of Lemma lemma1.11.2.2
    (cons(c_x, E(vi3)) = cons(c_y, vi2))
     => (ENQ(cons(c_x, E(vi3))) = ENQ(cons(c_y, vi2)))
    -> true
[] Proved by cases
    (c_x = c_y) | not(c_x = c_y)

The induction step in an inductive proof of Lemma lemma1.11.2.2 for the
induction step in the proof of Lemma lemma1.11.2
    (cons(c_x, vi1) = cons(c_y, vi2))
     => (ENQ(cons(c_x, vi1)) = ENQ(cons(c_y, vi2)))
    -> true
is vacuous.

Lemma lemma1.11.2.2 for the induction step in the proof of Lemma lemma1.11.2
    (cons(c_x, vi1) = cons(c_y, vi2))
     => (ENQ(cons(c_x, vi1)) = ENQ(cons(c_y, vi2)))
    -> true
[] Proved by induction over 'vi1::Ev' of sort 'Ev'.

Lemma lemma1.11.2 for the induction step in the proof of Conjecture lemma1.11
    (cons(c_x, vi1) = y) => (ENQ(cons(c_x, vi1)) = ENQ(y)) -> true
[] Proved by induction over 'y' of sort 'H'.

Conjecture lemma1.11
    (x = y) => (ENQ(x) = ENQ(y)) -> true
[] Proved by induction over 'x' of sort 'H'.

The system now contains 1 equation, 148 rewrite rules, and 12 deduction rules.

Ordered equation lemma1.11 into the rewrite rule:
  ((x = y) <=> false) | (ENQ(x) = ENQ(y)) -> true

The system now contains 149 rewrite rules and 12 deduction rules.

->  prove in_state(x,init)=>(x=null:->H) by induction x H

The basis step in an inductive proof of Conjecture lemma1.12
    in_state(x, init) => (null = x) -> true
involves proving the following lemma(s):

lemma1.12.1: in_state(null, init) => (null = null) -> true
             [] Proved by normalization

The induction step in an inductive proof of Conjecture lemma1.12
    in_state(x, init) => (null = x) -> true
```

```
uses the following equation(s) for the induction hypothesis:

Induct.20: in_state(c_x, init) => (c_x = null) -> true

The system now contains 1 equation, 149 rewrite rules, and 12 deduction rules.

Ordered equation Induct.20 into the rewrite rule:
  (false <=> in_state(c_x, init)) | (c_x = null) -> true

The system now contains 150 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma1.12.2: in_state(cons(c_x, vi1), init) => (cons(c_x, vi1) = null) -> true
                 which reduces to the equation
                 false <=> in_state(cons(c_x, vi1), init) -> true

Proof of Lemma lemma1.12.2 suspended.

-> resume by induction vi1 Ev

The basis step in an inductive proof of Lemma lemma1.12.2 for the induction
step in the proof of Conjecture lemma1.12
    in_state(cons(c_x, vi1), init) => (cons(c_x, vi1) = null) -> true
involves proving the following lemma(s):

lemma1.12.2.1: in_state(cons(c_x, E(vi2)), init) => (cons(c_x, E(vi2)) = null)
               -> true
                   which reduces to the equation
                   false <=> in_state(cons(c_x, E(vi2)), init) -> true
lemma1.12.2.2: in_state(cons(c_x, D(vi2)), init) => (cons(c_x, D(vi2)) = null)
               -> true
                   which reduces to the equation
                   false <=> in_state(cons(c_x, D(vi2)), init) -> true

Proof of Lemma lemma1.12.2.2 suspended.

-> resume by case in_state(cons(c_x,D(vi2::deq_rec)),init)

Case.11.1
    in_state(cons(c_x, D(c_vi2)), init) == true
involves proving Lemma lemma1.12.2.2.1
    in_state(cons(c_x, D(c_vi2)), init) => (cons(c_x, D(c_vi2)) = null) -> true

The case system now contains 1 equation.

Ordered equation Case.11.1 into the rewrite rule:
  in_state(cons(c_x, D(c_vi2)), init) -> true

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 150 rewrite rules, and 12 deduction rules.

Ordered equation Case.11.1 into the rewrite rule:
  in_state(cons(c_x, D(c_vi2)), init) -> true

The system now contains 151 rewrite rules and 12 deduction rules.

Lemma lemma1.12.2.2.1 in the proof by cases of Lemma lemma1.12.2.2
    in_state(cons(c_x, D(c_vi2)), init) => (cons(c_x, D(c_vi2)) = null) -> true
    Case.11.1: in_state(cons(c_x, D(c_vi2)), init)
is NOT provable using the current partially completed system.  It reduces to
the equation
    false -> true

Proof of Lemma lemma1.12.2.2.1 suspended.

-> crit case with Abstraction.3
```

```
Critical pairs between rule Case.11.1:
  in_state(cons(c_x, D(c_vi2)), init) -> true
and rule Abstraction.3:
  (in_stack(vd, deqd(xst)) & in_state(xh, xst))
   | (false <=> in_state(cons(xh, D(vd)), xst))
  -> true
  are as follows:
    false <=> in_state(cons(cons(c_x, D(c_vi2)), D(vd)), init) == true
    false == true

The system now contains 1 equation, 151 rewrite rules, and 12 deduction rules.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation lemma1.28:
  false <=> in_state(cons(cons(c_x, D(c_vi2)), D(vd)), init) == true
to yield the following equations:
  lemma1.28.1: false == in_state(cons(cons(c_x, D(c_vi2)), D(vd)), init)

Ordered equation lemma1.28.1 into the rewrite rule:
  in_state(cons(cons(c_x, D(c_vi2)), D(vd)), init) -> false

The system now contains 152 rewrite rules and 12 deduction rules.

The system now contains 1 equation, 152 rewrite rules, and 12 deduction rules.

Equation lemma1.29
    false == true
is inconsistent.

Lemma lemma1.12.2.2.1 in the proof by cases of Lemma lemma1.12.2.2
    in_state(cons(c_x, D(c_vi2)), init) => (cons(c_x, D(c_vi2)) = null) -> true
    Case.11.1: in_state(cons(c_x, D(c_vi2)), init)
[] Proved by impossible case.

Case.11.2
    not(in_state(cons(c_x, D(c_vi2)), init)) == true
involves proving Lemma lemma1.12.2.2.2
    in_state(cons(c_x, D(c_vi2)), init) => (cons(c_x, D(c_vi2)) = null) -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.11.2:
  false <=> in_state(cons(c_x, D(c_vi2)), init) == true
to yield the following equations:
  Case.11.2.1: false == in_state(cons(c_x, D(c_vi2)), init)

Ordered equation Case.11.2.1 into the rewrite rule:
  in_state(cons(c_x, D(c_vi2)), init) -> false

The case system now contains 1 rewrite rule.

Lemma lemma1.12.2.2.2 in the proof by cases of Lemma lemma1.12.2.2
    in_state(cons(c_x, D(c_vi2)), init) => (cons(c_x, D(c_vi2)) = null) -> true
    Case.11.2: not(in_state(cons(c_x, D(c_vi2)), init))
[] Proved by rewriting (with unreduced rules).

Lemma lemma1.12.2.2 for the basis step in the proof of Lemma lemma1.12.2
    in_state(cons(c_x, D(vi2)), init) => (cons(c_x, D(vi2)) = null) -> true
[] Proved by cases
    in_state(cons(c_x, D(vi2)), init) | not(in_state(cons(c_x, D(vi2)), init))

Lemma lemma1.12.2.1 for the basis step in the proof of Lemma lemma1.12.2
    in_state(cons(c_x, E(vi2)), init) => (cons(c_x, E(vi2)) = null) -> true
is NOT provable using the current partially completed system.  It reduces to
```

```
the equation
    false <=> in_state(cons(c_x, E(vi2)), init) -> true

Proof of Lemma lemma1.12.2.1 suspended.

Critical-pair computation abandoned because a theorem has been proved.

Computed 2 new critical pairs.  Added 2 of them to the system.

-> resume by case in_state(cons(c_x,E(vi2::enq_rec)),init)

Case.12.1
    in_state(cons(c_x, E(c_vi2)), init) == true
involves proving Lemma lemma1.12.2.1.1
    in_state(cons(c_x, E(c_vi2)), init) => (cons(c_x, E(c_vi2)) = null) -> true

The case system now contains 1 equation.

Ordered equation Case.12.1 into the rewrite rule:
  in_state(cons(c_x, E(c_vi2)), init) -> true

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 150 rewrite rules, and 12 deduction rules.

Ordered equation Case.12.1 into the rewrite rule:
  in_state(cons(c_x, E(c_vi2)), init) -> true

The system now contains 151 rewrite rules and 12 deduction rules.

Lemma lemma1.12.2.1.1 in the proof by cases of Lemma lemma1.12.2.1
    in_state(cons(c_x, E(c_vi2)), init) => (cons(c_x, E(c_vi2)) = null) -> true
    Case.12.1: in_state(cons(c_x, E(c_vi2)), init)
is NOT provable using the current partially completed system.  It reduces to
the equation
    false -> true

Proof of Lemma lemma1.12.2.1.1 suspended.

-> crit case with Abstraction.2

Critical pairs between rule Case.12.1:
  in_state(cons(c_x, E(c_vi2)), init) -> true
and rule Abstraction.2:
  (in(ue, enqd(xst)) & in_state(xh, xst))
   | (false <=> in_state(cons(xh, E(ue)), xst))
  -> true
  are as follows:
    false <=> in_state(cons(cons(c_x, E(c_vi2)), E(ue)), init) == true
    false == true

The system now contains 1 equation, 151 rewrite rules, and 12 deduction rules.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation lemma1.30:
  false <=> in_state(cons(cons(c_x, E(c_vi2)), E(ue)), init) == true
to yield the following equations:
  lemma1.30.1: false == in_state(cons(cons(c_x, E(c_vi2)), E(ue)), init)

Ordered equation lemma1.30.1 into the rewrite rule:
  in_state(cons(cons(c_x, E(c_vi2)), E(ue)), init) -> false

The system now contains 152 rewrite rules and 12 deduction rules.

The system now contains 1 equation, 152 rewrite rules, and 12 deduction rules.

Equation lemma1.31
```

```
    false == true
is inconsistent.

Lemma lemma1.12.2.1.1 in the proof by cases of Lemma lemma1.12.2.1
    in_state(cons(c_x, E(c_vi2)), init) => (cons(c_x, E(c_vi2)) = null) -> true
    Case.12.1: in_state(cons(c_x, E(c_vi2)), init)
[] Proved by impossible case.

Case.12.2
    not(in_state(cons(c_x, E(c_vi2)), init)) == true
involves proving Lemma lemma1.12.2.1.2
    in_state(cons(c_x, E(c_vi2)), init) => (cons(c_x, E(c_vi2)) = null) -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.12.2:
  false <=> in_state(cons(c_x, E(c_vi2)), init) == true
to yield the following equations:
  Case.12.2.1: false == in_state(cons(c_x, E(c_vi2)), init)

Ordered equation Case.12.2.1 into the rewrite rule:
  in_state(cons(c_x, E(c_vi2)), init) -> false

The case system now contains 1 rewrite rule.

Lemma lemma1.12.2.1.2 in the proof by cases of Lemma lemma1.12.2.1
    in_state(cons(c_x, E(c_vi2)), init) => (cons(c_x, E(c_vi2)) = null) -> true
    Case.12.2: not(in_state(cons(c_x, E(c_vi2)), init))
[] Proved by rewriting (with unreduced rules).

Lemma lemma1.12.2.1 for the basis step in the proof of Lemma lemma1.12.2
    in_state(cons(c_x, E(vi2)), init) => (cons(c_x, E(vi2)) = null) -> true
[] Proved by cases
    in_state(cons(c_x, E(vi2)), init) | not(in_state(cons(c_x, E(vi2)), init))

The induction step in an inductive proof of Lemma lemma1.12.2 for the induction
step in the proof of Conjecture lemma1.12
    in_state(cons(c_x, vi1), init) => (cons(c_x, vi1) = null) -> true
is vacuous.

Lemma lemma1.12.2 for the induction step in the proof of Conjecture lemma1.12
    in_state(cons(c_x, vi1), init) => (cons(c_x, vi1) = null) -> true
[] Proved by induction over 'vi1::Ev' of sort 'Ev'.

Conjecture lemma1.12
    in_state(x, init) => (null = x) -> true
[] Proved by induction over 'x' of sort 'H'.

The system now contains 1 equation, 149 rewrite rules, and 12 deduction rules.

Ordered equation lemma1.12 into the rewrite rule:
   (false <=> in_state(x, init)) | (null = x) -> true

The system now contains 150 rewrite rules and 12 deduction rules.

Critical-pair computation abandoned because a theorem has been proved.

Computed 2 new critical pairs.  Added 2 of them to the system.

-> prove prefix(cons:Seq,EL->Seq(x,z),y)=>prefix(x,y) by induction x
Please enter a sort for the induction: Seq

The basis step in an inductive proof of Conjecture lemma1.13
    prefix(cons(x, z), y) => prefix(x, y) -> true
involves proving the following lemma(s):
```

```
lemma1.13.1: prefix(cons(null, z), y) => prefix(null, y) -> true
             [] Proved by normalization

The induction step in an inductive proof of Conjecture lemma1.13
    prefix(cons(x, z), y) => prefix(x, y) -> true
uses the following equation(s) for the induction hypothesis:

Induct.24: prefix(cons(c_x, z), y) => prefix(c_x, y) -> true

The system now contains 1 equation, 151 rewrite rules, and 12 deduction rules.

Ordered equation Induct.24 into the rewrite rule:
  (false <=> prefix(cons(c_x, z), y)) | prefix(c_x, y) -> true

The system now contains 152 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma1.13.2: prefix(cons(cons(c_x, vi1), z), y) => prefix(cons(c_x, vi1), y)
             -> true
                 which reduces to the equation
                 (false <=> prefix(cons(cons(c_x, vi1), z), y))
                  | prefix(cons(c_x, vi1), y)
                 -> true

Proof of Lemma lemma1.13.2 suspended.

-> resume by induction y Seq

The basis step in an inductive proof of Lemma lemma1.13.2 for the induction
step in the proof of Conjecture lemma1.13
    prefix(cons(cons(c_x, vi1), z), y) => prefix(cons(c_x, vi1), y) -> true
involves proving the following lemma(s):

lemma1.13.2.1: prefix(cons(cons(c_x, vi1), z), null)
                => prefix(cons(c_x, vi1), null)
               -> true
               [] Proved by normalization

The induction step in an inductive proof of Lemma lemma1.13.2 for the induction
step in the proof of Conjecture lemma1.13
    prefix(cons(cons(c_x, vi1), z), y) => prefix(cons(c_x, vi1), y) -> true
uses the following equation(s) for the induction hypothesis:

Induct.25: prefix(cons(cons(c_x, vi1), z), c_y) => prefix(cons(c_x, vi1), c_y)
           -> true

The system now contains 1 equation, 152 rewrite rules, and 12 deduction rules.

Ordered equation Induct.25 into the rewrite rule:
  (false <=> prefix(cons(cons(c_x, vi1), z), c_y))
   | prefix(cons(c_x, vi1), c_y)
  -> true

The system now contains 153 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma1.13.2.2: prefix(cons(cons(c_x, vi1), z), cons(c_y, vi2))
                => prefix(cons(c_x, vi1), cons(c_y, vi2))
               -> true
                   which reduces to the equation
                   ((false <=> prefix(cons(cons(c_x, vi1), z), c_y))
                     & (((c_y = cons(c_x, vi1)) <=> false)
                         | ((vi2 = z) <=> false)))

                    | ((c_x = c_y) & (vi1 = vi2))
                    | prefix(cons(c_x, vi1), c_y)
                   -> true
```

```
Proof of Lemma lemma1.13.2.2 suspended.

-> resume by case prefix(cons(cons(c_x, vi1), z), c_y)

Case.13.1
    prefix(cons(cons(c_x, c_vi1), c_z), c_y) == true
involves proving Lemma lemma1.13.2.2.1
    prefix(cons(cons(c_x, c_vi1), c_z), cons(c_y, vi2))
     => prefix(cons(c_x, c_vi1), cons(c_y, vi2))
    -> true

The case system now contains 1 equation.

Ordered equation Case.13.1 into the rewrite rule:
  prefix(cons(cons(c_x, c_vi1), c_z), c_y) -> true

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 153 rewrite rules, and 12 deduction rules.

Ordered equation Case.13.1 into the rewrite rule:
  prefix(cons(cons(c_x, c_vi1), c_z), c_y) -> true

The system now contains 154 rewrite rules and 12 deduction rules.

Lemma lemma1.13.2.2.1 in the proof by cases of Lemma lemma1.13.2.2
    prefix(cons(cons(c_x, c_vi1), c_z), cons(c_y, vi2))
     => prefix(cons(c_x, c_vi1), cons(c_y, vi2))
    -> true
    Case.13.1: prefix(cons(cons(c_x, c_vi1), c_z), c_y)
is NOT provable using the current partially completed system.  It reduces to
the equation
    ((c_vi1 = vi2) & (c_x = c_y)) | prefix(cons(c_x, c_vi1), c_y) -> true

Proof of Lemma lemma1.13.2.2.1 suspended.

-> crit case with induct

Critical pairs between rule Case.13.1:
  prefix(cons(cons(c_x, c_vi1), c_z), c_y) -> true
and rule Induct.25:
  (false <=> prefix(cons(cons(c_x, vi1), z), c_y))
   | prefix(cons(c_x, vi1), c_y)
  -> true
  are as follows:
    prefix(cons(c_x, c_vi1), c_y) == true

The system now contains 1 equation, 154 rewrite rules, and 12 deduction rules.

Ordered equation lemma1.35 into the rewrite rule:
  prefix(cons(c_x, c_vi1), c_y) -> true

The system now contains 155 rewrite rules and 12 deduction rules.

Lemma lemma1.13.2.2.1 in the proof by cases of Lemma lemma1.13.2.2
    prefix(cons(cons(c_x, c_vi1), c_z), cons(c_y, vi2))
     => prefix(cons(c_x, c_vi1), cons(c_y, vi2))
    -> true
    Case.13.1: prefix(cons(cons(c_x, c_vi1), c_z), c_y)
[] Proved by rewriting.

Case.13.2
    not(prefix(cons(cons(c_x, c_vi1), c_z), c_y)) == true
involves proving Lemma lemma1.13.2.2.2
    prefix(cons(cons(c_x, c_vi1), c_z), cons(c_y, vi2))
     => prefix(cons(c_x, c_vi1), cons(c_y, vi2))
    -> true
```

```
The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.13.2:
  false <=> prefix(cons(cons(c_x, c_vil), c_z), c_y) == true
to yield the following equations:
  Case.13.2.1: false == prefix(cons(cons(c_x, c_vil), c_z), c_y)

Ordered equation Case.13.2.1 into the rewrite rule:
  prefix(cons(cons(c_x, c_vil), c_z), c_y) -> false

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 153 rewrite rules, and 12 deduction rules.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.13.2:
  false <=> prefix(cons(cons(c_x, c_vil), c_z), c_y) == true
to yield the following equations:
  Case.13.2.2: false == prefix(cons(cons(c_x, c_vil), c_z), c_y)

Ordered equation Case.13.2.2 into the rewrite rule:
  prefix(cons(cons(c_x, c_vil), c_z), c_y) -> false

The system now contains 154 rewrite rules and 12 deduction rules.

Lemma lemma1.13.2.2.2 in the proof by cases of Lemma lemma1.13.2.2
    prefix(cons(cons(c_x, c_vil), c_z), cons(c_y, vi2))
     => prefix(cons(c_x, c_vil), cons(c_y, vi2))
    -> true
    Case.13.2: not(prefix(cons(cons(c_x, c_vil), c_z), c_y))
is NOT provable using the current partially completed system.  It reduces to
the equation
     ((c_vil = vi2) & (c_x = c_y))
      | ((c_y = cons(c_x, c_vil)) <=> false)
      | ((c_z = vi2) <=> false)
      | prefix(cons(c_x, c_vil), c_y)
     -> true

Proof of Lemma lemma1.13.2.2.2 suspended.

Critical-pair computation abandoned because a theorem has been proved.

Computed 1 new critical pair.  Added 1 of them to the system.

-> resume by case c_z=vi2::EL

Case.14.1
    c_vi2 = c_z == true
involves proving Lemma lemma1.13.2.2.2.1
    prefix(cons(cons(c_x, c_vil), c_z), cons(c_y, c_vi2))
     => prefix(cons(c_x, c_vil), cons(c_y, c_vi2))
    -> true

The case system now contains 1 equation.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.14.1:
  c_vi2 = c_z == true
to yield the following equations:
  Case.14.1.1: c_vi2 == c_z

Ordered equation Case.14.1.1 into the rewrite rule:
```

```
  c_vi2 -> c_z

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 154 rewrite rules, and 12 deduction rules.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.14.1:
  c_vi2 = c_z == true
to yield the following equations:
  Case.14.1.2: c_vi2 == c_z

Ordered equation Case.14.1.2 into the rewrite rule:
  c_vi2 -> c_z

The system now contains 155 rewrite rules and 12 deduction rules.

Lemma lemma1.13.2.2.2.1 in the proof by cases of Lemma lemma1.13.2.2.2
    prefix(cons(cons(c_x, c_vi1), c_z), cons(c_y, c_vi2))
     => prefix(cons(c_x, c_vi1), cons(c_y, c_vi2))
    -> true
    Case.14.1: c_vi2 = c_z
is NOT provable using the current partially completed system.  It reduces to
the equation
    ((c_vi1 = c_z) & (c_x = c_y))
     | ((c_y = cons(c_x, c_vi1)) <=> false)
     | prefix(cons(c_x, c_vi1), c_y)
    -> true

Proof of Lemma lemma1.13.2.2.2.1 suspended.

-> resume by case c_y=cons:Seq,EL->Seq(c_x,c_vi1)

Case.15.1
    c_y = cons(c_x, c_vi1) == true
involves proving Lemma lemma1.13.2.2.2.1.1
    prefix(cons(cons(c_x, c_vi1), c_z), cons(c_y, c_vi2))
     => prefix(cons(c_x, c_vi1), cons(c_y, c_vi2))
    -> true

The case system now contains 1 equation.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.15.1:
  c_y = cons(c_x, c_vi1) == true
to yield the following equations:
  Case.15.1.1: c_y == cons(c_x, c_vi1)

Ordered equation Case.15.1.1 into the rewrite rule:
  c_y -> cons(c_x, c_vi1)

The case system now contains 1 rewrite rule.

Lemma lemma1.13.2.2.2.1.1 in the proof by cases of Lemma lemma1.13.2.2.2.1
    prefix(cons(cons(c_x, c_vi1), c_z), cons(c_y, c_vi2))
     => prefix(cons(c_x, c_vi1), cons(c_y, c_vi2))
    -> true
    Case.15.1: c_y = cons(c_x, c_vi1)
[] Proved by rewriting (with unreduced rules).

Case.15.2
    not(c_y = cons(c_x, c_vi1)) == true
involves proving Lemma lemma1.13.2.2.2.1.2
    prefix(cons(cons(c_x, c_vi1), c_z), cons(c_y, c_vi2))
     => prefix(cons(c_x, c_vi1), cons(c_y, c_vi2))
```

```
    -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.15.2:
  (c_y = cons(c_x, c_vi1)) <=> false == true
to yield the following equations:
  Case.15.2.1: c_y = cons(c_x, c_vi1) == false

Ordered equation Case.15.2.1 into the rewrite rule:
  c_y = cons(c_x, c_vi1) -> false

The case system now contains 1 rewrite rule.

Lemma lemma1.13.2.2.2.1.2 in the proof by cases of Lemma lemma1.13.2.2.2.1
    prefix(cons(cons(c_x, c_vi1), c_z), cons(c_y, c_vi2))
     => prefix(cons(c_x, c_vi1), cons(c_y, c_vi2))
    -> true
    Case.15.2: not(c_y = cons(c_x, c_vi1))
[] Proved by rewriting (with unreduced rules).

Lemma lemma1.13.2.2.2.1 in the proof by cases of Lemma lemma1.13.2.2.2
    prefix(cons(cons(c_x, c_vi1), c_z), cons(c_y, c_vi2))
     => prefix(cons(c_x, c_vi1), cons(c_y, c_vi2))
    -> true
    Case.14.1: c_vi2 = c_z
[] Proved by cases
    (c_y = cons(c_x, c_vi1)) | not(c_y = cons(c_x, c_vi1))

Case.14.2
    not(c_vi2 = c_z) == true
involves proving Lemma lemma1.13.2.2.2.2
    prefix(cons(cons(c_x, c_vi1), c_z), cons(c_y, c_vi2))
     => prefix(cons(c_x, c_vi1), cons(c_y, c_vi2))
    -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.14.2:
  (c_vi2 = c_z) <=> false == true
to yield the following equations:
  Case.14.2.1: c_vi2 = c_z == false

Ordered equation Case.14.2.1 into the rewrite rule:
  c_vi2 = c_z -> false

The case system now contains 1 rewrite rule.

Lemma lemma1.13.2.2.2.2 in the proof by cases of Lemma lemma1.13.2.2.2
    prefix(cons(cons(c_x, c_vi1), c_z), cons(c_y, c_vi2))
     => prefix(cons(c_x, c_vi1), cons(c_y, c_vi2))
    -> true
    Case.14.2: not(c_vi2 = c_z)
[] Proved by rewriting (with unreduced rules).

Lemma lemma1.13.2.2.2 in the proof by cases of Lemma lemma1.13.2.2
    prefix(cons(cons(c_x, c_vi1), c_z), cons(c_y, vi2))
     => prefix(cons(c_x, c_vi1), cons(c_y, vi2))
    -> true
    Case.13.2: not(prefix(cons(cons(c_x, c_vi1), c_z), c_y))
[] Proved by cases
    (c_z = vi2) | not(c_z = vi2)
```

```
Lemma lemma1.13.2.2 for the induction step in the proof of Lemma lemma1.13.2
    prefix(cons(cons(c_x, vi1), z), cons(c_y, vi2))
     => prefix(cons(c_x, vi1), cons(c_y, vi2))
    -> true
[] Proved by cases
    prefix(cons(cons(c_x, vi1), z), c_y)
     | not(prefix(cons(cons(c_x, vi1), z), c_y))


Lemma lemma1.13.2 for the induction step in the proof of Conjecture lemma1.13
    prefix(cons(cons(c_x, vi1), z), y) => prefix(cons(c_x, vi1), y) -> true
[] Proved by induction over 'y' of sort 'Seq'.

Conjecture lemma1.13
    prefix(cons(x, z), y) => prefix(x, y) -> true
[] Proved by induction over 'x' of sort 'Seq'.

The system now contains 1 equation, 151 rewrite rules, and 12 deduction rules.

Ordered equation lemma1.13 into the rewrite rule:
  (false <=> prefix(cons(x, z), y)) | prefix(x, y) -> true

The system now contains 152 rewrite rules and 12 deduction rules.

-> prove prefix(cons:Seq,EL->Seq(x,z),y)=>prefix(x,y) by induction x Seq

The basis step in an inductive proof of Conjecture lemma1.14
    prefix(cons(x, z), y) => prefix(x, y) -> true
involves proving the following lemma(s):

lemma1.14.1: prefix(cons(null, z), y) => prefix(null, y) -> true
            [] Proved by normalization

The induction step in an inductive proof of Conjecture lemma1.14
    prefix(cons(x, z), y) => prefix(x, y) -> true
uses the following equation(s) for the induction hypothesis:

Induct.1: prefix(cons(c_x, z), y) => prefix(c_x, y) -> true

The system now contains 1 equation, 138 rewrite rules, and 12 deduction rules.

Ordered equation Induct.1 into the rewrite rule:
  (false <=> prefix(cons(c_x, z), y)) | prefix(c_x, y) -> true

The system now contains 139 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma1.14.2: prefix(cons(cons(c_x, vi1), z), y) => prefix(cons(c_x, vi1), y)
            -> true
                which reduces to the equation
                (false <=> prefix(cons(cons(c_x, vi1), z), y))
                 | prefix(cons(c_x, vi1), y)
                -> true

Proof of Lemma lemma1.14.2 suspended.


-> resume by induction y Seq

The basis step in an inductive proof of Lemma lemma1.14.2 for the induction step
in the proof of Conjecture lemma1.14
    prefix(cons(cons(c_x, vi1), z), y) => prefix(cons(c_x, vi1), y) -> true
involves proving the following lemma(s):

lemma1.14.2.1: prefix(cons(cons(c_x, vi1), z), null)
                => prefix(cons(c_x, vi1), null)
               -> true
               [] Proved by normalization
```

```
The induction step in an inductive proof of Lemma lemma1.14.2 for the induction
step in the proof of Conjecture lemma1.14
    prefix(cons(cons(c_x, vi1), z), y) => prefix(cons(c_x, vi1), y) -> true
uses the following equation(s) for the induction hypothesis:

Induct.2: prefix(cons(cons(c_x, vi1), z), c_y) => prefix(cons(c_x, vi1), c_y)
          -> true

The system now contains 1 equation, 139 rewrite rules, and 12 deduction rules.

Ordered equation Induct.2 into the rewrite rule:
  (false <=> prefix(cons(cons(c_x, vi1), z), c_y))
   | prefix(cons(c_x, vi1), c_y)
  -> true

The system now contains 140 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma1.14.2.2: prefix(cons(cons(c_x, vi1), z), cons(c_y, vi2))
                => prefix(cons(c_x, vi1), cons(c_y, vi2))
               -> true
                    which reduces to the equation
                    ((false <=> prefix(cons(cons(c_x, vi1), z), c_y))
                      & (((c_y = cons(c_x, vi1)) <=> false)
                          | ((vi2 = z) <=> false)))

                     | ((c_x = c_y) & (vi1 = vi2))
                     | prefix(cons(c_x, vi1), c_y)
                    -> true

Proof of Lemma lemma1.14.2.2 suspended.

-> resume by case prefix(cons(cons(c_x,vi1),z),c_y)

Case.1.1
    prefix(cons(cons(c_x, c_vi1), c_z), c_y) == true
involves proving Lemma lemma1.14.2.2.1
    prefix(cons(cons(c_x, c_vi1), c_z), cons(c_y, vi2))
     => prefix(cons(c_x, c_vi1), cons(c_y, vi2))
    -> true

The case system now contains 1 equation.

Ordered equation Case.1.1 into the rewrite rule:
  prefix(cons(cons(c_x, c_vi1), c_z), c_y) -> true

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 140 rewrite rules, and 12 deduction rules.

Ordered equation Case.1.1 into the rewrite rule:
  prefix(cons(cons(c_x, c_vi1), c_z), c_y) -> true

The system now contains 141 rewrite rules and 12 deduction rules.

Lemma lemma1.14.2.2.1 in the proof by cases of Lemma lemma1.14.2.2
    prefix(cons(cons(c_x, c_vi1), c_z), cons(c_y, vi2))
     => prefix(cons(c_x, c_vi1), cons(c_y, vi2))
    -> true
    Case.1.1: prefix(cons(cons(c_x, c_vi1), c_z), c_y)
is NOT provable using the current partially completed system.  It reduces to
the equation
     ((c_vi1 = vi2) & (c_x = c_y)) | prefix(cons(c_x, c_vi1), c_y) -> true

Proof of Lemma lemma1.14.2.2.1 suspended.

-> crit case with induct
```

```
Critical pairs between rule Case.1.1:
  prefix(cons(cons(c_x, c_vil), c_z), c_y) -> true
and rule Induct.2:
  (false <=> prefix(cons(cons(c_x, vil), z), c_y))
   | prefix(cons(c_x, vil), c_y)
  -> true
  are as follows:
    prefix(cons(c_x, c_vil), c_y) == true

The system now contains 1 equation, 141 rewrite rules, and 12 deduction rules.

Ordered equation lemma1.2 into the rewrite rule:
  prefix(cons(c_x, c_vil), c_y) -> true

The system now contains 142 rewrite rules and 12 deduction rules.

Lemma lemma1.14.2.2.1 in the proof by cases of Lemma lemma1.14.2.2
    prefix(cons(cons(c_x, c_vil), c_z), cons(c_y, vi2))
     => prefix(cons(c_x, c_vil), cons(c_y, vi2))
    -> true
    Case.1.1: prefix(cons(cons(c_x, c_vil), c_z), c_y)
[] Proved by rewriting.

Case.1.2
    not(prefix(cons(cons(c_x, c_vil), c_z), c_y)) == true
involves proving Lemma lemma1.14.2.2.2
    prefix(cons(cons(c_x, c_vil), c_z), cons(c_y, vi2))
     => prefix(cons(c_x, c_vil), cons(c_y, vi2))
    -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.1.2:
  false <=> prefix(cons(cons(c_x, c_vil), c_z), c_y) == true
to yield the following equations:
  Case.1.2.1: false == prefix(cons(cons(c_x, c_vil), c_z), c_y)

Ordered equation Case.1.2.1 into the rewrite rule:
  prefix(cons(cons(c_x, c_vil), c_z), c_y) -> false

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 140 rewrite rules, and 12 deduction rules.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.1.2:
  false <=> prefix(cons(cons(c_x, c_vil), c_z), c_y) == true
to yield the following equations:
  Case.1.2.2: false == prefix(cons(cons(c_x, c_vil), c_z), c_y)

Ordered equation Case.1.2.2 into the rewrite rule:
  prefix(cons(cons(c_x, c_vil), c_z), c_y) -> false

The system now contains 141 rewrite rules and 12 deduction rules.

Lemma lemma1.14.2.2.2 in the proof by cases of Lemma lemma1.14.2.2
    prefix(cons(cons(c_x, c_vil), c_z), cons(c_y, vi2))
     => prefix(cons(c_x, c_vil), cons(c_y, vi2))
    -> true
    Case.1.2: not(prefix(cons(cons(c_x, c_vil), c_z), c_y))
is NOT provable using the current partially completed system.  It reduces to
the equation
     ((c_vil = vi2) & (c_x = c_y))
```

```
      | ((c_y = cons(c_x, c_vi1)) <=> false)
      | ((c_z = vi2) <=> false)
      | prefix(cons(c_x, c_vi1), c_y)
    -> true

Proof of Lemma lemma1.14.2.2.2 suspended.

Critical-pair computation abandoned because a theorem has been proved.

Computed 1 new critical pair.  Added 1 of them to the system.

-> resume by case (c_y=cons(c_x,c_vi1))&(c_z=vi2::EL)

Case.2.1
    (c_vi2 = c_z) & (c_y = cons(c_x, c_vi1)) == true
involves proving Lemma lemma1.14.2.2.2.1
    prefix(cons(cons(c_x, c_vi1), c_z), cons(c_y, c_vi2))
     => prefix(cons(c_x, c_vi1), cons(c_y, c_vi2))
    -> true

The case system now contains 1 equation.

Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation Case.2.1:
  (c_vi2 = c_z) & (c_y = cons(c_x, c_vi1)) == true
to yield the following equations:
  Case.2.1.1: c_vi2 = c_z == true
  Case.2.1.2: c_y = cons(c_x, c_vi1) == true

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.2.1.2:
  c_y = cons(c_x, c_vi1) == true
to yield the following equations:
  Case.2.1.2.1: c_y == cons(c_x, c_vi1)

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.2.1.1:
  c_vi2 = c_z == true
to yield the following equations:
  Case.2.1.1.1: c_vi2 == c_z

Ordered equation Case.2.1.1.1 into the rewrite rule:
  c_vi2 -> c_z

The case system now contains 1 equation and 1 rewrite rule.

Ordered equation Case.2.1.2.1 into the rewrite rule:
  c_y -> cons(c_x, c_vi1)

The case system now contains 2 rewrite rules.

Lemma lemma1.14.2.2.2.1 in the proof by cases of Lemma lemma1.14.2.2.2
    prefix(cons(cons(c_x, c_vi1), c_z), cons(c_y, c_vi2))
     => prefix(cons(c_x, c_vi1), cons(c_y, c_vi2))
    -> true
    Case.2.1: (c_vi2 = c_z) & (c_y = cons(c_x, c_vi1))
[] Proved by rewriting (with unreduced rules).

Case.2.2
    not((c_vi2 = c_z) & (c_y = cons(c_x, c_vi1))) == true
involves proving Lemma lemma1.14.2.2.2.2
    prefix(cons(cons(c_x, c_vi1), c_z), cons(c_y, c_vi2))
```

```
     => prefix(cons(c_x, c_vi1), cons(c_y, c_vi2))
    -> true

The case system now contains 1 equation.

Ordered equation Case.2.2 into the rewrite rule:
  ((c_vi2 = c_z) <=> false) | ((c_y = cons(c_x, c_vi1)) <=> false) -> true

The case system now contains 1 rewrite rule.

Lemma lemma1.14.2.2.2.2 in the proof by cases of Lemma lemma1.14.2.2.2
    prefix(cons(cons(c_x, c_vi1), c_z), cons(c_y, c_vi2))
     => prefix(cons(c_x, c_vi1), cons(c_y, c_vi2))
    -> true
    Case.2.2: not((c_vi2 = c_z) & (c_y = cons(c_x, c_vi1)))
[] Proved by rewriting (with unreduced rules).

Lemma lemma1.14.2.2.2 in the proof by cases of Lemma lemma1.14.2.2
    prefix(cons(cons(c_x, c_vi1), c_z), cons(c_y, vi2))
     => prefix(cons(c_x, c_vi1), cons(c_y, vi2))
    -> true
    Case.1.2: not(prefix(cons(cons(c_x, c_vi1), c_z), c_y))
[] Proved by cases
     ((c_y = cons(c_x, c_vi1)) & (c_z = vi2))
      | not((c_y = cons(c_x, c_vi1)) & (c_z = vi2))


Lemma lemma1.14.2.2 for the induction step in the proof of Lemma lemma1.14.2
    prefix(cons(cons(c_x, vi1), z), cons(c_y, vi2))
     => prefix(cons(c_x, vi1), cons(c_y, vi2))
    -> true
[] Proved by cases
    prefix(cons(cons(c_x, vi1), z), c_y)
     | not(prefix(cons(cons(c_x, vi1), z), c_y))


Lemma lemma1.14.2 for the induction step in the proof of Conjecture lemma1.14
    prefix(cons(cons(c_x, vi1), z), y) => prefix(cons(c_x, vi1), y) -> true
[] Proved by induction over 'y' of sort 'Seq'.

Conjecture lemma1.14
    prefix(cons(x, z), y) => prefix(x, y) -> true
[] Proved by induction over 'x' of sort 'Seq'.

The system now contains 1 equation, 138 rewrite rules, and 12 deduction rules.

Ordered equation lemma1.14 into the rewrite rule:
  (false <=> prefix(cons(x, z), y)) | prefix(x, y) -> true

The system now contains 139 rewrite rules and 12 deduction rules.

-> prove in_state(cons(xh,we::Ev),xst)=>in_state(xh,xst) by induction xh H

The basis step in an inductive proof of Conjecture lemma1.15
    in_state(cons(xh, we), xst) => in_state(xh, xst) -> true
involves proving the following lemma(s):

lemma1.15.1: in_state(cons(null, we), xst) => in_state(null, xst) -> true
             [] Proved by normalization

The induction step in an inductive proof of Conjecture lemma1.15
    in_state(cons(xh, we), xst) => in_state(xh, xst) -> true
uses the following equation(s) for the induction hypothesis:

Induct.26: in_state(cons(c_xh, we), xst) => in_state(c_xh, xst) -> true

The system now contains 1 equation, 152 rewrite rules, and 12 deduction rules.

Ordered equation Induct.26 into the rewrite rule:
```

```
  (false <=> in_state(cons(c_xh, we), xst)) | in_state(c_xh, xst) -> true

The system now contains 153 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma1.15.2: in_state(cons(cons(c_xh, vi1), we), xst)
              => in_state(cons(c_xh, vi1), xst)
             -> true
                which reduces to the equation
                (false <=> in_state(cons(cons(c_xh, vi1), we), xst))
                 | in_state(cons(c_xh, vi1), xst)
                -> true

Proof of Lemma lemma1.15.2 suspended.

-> resume by induction we Ev

The basis step in an inductive proof of Lemma lemma1.44.2 for the induction
step in the proof of Conjecture lemma1.44
    in_state(cons(cons(c_xh, vi1), we), xst) => in_state(cons(c_xh, vi1), xst)
    -> true
involves proving the following lemma(s):

lemma1.44.2.1: in_state(cons(cons(c_xh, vi1), E(vi2)), xst)
                => in_state(cons(c_xh, vi1), xst)
               -> true
                  which reduces to the equation
                  (false <=> in_state(cons(cons(c_xh, vi1), E(vi2)), xst))
                   | in_state(cons(c_xh, vi1), xst)
                  -> true
lemma1.44.2.2: in_state(cons(cons(c_xh, vi1), D(vi2)), xst)
                => in_state(cons(c_xh, vi1), xst)
               -> true
                  which reduces to the equation
                  (false <=> in_state(cons(cons(c_xh, vi1), D(vi2)), xst))
                   | in_state(cons(c_xh, vi1), xst)
                  -> true

Proof of Lemma lemma1.44.2.2 suspended.

-> resume by case in_state(cons(cons(c_xh, vi1), D(vi2::deq_rec)),xst)

Case.17.1
    in_state(cons(cons(c_xh, c_vi1), D(c_vi2)), c_xst) == true
involves proving Lemma lemma1.44.2.2.1
    in_state(cons(cons(c_xh, c_vi1), D(c_vi2)), c_xst)
     => in_state(cons(c_xh, c_vi1), c_xst)
    -> true

The case system now contains 1 equation.

Ordered equation Case.17.1 into the rewrite rule:
  in_state(cons(cons(c_xh, c_vi1), D(c_vi2)), c_xst) -> true

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 153 rewrite rules, and 12 deduction rules.

Ordered equation Case.17.1 into the rewrite rule:
  in_state(cons(cons(c_xh, c_vi1), D(c_vi2)), c_xst) -> true

The system now contains 154 rewrite rules and 12 deduction rules.

Lemma lemma1.44.2.2.1 in the proof by cases of Lemma lemma1.44.2.2
    in_state(cons(cons(c_xh, c_vi1), D(c_vi2)), c_xst)
     => in_state(cons(c_xh, c_vi1), c_xst)
    -> true
    Case.17.1: in_state(cons(cons(c_xh, c_vi1), D(c_vi2)), c_xst)
```

```
is NOT provable using the current partially completed system.  It reduces to
the equation
    in_state(cons(c_xh, c_vil), c_xst) -> true

Proof of Lemma lemma1.44.2.2.1 suspended.

-> crit case with Abstraction.3

Critical pairs between rule Case.17.1:
  in_state(cons(cons(c_xh, c_vil), D(c_vi2)), c_xst) -> true
and rule Abstraction.3:
  (in_stack(vd, deqd(xst)) & in_state(xh, xst))
   | (false <=> in_state(cons(xh, D(vd)), xst))
  -> true
  are as follows:
    (false <=> in_state(cons(cons(cons(c_xh, c_vil), D(c_vi2)), D(vd)), c_xst))
     | in_stack(vd, deqd(c_xst))
    == true
    in_stack(c_vi2, deqd(c_xst)) & in_state(cons(c_xh, c_vil), c_xst) == true

The system now contains 1 equation, 154 rewrite rules, and 12 deduction rules.

Ordered equation lemma1.45 into the rewrite rule:
  (false <=> in_state(cons(cons(cons(c_xh, c_vil), D(c_vi2)), D(vd)), c_xst))
   | in_stack(vd, deqd(c_xst))
  -> true

The system now contains 155 rewrite rules and 12 deduction rules.

The system now contains 1 equation, 155 rewrite rules, and 12 deduction rules.

Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation lemma1.46:
  in_stack(c_vi2, deqd(c_xst)) & in_state(cons(c_xh, c_vil), c_xst) == true
to yield the following equations:
  lemma1.46.1: in_stack(c_vi2, deqd(c_xst)) == true
  lemma1.46.2: in_state(cons(c_xh, c_vil), c_xst) == true

Ordered equation lemma1.46.2 into the rewrite rule:
  in_state(cons(c_xh, c_vil), c_xst) -> true

Ordered equation lemma1.46.1 into the rewrite rule:
  in_stack(c_vi2, deqd(c_xst)) -> true

The system now contains 157 rewrite rules and 12 deduction rules.

Lemma lemma1.15.2.2.1 in the proof by cases of Lemma lemma1.15.2.2
    in_state(cons(cons(c_xh, c_vil), D(c_vi2)), c_xst)
     => in_state(cons(c_xh, c_vil), c_xst)
    -> true
    Case.17.1: in_state(cons(cons(c_xh, c_vil), D(c_vi2)), c_xst)
[] Proved by rewriting.

Case.17.2
    not(in_state(cons(cons(c_xh, c_vil), D(c_vi2)), c_xst)) == true
involves proving Lemma lemma1.15.2.2.2
    in_state(cons(cons(c_xh, c_vil), D(c_vi2)), c_xst)
     => in_state(cons(c_xh, c_vil), c_xst)
    -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.17.2:
```

```
  false <=> in_state(cons(cons(c_xh, c_vi1), D(c_vi2)), c_xst) == true
to yield the following equations:
  Case.17.2.1: false == in_state(cons(cons(c_xh, c_vi1), D(c_vi2)), c_xst)

Ordered equation Case.17.2.1 into the rewrite rule:
  in_state(cons(cons(c_xh, c_vi1), D(c_vi2)), c_xst) -> false

The case system now contains 1 rewrite rule.

Lemma lemma1.15.2.2.2 in the proof by cases of Lemma lemma1.15.2.2
    in_state(cons(cons(c_xh, c_vi1), D(c_vi2)), c_xst)
     => in_state(cons(c_xh, c_vi1), c_xst)
    -> true
    Case.17.2: not(in_state(cons(cons(c_xh, c_vi1), D(c_vi2)), c_xst))
[] Proved by rewriting (with unreduced rules).

Lemma lemma1.15.2.2 for the basis step in the proof of Lemma lemma1.15.2
    in_state(cons(cons(c_xh, vi1), D(vi2)), xst)
     => in_state(cons(c_xh, vi1), xst)
    -> true
[] Proved by cases
    in_state(cons(cons(c_xh, vi1), D(vi2)), xst)
     | not(in_state(cons(cons(c_xh, vi1), D(vi2)), xst))


Lemma lemma1.15.2.1 for the basis step in the proof of Lemma lemma1.15.2
    in_state(cons(cons(c_xh, vi1), E(vi2)), xst)
     => in_state(cons(c_xh, vi1), xst)
    -> true
is NOT provable using the current partially completed system.  It reduces to
the equation
    (false <=> in_state(cons(cons(c_xh, vi1), E(vi2)), xst))
     | in_state(cons(c_xh, vi1), xst)
    -> true

Proof of Lemma lemma1.15.2.1 suspended.

Critical-pair computation abandoned because a theorem has been proved.

Computed 2 new critical pairs.  Added 2 of them to the system.

-> resume by case in_state(cons(cons(c_xh, vi1), E(vi2::enq_rec)),xst)

Case.18.1
    in_state(cons(cons(c_xh, c_vi1), E(c_vi2)), c_xst) == true
involves proving Lemma lemma1.15.2.1.1
    in_state(cons(cons(c_xh, c_vi1), E(c_vi2)), c_xst)
     => in_state(cons(c_xh, c_vi1), c_xst)
    -> true

The case system now contains 1 equation.

Ordered equation Case.18.1 into the rewrite rule:
  in_state(cons(cons(c_xh, c_vi1), E(c_vi2)), c_xst) -> true

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 153 rewrite rules, and 12 deduction rules.

Ordered equation Case.18.1 into the rewrite rule:
  in_state(cons(cons(c_xh, c_vi1), E(c_vi2)), c_xst) -> true

The system now contains 154 rewrite rules and 12 deduction rules.

Lemma lemma1.15.2.1.1 in the proof by cases of Lemma lemma1.15.2.1
    in_state(cons(cons(c_xh, c_vi1), E(c_vi2)), c_xst)
     => in_state(cons(c_xh, c_vi1), c_xst)
    -> true
    Case.18.1: in_state(cons(cons(c_xh, c_vi1), E(c_vi2)), c_xst)
```

```
is NOT provable using the current partially completed system.  It reduces to
the equation
    in_state(cons(c_xh, c_vil), c_xst) -> true

Proof of Lemma lemma1.15.2.1.1 suspended.

-> crit case with Abstraction.2

Critical pairs between rule Case.18.1:
  in_state(cons(cons(c_xh, c_vil), E(c_vi2)), c_xst) -> true
and rule Abstraction.2:
  (in(ue, enqd(xst)) & in_state(xh, xst))
   | (false <=> in_state(cons(xh, E(ue)), xst))
  -> true
  are as follows:
    (false <=> in_state(cons(cons(cons(c_xh, c_vil), E(c_vi2)), E(ue)), c_xst))
     | in(ue, enqd(c_xst))
    == true
    in(c_vi2, enqd(c_xst)) & in_state(cons(c_xh, c_vil), c_xst) == true

The system now contains 1 equation, 154 rewrite rules, and 12 deduction rules.

Ordered equation lemma1.47 into the rewrite rule:
  (false <=> in_state(cons(cons(cons(c_xh, c_vil), E(c_vi2)), E(ue)), c_xst))
   | in(ue, enqd(c_xst))
  -> true

The system now contains 155 rewrite rules and 12 deduction rules.

The system now contains 1 equation, 155 rewrite rules, and 12 deduction rules.

Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation lemma1.48:
  in(c_vi2, enqd(c_xst)) & in_state(cons(c_xh, c_vil), c_xst) == true
to yield the following equations:
  lemma1.48.1: in(c_vi2, enqd(c_xst)) == true
  lemma1.48.2: in_state(cons(c_xh, c_vil), c_xst) == true

Ordered equation lemma1.48.2 into the rewrite rule:
  in_state(cons(c_xh, c_vil), c_xst) -> true

Ordered equation lemma1.48.1 into the rewrite rule:
  in(c_vi2, enqd(c_xst)) -> true

The system now contains 157 rewrite rules and 12 deduction rules.

Lemma lemma1.15.2.1.1 in the proof by cases of Lemma lemma1.15.2.1
    in_state(cons(cons(c_xh, c_vil), E(c_vi2)), c_xst)
     => in_state(cons(c_xh, c_vil), c_xst)
    -> true
    Case.18.1: in_state(cons(cons(c_xh, c_vil), E(c_vi2)), c_xst)
[] Proved by rewriting.

Case.18.2
    not(in_state(cons(cons(c_xh, c_vil), E(c_vi2)), c_xst)) == true
involves proving Lemma lemma1.15.2.1.2
    in_state(cons(cons(c_xh, c_vil), E(c_vi2)), c_xst)
     => in_state(cons(c_xh, c_vil), c_xst)
    -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.18.2:
```

```
  false <=> in_state(cons(cons(c_xh, c_vi1), E(c_vi2)), c_xst) == true
to yield the following equations:
  Case.18.2.1: false == in_state(cons(cons(c_xh, c_vi1), E(c_vi2)), c_xst)

Ordered equation Case.18.2.1 into the rewrite rule:
  in_state(cons(cons(c_xh, c_vi1), E(c_vi2)), c_xst) -> false

The case system now contains 1 rewrite rule.

Lemma lemma1.15.2.1.2 in the proof by cases of Lemma lemma1.15.2.1
    in_state(cons(cons(c_xh, c_vi1), E(c_vi2)), c_xst)
     => in_state(cons(c_xh, c_vi1), c_xst)
    -> true
    Case.18.2: not(in_state(cons(cons(c_xh, c_vi1), E(c_vi2)), c_xst))
[] Proved by rewriting (with unreduced rules).

Lemma lemma1.15.2.1 for the basis step in the proof of Lemma lemma1.15.2
    in_state(cons(cons(c_xh, vi1), E(vi2)), xst)
     => in_state(cons(c_xh, vi1), xst)
    -> true
[] Proved by cases
    in_state(cons(cons(c_xh, vi1), E(vi2)), xst)
     | not(in_state(cons(cons(c_xh, vi1), E(vi2)), xst))


The induction step in an inductive proof of Lemma lemma1.15.2 for the induction
step in the proof of Conjecture lemma1.15
    in_state(cons(cons(c_xh, vi1), we), xst) => in_state(cons(c_xh, vi1), xst)
    -> true
is vacuous.

Lemma lemma1.15.2 for the induction step in the proof of Conjecture lemma1.15
    in_state(cons(cons(c_xh, vi1), we), xst) => in_state(cons(c_xh, vi1), xst)
    -> true
[] Proved by induction over 'we::Ev' of sort 'Ev'.

Conjecture lemma1.15
    in_state(cons(xh, we), xst) => in_state(xh, xst) -> true
[] Proved by induction over 'xh::H' of sort 'H'.

The system now contains 1 equation, 152 rewrite rules, and 12 deduction rules.

Ordered equation lemma1.15 into the rewrite rule:
  (false <=> in_state(cons(xh, we), xst)) | in_state(xh, xst) -> true

The system now contains 153 rewrite rules and 12 deduction rules.

Critical-pair computation abandoned because a theorem has been proved.

Computed 2 new critical pairs.  Added 2 of them to the system.

-> prove prefix(x,append(x,y)) by induction x Seq

The basis step in an inductive proof of Conjecture lemma1.16
    prefix(x, append(x, y)) -> true
involves proving the following lemma(s):

lemma1.16.1: prefix(null, append(null, y)) -> true
             [] Proved by normalization

The induction step in an inductive proof of Conjecture lemma1.16
    prefix(x, append(x, y)) -> true
uses the following equation(s) for the induction hypothesis:

Induct.29: prefix(c_x, append(c_x, y)) -> true

The system now contains 1 equation, 153 rewrite rules, and 12 deduction rules.

Ordered equation Induct.29 into the rewrite rule:
```

```
  prefix(c_x, append(c_x, y)) -> true

The system now contains 154 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma1.16.2: prefix(cons(c_x, vi1), append(cons(c_x, vi1), y)) -> true

Proof of Lemma lemma1.16.2 suspended.

-> resume by induction y Seq

The basis step in an inductive proof of Lemma lemma1.16.2 for the induction
step in the proof of Conjecture lemma1.16
    prefix(cons(c_x, vi1), append(cons(c_x, vi1), y)) -> true
involves proving the following lemma(s):

lemma1.16.2.1: prefix(cons(c_x, vi1), append(cons(c_x, vi1), null)) -> true
               [] Proved by normalization

The induction step in an inductive proof of Lemma lemma1.16.2 for the induction
step in the proof of Conjecture lemma1.16
    prefix(cons(c_x, vi1), append(cons(c_x, vi1), y)) -> true
uses the following equation(s) for the induction hypothesis:

Induct.30: prefix(cons(c_x, vi1), append(cons(c_x, vi1), c_y)) -> true

The system now contains 1 equation, 154 rewrite rules, and 12 deduction rules.

Ordered equation Induct.30 into the rewrite rule:
  prefix(cons(c_x, vi1), append(cons(c_x, vi1), c_y)) -> true

The system now contains 155 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma1.16.2.2: prefix(cons(c_x, vi1), append(cons(c_x, vi1), cons(c_y, vi2)))
               -> true
               [] Proved by normalization

Lemma lemma1.16.2 for the induction step in the proof of Conjecture lemma1.16
    prefix(cons(c_x, vi1), append(cons(c_x, vi1), y)) -> true
[] Proved by induction over 'y' of sort 'Seq'.

Conjecture lemma1.16
    prefix(x, append(x, y)) -> true
[] Proved by induction over 'x' of sort 'Seq'.

The system now contains 1 equation, 153 rewrite rules, and 12 deduction rules.

Ordered equation lemma1.16 into the rewrite rule:
  prefix(x, append(x, y)) -> true

The system now contains 154 rewrite rules and 12 deduction rules.

-> prove (in_state(xh,xst)&prefix(DEQ(xh),ENQ(xh)))=>prefix(DEQ(discard(xt,xh)),ENQ(discard(xt,xh)))
by induction xh H

The basis step in an inductive proof of Conjecture lemma1.17
    (in_state(xh, xst) & prefix(DEQ(xh), ENQ(xh)))
     => prefix(DEQ(discard(xt, xh)), ENQ(discard(xt, xh)))
    -> true
involves proving the following lemma(s):

lemma1.17.1: (in_state(null, xst) & prefix(DEQ(null), ENQ(null)))
              => prefix(DEQ(discard(xt, null)), ENQ(discard(xt, null)))
             -> true
             [] Proved by normalization
```

```
The induction step in an inductive proof of Conjecture lemma1.17
     (in_state(xh, xst) & prefix(DEQ(xh), ENQ(xh)))
      => prefix(DEQ(discard(xt, xh)), ENQ(discard(xt, xh)))
     -> true
uses the following equation(s) for the induction hypothesis:

Induct.2: (in_state(c_xh, xst) & prefix(DEQ(c_xh), ENQ(c_xh)))
           => prefix(DEQ(discard(xt, c_xh)), ENQ(discard(xt, c_xh)))
          -> true

The system now contains 1 equation, 154 rewrite rules, and 12 deduction rules.

Ordered equation Induct.2 into the rewrite rule:
   (false <=> in_state(c_xh, xst))
    | (false <=> prefix(DEQ(c_xh), ENQ(c_xh)))
    | prefix(DEQ(discard(xt, c_xh)), ENQ(discard(xt, c_xh)))
   -> true

The system now contains 155 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma1.17.2: (in_state(cons(c_xh, vi1), xst)
               & prefix(DEQ(cons(c_xh, vi1)), ENQ(cons(c_xh, vi1))))
              => prefix(DEQ(discard(xt, cons(c_xh, vi1))),
                        ENQ(discard(xt, cons(c_xh, vi1))))

             -> true
                 which reduces to the equation
                 (false <=> in_state(cons(c_xh, vi1), xst))
                  | (false
                      <=> prefix(DEQ(cons(c_xh, vi1)), ENQ(cons(c_xh, vi1))))

                  | prefix(DEQ(discard(xt, cons(c_xh, vi1))),
                           ENQ(discard(xt, cons(c_xh, vi1))))

                 -> true

Proof of Lemma lemma1.17.2 suspended.

-> resume by induction vi1 Ev

The basis step in an inductive proof of Lemma lemma1.17.2 for the induction
step in the proof of Conjecture lemma1.17
     (in_state(cons(c_xh, vi1), xst)
       & prefix(DEQ(cons(c_xh, vi1)), ENQ(cons(c_xh, vi1))))
      => prefix(DEQ(discard(xt, cons(c_xh, vi1))),
                ENQ(discard(xt, cons(c_xh, vi1))))

     -> true
involves proving the following lemma(s):

lemma1.17.2.1: (in_state(cons(c_xh, E(vi2)), xst)
                 & prefix(DEQ(cons(c_xh, E(vi2))), ENQ(cons(c_xh, E(vi2))))
                => prefix(DEQ(discard(xt, cons(c_xh, E(vi2)))),
                          ENQ(discard(xt, cons(c_xh, E(vi2)))))

               -> true
                   which reduces to the equation
                   ((enqt(vi2) = xt) <=> false)
                    | (false <=> in_state(cons(c_xh, E(vi2)), xst))
                    | (false
                        <=> prefix(DEQ(c_xh), cons(ENQ(c_xh), element(vi2))))

                    | prefix(DEQ(c_xh), ENQ(c_xh))
                   -> true
lemma1.17.2.2: (in_state(cons(c_xh, D(vi2)), xst)
                 & prefix(DEQ(cons(c_xh, D(vi2))), ENQ(cons(c_xh, D(vi2))))
                => prefix(DEQ(discard(xt, cons(c_xh, D(vi2)))),
```

```
                          ENQ(discard(xt, cons(c_xh, D(vi2)))))

               -> true
               [] Proved by normalization

Proof of Lemma lemma1.17.2.1 suspended.

->  resume by case in_state(cons(c_xh,E(vi2::enq_rec)),xst)

Case.1.1
    in_state(cons(c_xh, E(c_vi2)), c_xst) == true
involves proving Lemma lemma1.17.2.1.1
    (in_state(cons(c_xh, E(c_vi2)), c_xst)
      & prefix(DEQ(cons(c_xh, E(c_vi2))), ENQ(cons(c_xh, E(c_vi2)))))
     => prefix(DEQ(discard(xt, cons(c_xh, E(c_vi2)))),
               ENQ(discard(xt, cons(c_xh, E(c_vi2)))))

    -> true

The case system now contains 1 equation.

Ordered equation Case.1.1 into the rewrite rule:
  in_state(cons(c_xh, E(c_vi2)), c_xst) -> true

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 155 rewrite rules, and 12 deduction rules.

Ordered equation Case.1.1 into the rewrite rule:
  in_state(cons(c_xh, E(c_vi2)), c_xst) -> true

The system now contains 156 rewrite rules and 12 deduction rules.

Lemma lemma1.17.2.1.1 in the proof by cases of Lemma lemma1.17.2.1
    (in_state(cons(c_xh, E(c_vi2)), c_xst)
      & prefix(DEQ(cons(c_xh, E(c_vi2))), ENQ(cons(c_xh, E(c_vi2)))))
     => prefix(DEQ(discard(xt, cons(c_xh, E(c_vi2)))),
               ENQ(discard(xt, cons(c_xh, E(c_vi2)))))

    -> true
    Case.1.1: in_state(cons(c_xh, E(c_vi2)), c_xst)
is NOT provable using the current partially completed system.  It reduces to
the equation
    ((enqt(c_vi2) = xt) <=> false)
     | (false <=> prefix(DEQ(c_xh), cons(ENQ(c_xh), element(c_vi2))))
     | prefix(DEQ(c_xh), ENQ(c_xh))
    -> true

Proof of Lemma lemma1.17.2.1.1 suspended.

-> crit case with lemma1.15

Critical pairs between rule Case.1.1:
  in_state(cons(c_xh, E(c_vi2)), c_xst) -> true
and rule lemma1.15:
  (false <=> in_state(cons(xh, we), xst)) | in_state(xh, xst) -> true
  are as follows:
    in_state(c_xh, c_xst) == true

The system now contains 1 equation, 156 rewrite rules, and 12 deduction rules.

Ordered equation lemma1.20 into the rewrite rule:
  in_state(c_xh, c_xst) -> true

The system now contains 157 rewrite rules and 12 deduction rules.

Computed 2 new critical pairs, 1 of which reduced to an identity.  Added 1 of
them to the system.
```

```
-> crit lemma1.20 with Abstraction.4

Critical pairs between rule lemma1.20:
  in_state(c_xh, c_xst) -> true
and rule Abstraction.4:
  ((DEQ(xh) = cons(ENQ(xh), xe)) <=> false) | (false <=> in_state(xh, xst))
  -> true
  are as follows:
    (DEQ(c_xh) = cons(ENQ(c_xh), xe)) <=> false == true

The system now contains 1 equation, 157 rewrite rules, and 12 deduction rules.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation lemma1.21:
  (DEQ(c_xh) = cons(ENQ(c_xh), xe)) <=> false == true
to yield the following equations:
  lemma1.21.1: DEQ(c_xh) = cons(ENQ(c_xh), xe) == false

Ordered equation lemma1.21.1 into the rewrite rule:
  DEQ(c_xh) = cons(ENQ(c_xh), xe) -> false

The system now contains 158 rewrite rules and 12 deduction rules.

Computed 1 new critical pair.  Added 1 of them to the system.

-> resume by case enqt(c_vi2)=xt

Case.2.1
    c_xt = enqt(c_vi2) == true
involves proving Lemma lemma1.17.2.1.1.1
    (in_state(cons(c_xh, E(c_vi2)), c_xst)
      & prefix(DEQ(cons(c_xh, E(c_vi2))), ENQ(cons(c_xh, E(c_vi2)))))
     => prefix(DEQ(discard(c_xt, cons(c_xh, E(c_vi2)))),
               ENQ(discard(c_xt, cons(c_xh, E(c_vi2)))))

    -> true

The case system now contains 1 equation.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.2.1:
  c_xt = enqt(c_vi2) == true
to yield the following equations:
  Case.2.1.1: c_xt == enqt(c_vi2)

Ordered equation Case.2.1.1 into the rewrite rule:
  c_xt -> enqt(c_vi2)

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 158 rewrite rules, and 12 deduction rules.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.2.1:
  c_xt = enqt(c_vi2) == true
to yield the following equations:
  Case.2.1.2: c_xt == enqt(c_vi2)

Ordered equation Case.2.1.2 into the rewrite rule:
  c_xt -> enqt(c_vi2)

The system now contains 159 rewrite rules and 12 deduction rules.
```

```
Lemma lemma1.17.2.1.1.1 in the proof by cases of Lemma lemma1.17.2.1.1
    (in_state(cons(c_xh, E(c_vi2)), c_xst)
      & prefix(DEQ(cons(c_xh, E(c_vi2))), ENQ(cons(c_xh, E(c_vi2)))))
     => prefix(DEQ(discard(c_xt, cons(c_xh, E(c_vi2)))),
               ENQ(discard(c_xt, cons(c_xh, E(c_vi2)))))

    -> true
    Case.2.1: c_xt = enqt(c_vi2)
is NOT provable using the current partially completed system.  It reduces to
the equation
    (false <=> prefix(DEQ(c_xh), cons(ENQ(c_xh), element(c_vi2))))
     | prefix(DEQ(c_xh), ENQ(c_xh))
    -> true

Proof of Lemma lemma1.17.2.1.1.1 suspended.

-> resume by case prefix(DEQ(c_xh),cons:Seq,EL->Seq(ENQ(c_xh),element(c_vi2)))

Case.3.1
    prefix(DEQ(c_xh), cons(ENQ(c_xh), element(c_vi2))) == true
involves proving Lemma lemma1.17.2.1.1.1.1
    (in_state(cons(c_xh, E(c_vi2)), c_xst)
      & prefix(DEQ(cons(c_xh, E(c_vi2))), ENQ(cons(c_xh, E(c_vi2)))))
     => prefix(DEQ(discard(c_xt, cons(c_xh, E(c_vi2)))),
               ENQ(discard(c_xt, cons(c_xh, E(c_vi2)))))

    -> true

The case system now contains 1 equation.

Ordered equation Case.3.1 into the rewrite rule:
  prefix(DEQ(c_xh), cons(ENQ(c_xh), element(c_vi2))) -> true

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 159 rewrite rules, and 12 deduction rules.

Ordered equation Case.3.1 into the rewrite rule:
  prefix(DEQ(c_xh), cons(ENQ(c_xh), element(c_vi2))) -> true

The system now contains 160 rewrite rules and 12 deduction rules.

Lemma lemma1.17.2.1.1.1.1 in the proof by cases of Lemma lemma1.17.2.1.1.1
    (in_state(cons(c_xh, E(c_vi2)), c_xst)
      & prefix(DEQ(cons(c_xh, E(c_vi2))), ENQ(cons(c_xh, E(c_vi2)))))
     => prefix(DEQ(discard(c_xt, cons(c_xh, E(c_vi2)))),
               ENQ(discard(c_xt, cons(c_xh, E(c_vi2)))))

    -> true
    Case.3.1: prefix(DEQ(c_xh), cons(ENQ(c_xh), element(c_vi2)))
is NOT provable using the current partially completed system.  It reduces to
the equation
    prefix(DEQ(c_xh), ENQ(c_xh)) -> true

Proof of Lemma lemma1.17.2.1.1.1.1 suspended.

-> crit case with lemma1.3

Critical pairs between rule Case.3.1:
  prefix(DEQ(c_xh), cons(ENQ(c_xh), element(c_vi2))) -> true
and rule lemma1.3:
  (false <=> prefix(x, cons(y, z))) | (cons(y, z) = x) | prefix(x, y) -> true
  are as follows:
    prefix(DEQ(c_xh), ENQ(c_xh)) == true

The system now contains 1 equation, 160 rewrite rules, and 12 deduction rules.

Ordered equation lemma1.22 into the rewrite rule:
  prefix(DEQ(c_xh), ENQ(c_xh)) -> true
```

```
    Left-hand side reduced:
    (false <=> in_state(c_xh, xst))
     | (false <=> prefix(DEQ(c_xh), ENQ(c_xh)))
     | prefix(DEQ(discard(xt, c_xh)), ENQ(discard(xt, c_xh)))
    -> true
      became equation Induct.2:
      (false <=> in_state(c_xh, xst))
       | (false <=> true)
       | prefix(DEQ(discard(xt, c_xh)), ENQ(discard(xt, c_xh)))
      -> true

Ordered equation Induct.2 into the rewrite rule:
  (false <=> in_state(c_xh, xst))
   | prefix(DEQ(discard(xt, c_xh)), ENQ(discard(xt, c_xh)))
  -> true

The system now contains 161 rewrite rules and 12 deduction rules.

Lemma lemma1.17.2.1.1.1.1 in the proof by cases of Lemma lemma1.17.2.1.1.1
    (in_state(cons(c_xh, E(c_vi2)), c_xst)
      & prefix(DEQ(cons(c_xh, E(c_vi2))), ENQ(cons(c_xh, E(c_vi2)))))
     => prefix(DEQ(discard(c_xt, cons(c_xh, E(c_vi2)))),
               ENQ(discard(c_xt, cons(c_xh, E(c_vi2)))))

    -> true
    Case.3.1: prefix(DEQ(c_xh), cons(ENQ(c_xh), element(c_vi2)))
[] Proved by rewriting.

Case.3.2
    not(prefix(DEQ(c_xh), cons(ENQ(c_xh), element(c_vi2)))) == true
involves proving Lemma lemma1.17.2.1.1.1.2
    (in_state(cons(c_xh, E(c_vi2)), c_xst)
      & prefix(DEQ(cons(c_xh, E(c_vi2))), ENQ(cons(c_xh, E(c_vi2)))))
     => prefix(DEQ(discard(c_xt, cons(c_xh, E(c_vi2)))),
               ENQ(discard(c_xt, cons(c_xh, E(c_vi2)))))

    -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.3.2:
  false <=> prefix(DEQ(c_xh), cons(ENQ(c_xh), element(c_vi2))) == true
to yield the following equations:
  Case.3.2.1: false == prefix(DEQ(c_xh), cons(ENQ(c_xh), element(c_vi2)))

Ordered equation Case.3.2.1 into the rewrite rule:
  prefix(DEQ(c_xh), cons(ENQ(c_xh), element(c_vi2))) -> false

The case system now contains 1 rewrite rule.

Lemma lemma1.17.2.1.1.1.2 in the proof by cases of Lemma lemma1.17.2.1.1.1
    (in_state(cons(c_xh, E(c_vi2)), c_xst)
      & prefix(DEQ(cons(c_xh, E(c_vi2))), ENQ(cons(c_xh, E(c_vi2)))))
     => prefix(DEQ(discard(c_xt, cons(c_xh, E(c_vi2)))),
               ENQ(discard(c_xt, cons(c_xh, E(c_vi2)))))

    -> true
    Case.3.2: not(prefix(DEQ(c_xh), cons(ENQ(c_xh), element(c_vi2))))
[] Proved by rewriting (with unreduced rules).

Lemma lemma1.17.2.1.1.1 in the proof by cases of Lemma lemma1.17.2.1.1
    (in_state(cons(c_xh, E(c_vi2)), c_xst)
      & prefix(DEQ(cons(c_xh, E(c_vi2))), ENQ(cons(c_xh, E(c_vi2)))))
     => prefix(DEQ(discard(c_xt, cons(c_xh, E(c_vi2)))),
               ENQ(discard(c_xt, cons(c_xh, E(c_vi2)))))
```

```
    -> true
    Case.2.1: c_xt = enqt(c_vi2)
[] Proved by cases
    prefix(DEQ(c_xh), cons(ENQ(c_xh), element(c_vi2)))
     | not(prefix(DEQ(c_xh), cons(ENQ(c_xh), element(c_vi2))))


Case.2.2
    not(c_xt = enqt(c_vi2)) == true
involves proving Lemma lemma1.17.2.1.1.2
    (in_state(cons(c_xh, E(c_vi2)), c_xst)
      & prefix(DEQ(cons(c_xh, E(c_vi2))), ENQ(cons(c_xh, E(c_vi2)))))
     => prefix(DEQ(discard(c_xt, cons(c_xh, E(c_vi2)))),
              ENQ(discard(c_xt, cons(c_xh, E(c_vi2)))))

    -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.2.2:
  (c_xt = enqt(c_vi2)) <=> false == true
to yield the following equations:
  Case.2.2.1: c_xt = enqt(c_vi2) == false

Ordered equation Case.2.2.1 into the rewrite rule:
  c_xt = enqt(c_vi2) -> false

The case system now contains 1 rewrite rule.

Lemma lemma1.17.2.1.1.2 in the proof by cases of Lemma lemma1.17.2.1.1
    (in_state(cons(c_xh, E(c_vi2)), c_xst)
      & prefix(DEQ(cons(c_xh, E(c_vi2))), ENQ(cons(c_xh, E(c_vi2)))))
     => prefix(DEQ(discard(c_xt, cons(c_xh, E(c_vi2)))),
              ENQ(discard(c_xt, cons(c_xh, E(c_vi2)))))

    -> true
    Case.2.2: not(c_xt = enqt(c_vi2))
[] Proved by rewriting (with unreduced rules).

Lemma lemma1.17.2.1.1 in the proof by cases of Lemma lemma1.17.2.1
    (in_state(cons(c_xh, E(c_vi2)), c_xst)
      & prefix(DEQ(cons(c_xh, E(c_vi2))), ENQ(cons(c_xh, E(c_vi2)))))
     => prefix(DEQ(discard(xt, cons(c_xh, E(c_vi2)))),
              ENQ(discard(xt, cons(c_xh, E(c_vi2)))))

    -> true
    Case.1.1: in_state(cons(c_xh, E(c_vi2)), c_xst)
[] Proved by cases
    (enqt(c_vi2) = xt) | not(enqt(c_vi2) = xt)

Case.1.2
    not(in_state(cons(c_xh, E(c_vi2)), c_xst)) == true
involves proving Lemma lemma1.17.2.1.2
    (in_state(cons(c_xh, E(c_vi2)), c_xst)
      & prefix(DEQ(cons(c_xh, E(c_vi2))), ENQ(cons(c_xh, E(c_vi2)))))
     => prefix(DEQ(discard(xt, cons(c_xh, E(c_vi2)))),
              ENQ(discard(xt, cons(c_xh, E(c_vi2)))))

    -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
```

```
has been applied to equation Case.1.2:
  false <=> in_state(cons(c_xh, E(c_vi2)), c_xst) == true
to yield the following equations:
  Case.1.2.1: false == in_state(cons(c_xh, E(c_vi2)), c_xst)

Ordered equation Case.1.2.1 into the rewrite rule:
  in_state(cons(c_xh, E(c_vi2)), c_xst) -> false

The case system now contains 1 rewrite rule.

Lemma lemma1.17.2.1.2 in the proof by cases of Lemma lemma1.17.2.1
    (in_state(cons(c_xh, E(c_vi2)), c_xst)
      & prefix(DEQ(cons(c_xh, E(c_vi2))), ENQ(cons(c_xh, E(c_vi2)))))
     => prefix(DEQ(discard(xt, cons(c_xh, E(c_vi2)))),
               ENQ(discard(xt, cons(c_xh, E(c_vi2)))))

    -> true
    Case.1.2: not(in_state(cons(c_xh, E(c_vi2)), c_xst))
[] Proved by rewriting (with unreduced rules).

Lemma lemma1.17.2.1 for the basis step in the proof of Lemma lemma1.17.2
    (in_state(cons(c_xh, E(vi2)), xst)
      & prefix(DEQ(cons(c_xh, E(vi2))), ENQ(cons(c_xh, E(vi2)))))
     => prefix(DEQ(discard(xt, cons(c_xh, E(vi2)))),
               ENQ(discard(xt, cons(c_xh, E(vi2)))))

    -> true
[] Proved by cases
    in_state(cons(c_xh, E(vi2)), xst) | not(in_state(cons(c_xh, E(vi2)), xst))

The induction step in an inductive proof of Lemma lemma1.17.2 for the induction
step in the proof of Conjecture lemma1.17
    (in_state(cons(c_xh, vi1), xst)
      & prefix(DEQ(cons(c_xh, vi1)), ENQ(cons(c_xh, vi1))))
     => prefix(DEQ(discard(xt, cons(c_xh, vi1))),
               ENQ(discard(xt, cons(c_xh, vi1))))

    -> true
is vacuous.

Lemma lemma1.17.2 for the induction step in the proof of Conjecture lemma1.17
    (in_state(cons(c_xh, vi1), xst)
      & prefix(DEQ(cons(c_xh, vi1)), ENQ(cons(c_xh, vi1))))
     => prefix(DEQ(discard(xt, cons(c_xh, vi1))),
               ENQ(discard(xt, cons(c_xh, vi1))))

    -> true
[] Proved by induction over 'vi1::Ev' of sort 'Ev'.

Conjecture lemma1.17
    (in_state(xh, xst) & prefix(DEQ(xh), ENQ(xh)))
     => prefix(DEQ(discard(xt, xh)), ENQ(discard(xt, xh)))
    -> true
[] Proved by induction over 'xh::H' of sort 'H'.

The system now contains 1 equation, 154 rewrite rules, and 12 deduction rules.

Ordered equation lemma1.17 into the rewrite rule:
  (false <=> in_state(xh, xst))
   | (false <=> prefix(DEQ(xh), ENQ(xh)))
   | prefix(DEQ(discard(xt, xh)), ENQ(discard(xt, xh)))
  -> true

The system now contains 155 rewrite rules and 12 deduction rules.

Critical-pair computation abandoned because a theorem has been proved.

Computed 2 new critical pairs, 1 of which reduced to an identity.  Added 1 of
them to the system.
```

-> q

5.5. Helping Lemma Set 2

```
add
  ((deqd(xst)=new)&in_state(xh,xst))=>(DEQ(xh)=null:->Seq)
  (xh=xh1)=>(ordered(xh)<=>ordered(xh1))
  ((xh=append(cons:H,Ev->H(xh1,E(pair(xe,xt))),xh2)) & ordered(xh) &
    prefix(DEQ(append(xh1,xh2)),ENQ(append(xh1,xh2))) &
    in(append(xh1,xh2), af(xst)) & (enqr(top(deqd(xst)))<xt)) =>
  prefix(DEQ(xh),ENQ(xh))
..
```

5.6. LP Proof Session of Lemma Set 2

```
Larch Prover (28 Jun 89) scripting on 14 July 1989 13:08:17 to
'/usr0/cgong/verify1/lemma2.scr'.

-> thaw theory1

System thawed from 'theory1.frz'.

-> set name lemma2

The name prefix is now 'lemma2'.

-> prove ((deqd(xst)=new)&in_state(xh,xst))=>(DEQ(xh)=null:->Seq) by induction xh H

The basis step in an inductive proof of Conjecture lemma2.1
    ((deqd(xst) = new) & in_state(xh, xst)) => (DEQ(xh) = null) -> true
involves proving the following lemma(s):

lemma2.1.1: ((deqd(xst) = new) & in_state(null, xst)) => (DEQ(null) = null)
            -> true
            [] Proved by normalization

The induction step in an inductive proof of Conjecture lemma2.1
    ((deqd(xst) = new) & in_state(xh, xst)) => (DEQ(xh) = null) -> true
uses the following equation(s) for the induction hypothesis:

Induct.1: ((deqd(xst) = new) & in_state(c_xh, xst)) => (DEQ(c_xh) = null)
          -> true

The system now contains 1 equation, 155 rewrite rules, and 12 deduction rules.

Ordered equation Induct.1 into the rewrite rule:
  ((deqd(xst) = new) <=> false)
   | (false <=> in_state(c_xh, xst))
   | (DEQ(c_xh) = null)
  -> true

The system now contains 156 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma2.1.2: ((deqd(xst) = new) & in_state(cons(c_xh, vi1), xst))
             => (DEQ(cons(c_xh, vi1)) = null)
            -> true
               which reduces to the equation
               ((deqd(xst) = new) <=> false)
                | (false <=> in_state(cons(c_xh, vi1), xst))
                | (DEQ(cons(c_xh, vi1)) = null)
               -> true

Proof of Lemma lemma2.1.2 suspended.

-> resume by induction vi1 Ev

The basis step in an inductive proof of Lemma lemma2.1.2 for the induction step
in the proof of Conjecture lemma2.1
    ((deqd(xst) = new) & in_state(cons(c_xh, vi1), xst))
     => (DEQ(cons(c_xh, vi1)) = null)
    -> true
involves proving the following lemma(s):

lemma2.1.2.1: ((deqd(xst) = new) & in_state(cons(c_xh, E(vi2)), xst))
               => (DEQ(cons(c_xh, E(vi2))) = null)
              -> true
                 which reduces to the equation
                 ((deqd(xst) = new) <=> false)
                  | (false <=> in_state(cons(c_xh, E(vi2)), xst))
```

```
                    | (DEQ(c_xh) = null)
                   -> true
lemma2.1.2.2: ((deqd(xst) = new) & in_state(cons(c_xh, D(vi2)), xst))
               => (DEQ(cons(c_xh, D(vi2))) = null)
              -> true
                   which reduces to the equation
                   ((deqd(xst) = new) <=> false)
                    | (false <=> in_state(cons(c_xh, D(vi2)), xst))
                   -> true

Proof of Lemma lemma2.1.2.2 suspended.

-> resume by case deqd(xst)=new

Case.1.1
    deqd(c_xst) = new == true
involves proving Lemma lemma2.1.2.2.1
    ((deqd(c_xst) = new) & in_state(cons(c_xh, D(vi2)), c_xst))
     => (DEQ(cons(c_xh, D(vi2))) = null)
    -> true

The case system now contains 1 equation.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.1.1:
  deqd(c_xst) = new == true
to yield the following equations:
  Case.1.1.1: deqd(c_xst) == new

Ordered equation Case.1.1.1 into the rewrite rule:
  deqd(c_xst) -> new

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 156 rewrite rules, and 12 deduction rules.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.1.1:
  deqd(c_xst) = new == true
to yield the following equations:
  Case.1.1.2: deqd(c_xst) == new

Ordered equation Case.1.1.2 into the rewrite rule:
  deqd(c_xst) -> new

The system now contains 157 rewrite rules and 12 deduction rules.

Lemma lemma2.1.2.2.1 in the proof by cases of Lemma lemma2.1.2.2
    ((deqd(c_xst) = new) & in_state(cons(c_xh, D(vi2)), c_xst))
     => (DEQ(cons(c_xh, D(vi2))) = null)
    -> true
    Case.1.1: deqd(c_xst) = new
is NOT provable using the current partially completed system.  It reduces to
the equation
    false <=> in_state(cons(c_xh, D(vi2)), c_xst) -> true

Proof of Lemma lemma2.1.2.2.1 suspended.

-> resume by case in_state(cons(c_xh,D(vi2::deq_rec)),c_xst)

Case.2.1
    in_state(cons(c_xh, D(c_vi2)), c_xst) == true
involves proving Lemma lemma2.1.2.2.1.1
    ((deqd(c_xst) = new) & in_state(cons(c_xh, D(c_vi2)), c_xst))
     => (DEQ(cons(c_xh, D(c_vi2))) = null)
```

```
    -> true

The case system now contains 1 equation.

Ordered equation Case.2.1 into the rewrite rule:
  in_state(cons(c_xh, D(c_vi2)), c_xst) -> true

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 157 rewrite rules, and 12 deduction rules.

Ordered equation Case.2.1 into the rewrite rule:
  in_state(cons(c_xh, D(c_vi2)), c_xst) -> true

The system now contains 158 rewrite rules and 12 deduction rules.

Lemma lemma2.1.2.2.1.1 in the proof by cases of Lemma lemma2.1.2.2.1
    ((deqd(c_xst) = new) & in_state(cons(c_xh, D(c_vi2)), c_xst))
     => (DEQ(cons(c_xh, D(c_vi2))) = null)
    -> true
    Case.2.1: in_state(cons(c_xh, D(c_vi2)), c_xst)
is NOT provable using the current partially completed system.  It reduces to
the equation
    false -> true

Proof of Lemma lemma2.1.2.2.1.1 suspended.

-> crit case with Abstraction.3

Critical pairs between rule Case.1.1.2:
  deqd(c_xst) -> new
and rule Abstraction.3:
  (in_stack(vd, deqd(xst)) & in_state(xh, xst))
   | (false <=> in_state(cons(xh, D(vd)), xst))
  -> true
  are as follows:
    false <=> in_state(cons(xh, D(vd)), c_xst) == true

The system now contains 1 equation, 158 rewrite rules, and 12 deduction rules.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation lemma2.2:
  false <=> in_state(cons(xh, D(vd)), c_xst) == true
to yield the following equations:
  lemma2.2.1: false == in_state(cons(xh, D(vd)), c_xst)

Ordered equation lemma2.2.1 into the rewrite rule:
  in_state(cons(xh, D(vd)), c_xst) -> false

    Left-hand side reduced:
    in_state(cons(c_xh, D(c_vi2)), c_xst) -> true
      became equation Case.2.1:
      false == true

Equation Case.2.1
    false == true
is inconsistent.

Lemma lemma2.1.2.2.1.1 in the proof by cases of Lemma lemma2.1.2.2.1
    ((deqd(c_xst) = new) & in_state(cons(c_xh, D(c_vi2)), c_xst))
     => (DEQ(cons(c_xh, D(c_vi2))) = null)
    -> true
    Case.2.1: in_state(cons(c_xh, D(c_vi2)), c_xst)
[] Proved by impossible case.

Case.2.2
    not(in_state(cons(c_xh, D(c_vi2)), c_xst)) == true
```

```
involves proving Lemma lemma2.1.2.2.1.2
    ((deqd(c_xst) = new) & in_state(cons(c_xh, D(c_vi2)), c_xst))
     => (DEQ(cons(c_xh, D(c_vi2))) = null)
    -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.2.2:
  false <=> in_state(cons(c_xh, D(c_vi2)), c_xst) == true
to yield the following equations:
  Case.2.2.1: false == in_state(cons(c_xh, D(c_vi2)), c_xst)

Ordered equation Case.2.2.1 into the rewrite rule:
  in_state(cons(c_xh, D(c_vi2)), c_xst) -> false

The case system now contains 1 rewrite rule.

Lemma lemma2.1.2.2.1.2 in the proof by cases of Lemma lemma2.1.2.2.1
    ((deqd(c_xst) = new) & in_state(cons(c_xh, D(c_vi2)), c_xst))
     => (DEQ(cons(c_xh, D(c_vi2))) = null)
    -> true
    Case.2.2: not(in_state(cons(c_xh, D(c_vi2)), c_xst))
[] Proved by rewriting (with unreduced rules).

Lemma lemma2.1.2.2.1 in the proof by cases of Lemma lemma2.1.2.2
    ((deqd(c_xst) = new) & in_state(cons(c_xh, D(vi2)), c_xst))
     => (DEQ(cons(c_xh, D(vi2))) = null)
    -> true
    Case.1.1: deqd(c_xst) = new
[] Proved by cases
    in_state(cons(c_xh, D(vi2)), c_xst)
     | not(in_state(cons(c_xh, D(vi2)), c_xst))


Case.1.2
    not(deqd(c_xst) = new) == true
involves proving Lemma lemma2.1.2.2.2
    ((deqd(c_xst) = new) & in_state(cons(c_xh, D(vi2)), c_xst))
     => (DEQ(cons(c_xh, D(vi2))) = null)
    -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.1.2:
  (deqd(c_xst) = new) <=> false == true
to yield the following equations:
  Case.1.2.1: deqd(c_xst) = new == false

Ordered equation Case.1.2.1 into the rewrite rule:
  deqd(c_xst) = new -> false

The case system now contains 1 rewrite rule.

Lemma lemma2.1.2.2.2 in the proof by cases of Lemma lemma2.1.2.2
    ((deqd(c_xst) = new) & in_state(cons(c_xh, D(vi2)), c_xst))
     => (DEQ(cons(c_xh, D(vi2))) = null)
    -> true
    Case.1.2: not(deqd(c_xst) = new)
[] Proved by rewriting (with unreduced rules).

Lemma lemma2.1.2.2 for the basis step in the proof of Lemma lemma2.1.2
    ((deqd(xst) = new) & in_state(cons(c_xh, D(vi2)), xst))
     => (DEQ(cons(c_xh, D(vi2))) = null)
```

```
    -> true
[] Proved by cases
    (deqd(xst) = new) | not(deqd(xst) = new)

Lemma lemma2.1.2.1 for the basis step in the proof of Lemma lemma2.1.2
    ((deqd(xst) = new) & in_state(cons(c_xh, E(vi2)), xst))
     => (DEQ(cons(c_xh, E(vi2))) = null)
    -> true
is NOT provable using the current partially completed system.  It reduces to
the equation
    ((deqd(xst) = new) <=> false)
     | (false <=> in_state(cons(c_xh, E(vi2)), xst))
     | (DEQ(c_xh) = null)
    -> true

Proof of Lemma lemma2.1.2.1 suspended.

Critical-pair computation abandoned because a theorem has been proved.

Computed 1 new critical pair.  Added 1 of them to the system.

-> resume by case deqd(xst)=new

Case.3.1
    deqd(c_xst) = new == true
involves proving Lemma lemma2.1.2.1.1
    ((deqd(c_xst) = new) & in_state(cons(c_xh, E(vi2)), c_xst))
     => (DEQ(cons(c_xh, E(vi2))) = null)
    -> true

The case system now contains 1 equation.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.3.1:
  deqd(c_xst) = new == true
to yield the following equations:
  Case.3.1.1: deqd(c_xst) == new

Ordered equation Case.3.1.1 into the rewrite rule:
  deqd(c_xst) -> new

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 156 rewrite rules, and 12 deduction rules.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.3.1:
  deqd(c_xst) = new == true
to yield the following equations:
  Case.3.1.2: deqd(c_xst) == new

Ordered equation Case.3.1.2 into the rewrite rule:
  deqd(c_xst) -> new

The system now contains 157 rewrite rules and 12 deduction rules.

Lemma lemma2.1.2.1.1 in the proof by cases of Lemma lemma2.1.2.1
    ((deqd(c_xst) = new) & in_state(cons(c_xh, E(vi2)), c_xst))
     => (DEQ(cons(c_xh, E(vi2))) = null)
    -> true
    Case.3.1: deqd(c_xst) = new
is NOT provable using the current partially completed system.  It reduces to
the equation
    (false <=> in_state(cons(c_xh, E(vi2)), c_xst)) | (DEQ(c_xh) = null)
    -> true
```

```
Proof of Lemma lemma2.1.2.1.1 suspended.

-> resume by case in_state(cons(c_xh,E(vi2::enq_rec)),c_xst)

Case.4.1
    in_state(cons(c_xh, E(c_vi2)), c_xst) == true
involves proving Lemma lemma2.1.2.1.1.1
    ((deqd(c_xst) = new) & in_state(cons(c_xh, E(c_vi2)), c_xst))
     => (DEQ(cons(c_xh, E(c_vi2))) = null)
    -> true

The case system now contains 1 equation.

Ordered equation Case.4.1 into the rewrite rule:
  in_state(cons(c_xh, E(c_vi2)), c_xst) -> true

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 157 rewrite rules, and 12 deduction rules.

Ordered equation Case.4.1 into the rewrite rule:
  in_state(cons(c_xh, E(c_vi2)), c_xst) -> true

The system now contains 158 rewrite rules and 12 deduction rules.

Lemma lemma2.1.2.1.1.1 in the proof by cases of Lemma lemma2.1.2.1.1
    ((deqd(c_xst) = new) & in_state(cons(c_xh, E(c_vi2)), c_xst))
     => (DEQ(cons(c_xh, E(c_vi2))) = null)
    -> true
    Case.4.1: in_state(cons(c_xh, E(c_vi2)), c_xst)
is NOT provable using the current partially completed system.  It reduces to
the equation
    DEQ(c_xh) = null -> true

Proof of Lemma lemma2.1.2.1.1.1 suspended.

-> crit case with lemma1.15

Critical pairs between rule Case.4.1:
  in_state(cons(c_xh, E(c_vi2)), c_xst) -> true
and rule lemma1.15:
  (false <=> in_state(cons(xh, we), xst)) | in_state(xh, xst) -> true
  are as follows:
    in_state(c_xh, c_xst) == true

The system now contains 1 equation, 158 rewrite rules, and 12 deduction rules.

Ordered equation lemma2.3 into the rewrite rule:
  in_state(c_xh, c_xst) -> true

The system now contains 159 rewrite rules and 12 deduction rules.

Computed 2 new critical pairs, 1 of which reduced to an identity.  Added 1 of
them to the system.

-> crit lemma2 with induct

Critical pairs between rule lemma2.3:
  in_state(c_xh, c_xst) -> true
and rule Induct.1:
  ((deqd(xst) = new) <=> false)
   | (false <=> in_state(c_xh, xst))
   | (DEQ(c_xh) = null)
  -> true
  are as follows:
    DEQ(c_xh) = null == true

The system now contains 1 equation, 159 rewrite rules, and 12 deduction rules.
```

```
Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation lemma2.4:
  DEQ(c_xh) = null == true
to yield the following equations:
  lemma2.4.1: DEQ(c_xh) == null

Ordered equation lemma2.4.1 into the rewrite rule:
  DEQ(c_xh) -> null

    Left-hand side reduced:
    ((deqd(xst) = new) <=> false)
     | (false <=> in_state(c_xh, xst))
     | (DEQ(c_xh) = null)
    -> true
      became equation Induct.1:
      ((deqd(xst) = new) <=> false)
       | (false <=> in_state(c_xh, xst))
       | (null = null)
      -> true

Lemma lemma2.1.2.1.1.1 in the proof by cases of Lemma lemma2.1.2.1.1
    ((deqd(c_xst) = new) & in_state(cons(c_xh, E(c_vi2)), c_xst))
     => (DEQ(cons(c_xh, E(c_vi2))) = null)
    -> true
    Case.4.1: in_state(cons(c_xh, E(c_vi2)), c_xst)
[] Proved by rewriting.

Case.4.2
    not(in_state(cons(c_xh, E(c_vi2)), c_xst)) == true
involves proving Lemma lemma2.1.2.1.1.2
    ((deqd(c_xst) = new) & in_state(cons(c_xh, E(c_vi2)), c_xst))
     => (DEQ(cons(c_xh, E(c_vi2))) = null)
    -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.4.2:
  false <=> in_state(cons(c_xh, E(c_vi2)), c_xst) == true
to yield the following equations:
  Case.4.2.1: false == in_state(cons(c_xh, E(c_vi2)), c_xst)

Ordered equation Case.4.2.1 into the rewrite rule:
  in_state(cons(c_xh, E(c_vi2)), c_xst) -> false

The case system now contains 1 rewrite rule.

Lemma lemma2.1.2.1.1.2 in the proof by cases of Lemma lemma2.1.2.1.1
    ((deqd(c_xst) = new) & in_state(cons(c_xh, E(c_vi2)), c_xst))
     => (DEQ(cons(c_xh, E(c_vi2))) = null)
    -> true
    Case.4.2: not(in_state(cons(c_xh, E(c_vi2)), c_xst))
[] Proved by rewriting (with unreduced rules).

Lemma lemma2.1.2.1.1 in the proof by cases of Lemma lemma2.1.2.1
    ((deqd(c_xst) = new) & in_state(cons(c_xh, E(vi2)), c_xst))
     => (DEQ(cons(c_xh, E(vi2))) = null)
    -> true
    Case.3.1: deqd(c_xst) = new
[] Proved by cases
    in_state(cons(c_xh, E(vi2)), c_xst)
     | not(in_state(cons(c_xh, E(vi2)), c_xst))
```

```
Case.3.2
    not(deqd(c_xst) = new) == true
involves proving Lemma lemma2.1.2.1.2
    ((deqd(c_xst) = new) & in_state(cons(c_xh, E(vi2)), c_xst))
     => (DEQ(cons(c_xh, E(vi2))) = null)
    -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.3.2:
  (deqd(c_xst) = new) <=> false == true
to yield the following equations:
  Case.3.2.1: deqd(c_xst) = new == false

Ordered equation Case.3.2.1 into the rewrite rule:
  deqd(c_xst) = new -> false

The case system now contains 1 rewrite rule.

Lemma lemma2.1.2.1.2 in the proof by cases of Lemma lemma2.1.2.1
    ((deqd(c_xst) = new) & in_state(cons(c_xh, E(vi2)), c_xst))
     => (DEQ(cons(c_xh, E(vi2))) = null)
    -> true
    Case.3.2: not(deqd(c_xst) = new)
[] Proved by rewriting (with unreduced rules).

Lemma lemma2.1.2.1 for the basis step in the proof of Lemma lemma2.1.2
    ((deqd(xst) = new) & in_state(cons(c_xh, E(vi2)), xst))
     => (DEQ(cons(c_xh, E(vi2))) = null)
    -> true
[] Proved by cases
    (deqd(xst) = new) | not(deqd(xst) = new)

The induction step in an inductive proof of Lemma lemma2.1.2 for the induction
step in the proof of Conjecture lemma2.1
    ((deqd(xst) = new) & in_state(cons(c_xh, vi1), xst))
     => (DEQ(cons(c_xh, vi1)) = null)
    -> true
is vacuous.

Lemma lemma2.1.2 for the induction step in the proof of Conjecture lemma2.1
    ((deqd(xst) = new) & in_state(cons(c_xh, vi1), xst))
     => (DEQ(cons(c_xh, vi1)) = null)
    -> true
[] Proved by induction over 'vi1::Ev' of sort 'Ev'.

Conjecture lemma2.1
    ((deqd(xst) = new) & in_state(xh, xst)) => (DEQ(xh) = null) -> true
[] Proved by induction over 'xh::H' of sort 'H'.

The system now contains 1 equation, 155 rewrite rules, and 12 deduction rules.

Ordered equation lemma2.1 into the rewrite rule:
  ((deqd(xst) = new) <=> false)
   | (false <=> in_state(xh, xst))
   | (DEQ(xh) = null)
  -> true

The system now contains 156 rewrite rules and 12 deduction rules.

Critical-pair computation abandoned because a theorem has been proved.

Computed 1 new critical pair.  Added 1 of them to the system.

-> prove   (xh=xh1)=>(ordered(xh)<=>ordered(xh1)) by induction xh H
```

```
The basis step in an inductive proof of Conjecture lemma2.5
    (xh = xhl) => (ordered(xh) <=> ordered(xhl)) -> true
involves proving the following lemma(s):

lemma2.5.1: (null = xhl) => (ordered(null) <=> ordered(xhl)) -> true
                which reduces to the equation
                ((null = xhl) <=> false) | ordered(xhl) -> true

Proof of Lemma lemma2.5.1 suspended.

-> resume by induction xhl H

The basis step in an inductive proof of Lemma lemma2.5.1 for the basis step in
the proof of Conjecture lemma2.5
    (null = xhl) => (ordered(null) <=> ordered(xhl)) -> true
involves proving the following lemma(s):

lemma2.5.1.1: (null = null) => (ordered(null) <=> ordered(null)) -> true
              [] Proved by normalization

The induction step in an inductive proof of Lemma lemma2.5.1 for the basis step
in the proof of Conjecture lemma2.5
    (null = xhl) => (ordered(null) <=> ordered(xhl)) -> true
uses the following equation(s) for the induction hypothesis:

Induct.2: (c_xhl = null) => (ordered(c_xhl) <=> ordered(null)) -> true

The system now contains 1 equation, 156 rewrite rules, and 12 deduction rules.

Ordered equation Induct.2 into the rewrite rule:
  ((c_xhl = null) <=> false) | ordered(c_xhl) -> true

The system now contains 157 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma2.5.1.2: (cons(c_xhl, vil) = null)
               => (ordered(cons(c_xhl, vil)) <=> ordered(null))
              -> true
              [] Proved by normalization

Lemma lemma2.5.1 for the basis step in the proof of Conjecture lemma2.5
    (null = xhl) => (ordered(null) <=> ordered(xhl)) -> true
[] Proved by induction over 'xhl' of sort 'H'.

The induction step in an inductive proof of Conjecture lemma2.5
    (xh = xhl) => (ordered(xh) <=> ordered(xhl)) -> true
uses the following equation(s) for the induction hypothesis:

Induct.3: (c_xh = xhl) => (ordered(c_xh) <=> ordered(xhl)) -> true

The system now contains 1 equation, 156 rewrite rules, and 12 deduction rules.

Ordered equation Induct.3 into the rewrite rule:
  ((c_xh = xhl) <=> false) | (ordered(c_xh) <=> ordered(xhl)) -> true

The system now contains 157 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma2.5.2: (cons(c_xh, vil) = xhl)
             => (ordered(cons(c_xh, vil)) <=> ordered(xhl))
            -> true
                which reduces to the equation
                ((cons(c_xh, vil) = xhl) <=> false)
                 | (ordered(cons(c_xh, vil)) <=> ordered(xhl))
                -> true

Proof of Lemma lemma2.5.2 suspended.
```

```
-> resume by induction xh1 H

The basis step in an inductive proof of Lemma lemma2.5.2 for the induction step
in the proof of Conjecture lemma2.5
    (cons(c_xh, vi1) = xh1) => (ordered(cons(c_xh, vi1)) <=> ordered(xh1))
    -> true
involves proving the following lemma(s):

lemma2.5.2.1: (cons(c_xh, vi1) = null)
               => (ordered(cons(c_xh, vi1)) <=> ordered(null))
              -> true
              [] Proved by normalization

The induction step in an inductive proof of Lemma lemma2.5.2 for the induction
step in the proof of Conjecture lemma2.5
    (cons(c_xh, vi1) = xh1) => (ordered(cons(c_xh, vi1)) <=> ordered(xh1))
    -> true
uses the following equation(s) for the induction hypothesis:

Induct.4: (c_xh1 = cons(c_xh, vi1))
           => (ordered(c_xh1) <=> ordered(cons(c_xh, vi1)))
          -> true

The system now contains 1 equation, 157 rewrite rules, and 12 deduction rules.

Ordered equation Induct.4 into the rewrite rule:
  ((c_xh1 = cons(c_xh, vi1)) <=> false)
   | (ordered(c_xh1) <=> ordered(cons(c_xh, vi1)))
  -> true

The system now contains 158 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma2.5.2.2: (cons(c_xh, vi1) = cons(c_xh1, vi2))
               => (ordered(cons(c_xh, vi1)) <=> ordered(cons(c_xh1, vi2)))
              -> true
                  which reduces to the equation
                  ((c_xh = c_xh1) <=> false)
                   | ((vi1 = vi2) <=> false)
                   | (ordered(cons(c_xh, vi1)) <=> ordered(cons(c_xh1, vi2)))
                  -> true

Proof of Lemma lemma2.5.2.2 suspended.

-> resume by case (c_xh=c_xh1)&(vi1=vi2::Ev)

Case.5.1
    (c_vi1 = c_vi2) & (c_xh = c_xh1) == true
involves proving Lemma lemma2.5.2.2.1
    (cons(c_xh, c_vi1) = cons(c_xh1, c_vi2))
     => (ordered(cons(c_xh, c_vi1)) <=> ordered(cons(c_xh1, c_vi2)))
    -> true

The case system now contains 1 equation.

Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation Case.5.1:
  (c_vi1 = c_vi2) & (c_xh = c_xh1) == true
to yield the following equations:
  Case.5.1.1: c_vi1 = c_vi2 == true
  Case.5.1.2: c_xh = c_xh1 == true

Deduction rule equality.4:
  when  x = y == true
```

```
  yield x == y
has been applied to equation Case.5.1.2:
  c_xh = c_xh1 == true
to yield the following equations:
  Case.5.1.2.1: c_xh == c_xh1

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.5.1.1:
  c_vi1 = c_vi2 == true
to yield the following equations:
  Case.5.1.1.1: c_vi1 == c_vi2

Ordered equation Case.5.1.2.1 into the rewrite rule:
  c_xh -> c_xh1

The case system now contains 1 equation and 1 rewrite rule.

Ordered equation Case.5.1.1.1 into the rewrite rule:
  c_vi1 -> c_vi2

The case system now contains 2 rewrite rules.

Lemma lemma2.5.2.2.1 in the proof by cases of Lemma lemma2.5.2.2
    (cons(c_xh, c_vi1) = cons(c_xh1, c_vi2))
     => (ordered(cons(c_xh, c_vi1)) <=> ordered(cons(c_xh1, c_vi2)))
    -> true
    Case.5.1: (c_vi1 = c_vi2) & (c_xh = c_xh1)
[] Proved by rewriting (with unreduced rules).

Case.5.2
    not((c_vi1 = c_vi2) & (c_xh = c_xh1)) == true
involves proving Lemma lemma2.5.2.2.2
    (cons(c_xh, c_vi1) = cons(c_xh1, c_vi2))
     => (ordered(cons(c_xh, c_vi1)) <=> ordered(cons(c_xh1, c_vi2)))
    -> true

The case system now contains 1 equation.

Ordered equation Case.5.2 into the rewrite rule:
  ((c_vi1 = c_vi2) <=> false) | ((c_xh = c_xh1) <=> false) -> true

The case system now contains 1 rewrite rule.

Lemma lemma2.5.2.2.2 in the proof by cases of Lemma lemma2.5.2.2
    (cons(c_xh, c_vi1) = cons(c_xh1, c_vi2))
     => (ordered(cons(c_xh, c_vi1)) <=> ordered(cons(c_xh1, c_vi2)))
    -> true
    Case.5.2: not((c_vi1 = c_vi2) & (c_xh = c_xh1))
[] Proved by rewriting (with unreduced rules).

Lemma lemma2.5.2.2 for the induction step in the proof of Lemma lemma2.5.2
    (cons(c_xh, vi1) = cons(c_xh1, vi2))
     => (ordered(cons(c_xh, vi1)) <=> ordered(cons(c_xh1, vi2)))
    -> true
[] Proved by cases
    ((c_xh = c_xh1) & (vi1 = vi2)) | not((c_xh = c_xh1) & (vi1 = vi2))

Lemma lemma2.5.2 for the induction step in the proof of Conjecture lemma2.5
    (cons(c_xh, vi1) = xh1) => (ordered(cons(c_xh, vi1)) <=> ordered(xh1))
    -> true
[] Proved by induction over 'xh1' of sort 'H'.

Conjecture lemma2.5
    (xh = xh1) => (ordered(xh) <=> ordered(xh1)) -> true
[] Proved by induction over 'xh::H' of sort 'H'.

The system now contains 1 equation, 156 rewrite rules, and 12 deduction rules.
```

```
Ordered equation lemma2.5 into the rewrite rule:
  ((xh = xhl) <=> false) | (ordered(xh) <=> ordered(xhl)) -> true

The system now contains 157 rewrite rules and 12 deduction rules.

-> prove
Please enter an equation to prove, terminated with a '..' line, or '?' for help:
  ((xh=append(cons:H,Ev->H(xhl,E(pair(xe,xt))),xh2)) & ordered(xh) &
    prefix(DEQ(append(xhl,xh2)),ENQ(append(xhl,xh2))) &
    in(append(xhl,xh2), af(xst)) & (enqr(top(deqd(xst)))<xt)) =>
  prefix(DEQ(xh),ENQ(xh))

..


Conjecture lemma2.3
    ((enqr(top(deqd(xst))) < xt)
      & (append(cons(xhl, E(pair(xe, xt))), xh2) = xh)
      & in(append(xhl, xh2), af(xst))
      & ordered(xh)
      & prefix(DEQ(append(xhl, xh2)), ENQ(append(xhl, xh2))))
     => prefix(DEQ(xh), ENQ(xh))
    -> true
is NOT provable using the current partially completed system.  It reduces to
the equation
    ((enqr(top(deqd(xst))) < xt) <=> false)
     | ((append(cons(xhl, E(pair(xe, xt))), xh2) = xh) <=> false)
     | (false <=> in(append(xhl, xh2), af(xst)))
     | (false <=> ordered(xh))
     | (false <=> prefix(DEQ(append(xhl, xh2)), ENQ(append(xhl, xh2))))
     | prefix(DEQ(xh), ENQ(xh))
    -> true

Proof of Conjecture lemma2.3 suspended.


-> resume by case (enqr(top(deqd(xst)))<xt)&(append(cons(xhl,E(pair(xe,xt))),
  xh2)=xh)&in(append(xhl,xh2),af(xst))&ordered(xh)&prefix(DEQ(append(xhl,xh2)),
  ENQ(append(xhl, xh2)))

Case.1.1
    (enqr(top(deqd(c_xst))) < c_xtl)
     & (append(cons(c_xhl, E(pair(c_xe, c_xtl))), c_xh2) = c_xh)
     & in(append(c_xhl, c_xh2), af(c_xst))
     & ordered(c_xh)
     & prefix(DEQ(append(c_xhl, c_xh2)), ENQ(append(c_xhl, c_xh2)))
    == true
involves proving Lemma lemma2.3.1
    ((enqr(top(deqd(c_xst))) < c_xtl)
      & (append(cons(c_xhl, E(pair(c_xe, c_xtl))), c_xh2) = c_xh)
      & in(append(c_xhl, c_xh2), af(c_xst))
      & ordered(c_xh)
      & prefix(DEQ(append(c_xhl, c_xh2)), ENQ(append(c_xhl, c_xh2))))
     => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true

The case system now contains 1 equation.

Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation Case.1.1:
  (enqr(top(deqd(c_xst))) < c_xtl)
   & (append(cons(c_xhl, E(pair(c_xe, c_xtl))), c_xh2) = c_xh)
   & in(append(c_xhl, c_xh2), af(c_xst))
   & ordered(c_xh)
   & prefix(DEQ(append(c_xhl, c_xh2)), ENQ(append(c_xhl, c_xh2)))
```

```
  == true
to yield the following equations:
  Case.1.1.1: enqr(top(deqd(c_xst))) < c_xt1 == true
  Case.1.1.2: append(cons(c_xh1, E(pair(c_xe, c_xt1))), c_xh2) = c_xh == true
  Case.1.1.3: in(append(c_xh1, c_xh2), af(c_xst)) == true
  Case.1.1.4: ordered(c_xh) == true
  Case.1.1.5: prefix(DEQ(append(c_xh1, c_xh2)), ENQ(append(c_xh1, c_xh2)))
              == true

Ordered equation Case.1.1.5 into the rewrite rule:
  prefix(DEQ(append(c_xh1, c_xh2)), ENQ(append(c_xh1, c_xh2))) -> true

Ordered equation Case.1.1.4 into the rewrite rule:
  ordered(c_xh) -> true

Ordered equation Case.1.1.3 into the rewrite rule:
  in(append(c_xh1, c_xh2), af(c_xst)) -> true

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.1.1.2:
  append(cons(c_xh1, E(pair(c_xe, c_xt1))), c_xh2) = c_xh == true
to yield the following equations:
  Case.1.1.2.1: append(cons(c_xh1, E(pair(c_xe, c_xt1))), c_xh2) == c_xh

Ordered equation Case.1.1.1 into the rewrite rule:
  enqr(top(deqd(c_xst))) < c_xt1 -> true

The case system now contains 1 equation and 4 rewrite rules.

Ordered equation Case.1.1.2.1 into the rewrite rule:
  append(cons(c_xh1, E(pair(c_xe, c_xt1))), c_xh2) -> c_xh

The case system now contains 5 rewrite rules.

The system now contains 1 equation, 159 rewrite rules, and 12 deduction rules.

Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation Case.1.1:
  (enqr(top(deqd(c_xst))) < c_xt1)
   & (append(cons(c_xh1, E(pair(c_xe, c_xt1))), c_xh2) = c_xh)
   & in(append(c_xh1, c_xh2), af(c_xst))
   & ordered(c_xh)
   & prefix(DEQ(append(c_xh1, c_xh2)), ENQ(append(c_xh1, c_xh2)))
  == true
to yield the following equations:
  Case.1.1.6: enqr(top(deqd(c_xst))) < c_xt1 == true
  Case.1.1.7: append(cons(c_xh1, E(pair(c_xe, c_xt1))), c_xh2) = c_xh == true
  Case.1.1.8: in(append(c_xh1, c_xh2), af(c_xst)) == true
  Case.1.1.9: ordered(c_xh) == true
  Case.1.1.10: prefix(DEQ(append(c_xh1, c_xh2)), ENQ(append(c_xh1, c_xh2)))
               == true

Ordered equation Case.1.1.10 into the rewrite rule:
  prefix(DEQ(append(c_xh1, c_xh2)), ENQ(append(c_xh1, c_xh2))) -> true

Ordered equation Case.1.1.9 into the rewrite rule:
  ordered(c_xh) -> true

Ordered equation Case.1.1.8 into the rewrite rule:
  in(append(c_xh1, c_xh2), af(c_xst)) -> true

Deduction rule equality.4:
  when  x = y == true
  yield x == y
```

```
has been applied to equation Case.1.1.7:
  append(cons(c_xh1, E(pair(c_xe, c_xt1))), c_xh2) = c_xh == true
to yield the following equations:
  Case.1.1.7.1: append(cons(c_xh1, E(pair(c_xe, c_xt1))), c_xh2) == c_xh

Ordered equation Case.1.1.7.1 into the rewrite rule:
  append(cons(c_xh1, E(pair(c_xe, c_xt1))), c_xh2) -> c_xh

Ordered equation Case.1.1.6 into the rewrite rule:
  enqr(top(deqd(c_xst))) < c_xt1 -> true

The system now contains 164 rewrite rules and 12 deduction rules.

Lemma lemma2.3.1 in the proof by cases of Conjecture lemma2.3
    ((enqr(top(deqd(c_xst))) < c_xt1)
      & (append(cons(c_xh1, E(pair(c_xe, c_xt1))), c_xh2) = c_xh)
      & in(append(c_xh1, c_xh2), af(c_xst))
      & ordered(c_xh)
      & prefix(DEQ(append(c_xh1, c_xh2)), ENQ(append(c_xh1, c_xh2))))
     => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true
    Case.1.1:  (enqr(top(deqd(c_xst))) < c_xt1)
               & (append(cons(c_xh1, E(pair(c_xe, c_xt1))), c_xh2) = c_xh)
               & in(append(c_xh1, c_xh2), af(c_xst))
               & ordered(c_xh)
               & prefix(DEQ(append(c_xh1, c_xh2)), ENQ(append(c_xh1, c_xh2)))

is NOT provable using the current partially completed system.  It reduces to
the equation
    prefix(DEQ(c_xh), ENQ(c_xh)) -> true

Proof of Lemma lemma2.3.1 suspended.
-> crit case.1.1.8 with Abstraction.10

Critical pairs between rule Case.1.1.8:
  in(append(c_xh1, c_xh2), af(c_xst)) -> true
and rule Abstraction.10:
  ((enqr(top(deqd(xst))) < xt) <=> false)
   | (false <=> in(append(xh1, xh2), af(xst)))
   | (false <=> ordered(append(cons(xh1, E(pair(xe, xt))), xh2)))
   | (false <=> prefix(DEQ(append(xh1, xh2)), ENQ(append(xh1, xh2))))
   | prefix(DEQ(append(xh1, xh2)), append(cons(ENQ(xh1), xe), ENQ(xh2)))
  -> true
  are as follows:
    ((enqr(top(deqd(c_xst))) < xt) <=> false)
     | (false <=> ordered(append(cons(c_xh1, E(pair(xe, xt))), c_xh2)))
     | prefix(DEQ(append(c_xh1, c_xh2)),
              append(cons(ENQ(c_xh1), xe), ENQ(c_xh2)))

    == true

The system now contains 1 equation, 164 rewrite rules, and 12 deduction rules.

Ordered equation lemma2.4 into the rewrite rule:
  ((enqr(top(deqd(c_xst))) < xt) <=> false)
   | (false <=> ordered(append(cons(c_xh1, E(pair(xe, xt))), c_xh2)))
   | prefix(DEQ(append(c_xh1, c_xh2)),
            append(cons(ENQ(c_xh1), xe), ENQ(c_xh2)))

  -> true

The system now contains 165 rewrite rules and 12 deduction rules.

Computed 1 new critical pair.  Added 1 of them to the system.

-> crit case with lemma2.2

Critical pairs between rule Case.1.1.9:
  ordered(c_xh) -> true
```

```
and rule lemma2.2:
  ((xh = xh1) <=> false) | (ordered(xh) <=> ordered(xh1)) -> true
  are as follows:
    ((c_xh = xh1) <=> false) | ordered(xh1) == true
    ((c_xh = xh) <=> false) | ordered(xh) == true

The system now contains 1 equation, 165 rewrite rules, and 12 deduction rules.

Ordered equation lemma2.5 into the rewrite rule:
  ((c_xh = xh1) <=> false) | ordered(xh1) -> true

The system now contains 166 rewrite rules and 12 deduction rules.

Computed 2 new critical pairs, 1 of which reduced to an identity.  Added 1 of
them to the system.

-> crit case.1.1.6 with lemma2

Critical pairs between rule Case.1.1.6:
  enqr(top(deqd(c_xst))) < c_xt1 -> true
and rule lemma2.4:
  ((enqr(top(deqd(c_xst))) < xt) <=> false)
   | (false <=> ordered(append(cons(c_xh1, E(pair(xe, xt))), c_xh2)))
   | prefix(DEQ(append(c_xh1, c_xh2)),
            append(cons(ENQ(c_xh1), xe), ENQ(c_xh2)))

  -> true
  are as follows:
    (false <=> ordered(append(cons(c_xh1, E(pair(xe, c_xt1))), c_xh2)))
     | prefix(DEQ(append(c_xh1, c_xh2)),
              append(cons(ENQ(c_xh1), xe), ENQ(c_xh2)))

    == true

The system now contains 1 equation, 166 rewrite rules, and 12 deduction rules.

Ordered equation lemma2.6 into the rewrite rule:
  (false <=> ordered(append(cons(c_xh1, E(pair(xe, c_xt1))), c_xh2)))
   | prefix(DEQ(append(c_xh1, c_xh2)),
            append(cons(ENQ(c_xh1), xe), ENQ(c_xh2)))

  -> true

The system now contains 167 rewrite rules and 12 deduction rules.

Computed 1 new critical pair.  Added 1 of them to the system.


-> crit case.1.1.7.1 with lemma2

Critical pairs between rule Case.1.1.7.1:
  append(cons(c_xh1, E(pair(c_xe, c_xt1))), c_xh2) -> c_xh
and rule lemma2.6:
  (false <=> ordered(append(cons(c_xh1, E(pair(xe, c_xt1))), c_xh2)))
   | prefix(DEQ(append(c_xh1, c_xh2)),
            append(cons(ENQ(c_xh1), xe), ENQ(c_xh2)))

  -> true
  are as follows:
    prefix(DEQ(append(c_xh1, c_xh2)),
           append(cons(ENQ(c_xh1), c_xe), ENQ(c_xh2)))
    == true

The system now contains 1 equation, 167 rewrite rules, and 12 deduction rules.

Ordered equation lemma2.7 into the rewrite rule:
  prefix(DEQ(append(c_xh1, c_xh2)), append(cons(ENQ(c_xh1), c_xe), ENQ(c_xh2)))
  -> true
```

```
The system now contains 168 rewrite rules and 12 deduction rules.

Computed 2 new critical pairs, 1 of which reduced to an identity.  Added 1 of
them to the system.

-> crit case.1.1.7.1 with lemma1.6

Critical pairs between rule Case.1.1.7.1:
  append(cons(c_xh1, E(pair(c_xe, c_xt1))), c_xh2) -> c_xh
and rule lemma1.6:
  ENQ(append(cons(x, E(y)), z)) -> append(cons(ENQ(x), element(y)), ENQ(z))
  are as follows:
    ENQ(c_xh) == append(cons(ENQ(c_xh1), c_xe), ENQ(c_xh2))

The system now contains 1 equation, 168 rewrite rules, and 12 deduction rules.

Ordered equation lemma2.8 into the rewrite rule:
  append(cons(ENQ(c_xh1), c_xe), ENQ(c_xh2)) -> ENQ(c_xh)

    Left-hand side reduced:
    prefix(DEQ(append(c_xh1, c_xh2)),
           append(cons(ENQ(c_xh1), c_xe), ENQ(c_xh2)))
    -> true
      became equation lemma2.7:
      prefix(DEQ(append(c_xh1, c_xh2)), ENQ(c_xh)) == true

Ordered equation lemma2.7 into the rewrite rule:
  prefix(DEQ(append(c_xh1, c_xh2)), ENQ(c_xh)) -> true

The system now contains 169 rewrite rules and 12 deduction rules.

Computed 1 new critical pair.  Added 1 of them to the system.

-> crit case.1.1.7.1 with lemma1.8

Critical pairs between rule Case.1.1.7.1:
  append(cons(c_xh1, E(pair(c_xe, c_xt1))), c_xh2) -> c_xh
and rule lemma1.8:
  DEQ(append(cons(x, E(y)), z)) -> DEQ(append(x, z))
  are as follows:
    DEQ(c_xh) == DEQ(append(c_xh1, c_xh2))

The system now contains 1 equation, 169 rewrite rules, and 12 deduction rules.

Ordered equation lemma2.9 into the rewrite rule:
  DEQ(append(c_xh1, c_xh2)) -> DEQ(c_xh)

    Following 4 left-hand sides reduced:
    prefix(DEQ(append(c_xh1, c_xh2)), ENQ(append(c_xh1, c_xh2))) -> true
      became equation Case.1.1.10:
      prefix(DEQ(c_xh), ENQ(append(c_xh1, c_xh2))) == true
    ((enqr(top(deqd(c_xst))) < xt) <=> false)
     | (false <=> ordered(append(cons(c_xh1, E(pair(xe, xt))), c_xh2)))
     | prefix(DEQ(append(c_xh1, c_xh2)),
              append(cons(ENQ(c_xh1), xe), ENQ(c_xh2)))

    -> true
      became equation lemma2.4:
      ((enqr(top(deqd(c_xst))) < xt) <=> false)
       | (false <=> ordered(append(cons(c_xh1, E(pair(xe, xt))), c_xh2)))
       | prefix(DEQ(c_xh), append(cons(ENQ(c_xh1), xe), ENQ(c_xh2)))
      == true
    (false <=> ordered(append(cons(c_xh1, E(pair(xe, c_xt1))), c_xh2)))
     | prefix(DEQ(append(c_xh1, c_xh2)),
              append(cons(ENQ(c_xh1), xe), ENQ(c_xh2)))

    -> true
      became equation lemma2.6:
      (false <=> ordered(append(cons(c_xh1, E(pair(xe, c_xt1))), c_xh2)))
```

```
       | prefix(DEQ(c_xh), append(cons(ENQ(c_xh1), xe), ENQ(c_xh2)))
      == true
    prefix(DEQ(append(c_xh1, c_xh2)), ENQ(c_xh)) -> true
      became equation lemma2.7:
      prefix(DEQ(c_xh), ENQ(c_xh)) == true

Ordered equation Case.1.1.10 into the rewrite rule:
  prefix(DEQ(c_xh), ENQ(append(c_xh1, c_xh2))) -> true

Ordered equation lemma2.4 into the rewrite rule:
  ((enqr(top(deqd(c_xst))) < xt) <=> false)
   | (false <=> ordered(append(cons(c_xh1, E(pair(xe, xt))), c_xh2)))
   | prefix(DEQ(c_xh), append(cons(ENQ(c_xh1), xe), ENQ(c_xh2)))
  -> true

Ordered equation lemma2.6 into the rewrite rule:
  (false <=> ordered(append(cons(c_xh1, E(pair(xe, c_xt1))), c_xh2)))
   | prefix(DEQ(c_xh), append(cons(ENQ(c_xh1), xe), ENQ(c_xh2)))
  -> true

Ordered equation lemma2.7 into the rewrite rule:
  prefix(DEQ(c_xh), ENQ(c_xh)) -> true

The system now contains 170 rewrite rules and 12 deduction rules.

Lemma lemma2.3.1 in the proof by cases of Conjecture lemma2.3
    ((enqr(top(deqd(c_xst))) < c_xt1)
      & (append(cons(c_xh1, E(pair(c_xe, c_xt1))), c_xh2) = c_xh)
      & in(append(c_xh1, c_xh2), af(c_xst))
      & ordered(c_xh)
      & prefix(DEQ(append(c_xh1, c_xh2)), ENQ(append(c_xh1, c_xh2))))
     => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true
    Case.1.1: (enqr(top(deqd(c_xst))) < c_xt1)
               & (append(cons(c_xh1, E(pair(c_xe, c_xt1))), c_xh2) = c_xh)
               & in(append(c_xh1, c_xh2), af(c_xst))
               & ordered(c_xh)
               & prefix(DEQ(append(c_xh1, c_xh2)), ENQ(append(c_xh1, c_xh2)))

[] Proved by rewriting.

Case.1.2
    not((enqr(top(deqd(c_xst))) < c_xt1)
         & (append(cons(c_xh1, E(pair(c_xe, c_xt1))), c_xh2) = c_xh)
         & in(append(c_xh1, c_xh2), af(c_xst))
         & ordered(c_xh)
         & prefix(DEQ(append(c_xh1, c_xh2)), ENQ(append(c_xh1, c_xh2))))

    == true
involves proving Lemma lemma2.3.2
    ((enqr(top(deqd(c_xst))) < c_xt1)
      & (append(cons(c_xh1, E(pair(c_xe, c_xt1))), c_xh2) = c_xh)
      & in(append(c_xh1, c_xh2), af(c_xst))
      & ordered(c_xh)
      & prefix(DEQ(append(c_xh1, c_xh2)), ENQ(append(c_xh1, c_xh2))))
     => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true

The case system now contains 1 equation.

Ordered equation Case.1.2 into the rewrite rule:
  ((enqr(top(deqd(c_xst))) < c_xt1) <=> false)
   | ((append(cons(c_xh1, E(pair(c_xe, c_xt1))), c_xh2) = c_xh) <=> false)
   | (false <=> in(append(c_xh1, c_xh2), af(c_xst)))
   | (false <=> ordered(c_xh))
   | (false <=> prefix(DEQ(append(c_xh1, c_xh2)), ENQ(append(c_xh1, c_xh2))))
  -> true

The case system now contains 1 rewrite rule.
```

```
Lemma lemma2.3.2 in the proof by cases of Conjecture lemma2.3
     ((enqr(top(deqd(c_xst))) < c_xtl)
       & (append(cons(c_xhl, E(pair(c_xe, c_xtl))), c_xh2) = c_xh)
       & in(append(c_xhl, c_xh2), af(c_xst))
       & ordered(c_xh)
       & prefix(DEQ(append(c_xhl, c_xh2)), ENQ(append(c_xhl, c_xh2))))
      => prefix(DEQ(c_xh), ENQ(c_xh))
     -> true
     Case.1.2: not((enqr(top(deqd(c_xst))) < c_xtl)
                      & (append(cons(c_xhl, E(pair(c_xe, c_xtl))), c_xh2) = c_xh)
                      & in(append(c_xhl, c_xh2), af(c_xst))
                      & ordered(c_xh)
                      & prefix(DEQ(append(c_xhl, c_xh2)),
                               ENQ(append(c_xhl, c_xh2))))


[] Proved by rewriting (with unreduced rules).

Conjecture lemma2.3
     ((enqr(top(deqd(xst))) < xt)
       & (append(cons(xhl, E(pair(xe, xt))), xh2) = xh)
       & in(append(xhl, xh2), af(xst))
       & ordered(xh)
       & prefix(DEQ(append(xhl, xh2)), ENQ(append(xhl, xh2))))
      => prefix(DEQ(xh), ENQ(xh))
     -> true
[] Proved by cases
     ((enqr(top(deqd(xst))) < xt)
       & (append(cons(xhl, E(pair(xe, xt))), xh2) = xh)
       & in(append(xhl, xh2), af(xst))
       & ordered(xh)
       & prefix(DEQ(append(xhl, xh2)), ENQ(append(xhl, xh2))))
      | not((enqr(top(deqd(xst))) < xt)
            & (append(cons(xhl, E(pair(xe, xt))), xh2) = xh)
            & in(append(xhl, xh2), af(xst))
            & ordered(xh)
            & prefix(DEQ(append(xhl, xh2)), ENQ(append(xhl, xh2))))


The system now contains 1 equation, 159 rewrite rules, and 12 deduction rules.

Ordered equation lemma2.3 into the rewrite rule:
   ((enqr(top(deqd(xst))) < xt) <=> false)
    | ((append(cons(xhl, E(pair(xe, xt))), xh2) = xh) <=> false)
    | (false <=> in(append(xhl, xh2), af(xst)))
    | (false <=> ordered(xh))
    | (false <=> prefix(DEQ(append(xhl, xh2)), ENQ(append(xhl, xh2))))
    | prefix(DEQ(xh), ENQ(xh))
   -> true

The system now contains 160 rewrite rules and 12 deduction rules.

Critical-pair computation abandoned because a theorem has been proved.

Computed 1 new critical pair.  Added 1 of them to the system.

-> forget undo

Undo stack cleared.

-> freeze theory2

System frozen in 'theory2.frz'.

-> q
```

5.7. Helping Lemma Set 3

```
add
(DEQ(append(xh ,xh1))=null:->Seq)=>((DEQ(xh)=null:->Seq)&(DEQ(xh1)=null:->Seq))
((xh=append(cons(xh1, D(trip(element(xn), enqt(xn), xt))), xh2)) &
 (DEQ(xh1)=null:->Seq) & (DEQ(xh2)=null:->Seq) & in(append(xh1,xh2),af(xst)) &
 in(xn,enqd(xst)) & least(xn,enqd(xst))) => prefix(DEQ(xh),ENQ(xh))
((xh=append(cons(xh1, D(trip(element(xn), enqt(xn), xt))), xh2)) &
 in(append(xh1,xh2),af(xst)) & in(xn,enqd(xst)) & least(xn,enqd(xst)) &
 prefix(DEQ(append(xh1, xh2)), ENQ(append(xh1, xh2)))&(DEQ(xh2)=null:->Seq)&
 (enqr(top(deqd(xst))) < enqt(xn))) => prefix(DEQ(xh),ENQ(xh))
..
```

5.8. LP Proof Session of Lemma Set 3

```
-> thaw theory2

System thawed from 'theory2.frz'.

-> set name lemma3

The name prefix is now 'lemma3'.

-> prove (DEQ(append(xh ,xh1))=null:->Seq)=>((DEQ(xh)=null:->Seq)&(DEQ(xh1)=null:->Seq)) by induction
xh H

The basis step in an inductive proof of Conjecture lemma3.1
     (DEQ(append(xh, xh1)) = null) => ((DEQ(xh) = null) & (DEQ(xh1) = null))
     -> true
involves proving the following lemma(s):

lemma3.1.1: (DEQ(append(null, xh1)) = null)
             => ((DEQ(null) = null) & (DEQ(xh1) = null))
            -> true
            [] Proved by normalization

The induction step in an inductive proof of Conjecture lemma3.1
     (DEQ(append(xh, xh1)) = null) => ((DEQ(xh) = null) & (DEQ(xh1) = null))
     -> true
uses the following equation(s) for the induction hypothesis:

Induct.1: (DEQ(append(c_xh, xh1)) = null)
           => ((DEQ(c_xh) = null) & (DEQ(xh1) = null))
          -> true

The system now contains 1 equation, 160 rewrite rules, and 12 deduction rules.

Ordered equation Induct.1 into the rewrite rule:
   ((DEQ(c_xh) = null) & (DEQ(xh1) = null))
    | ((DEQ(append(c_xh, xh1)) = null) <=> false)
   -> true

The system now contains 161 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma3.1.2: (DEQ(append(cons(c_xh, vi1), xh1)) = null)
             => ((DEQ(cons(c_xh, vi1)) = null) & (DEQ(xh1) = null))
            -> true
                which reduces to the equation
                ((DEQ(cons(c_xh, vi1)) = null) & (DEQ(xh1) = null))
                 | ((DEQ(append(cons(c_xh, vi1), xh1)) = null) <=> false)
                -> true

Proof of Lemma lemma3.1.2 suspended.

-> resume by induction xh1
Please enter a sort for the induction: H

The basis step in an inductive proof of Lemma lemma3.1.2 for the induction step
in the proof of Conjecture lemma3.1
     (DEQ(append(cons(c_xh, vi1), xh1)) = null)
      => ((DEQ(cons(c_xh, vi1)) = null) & (DEQ(xh1) = null))
     -> true
involves proving the following lemma(s):

lemma3.1.2.1: (DEQ(append(cons(c_xh, vi1), null)) = null)
               => ((DEQ(cons(c_xh, vi1)) = null) & (DEQ(null) = null))
              -> true
              [] Proved by normalization
```

```
The induction step in an inductive proof of Lemma lemma3.1.2 for the induction
step in the proof of Conjecture lemma3.1
    (DEQ(append(cons(c_xh, vi1), xh1)) = null)
     => ((DEQ(cons(c_xh, vi1)) = null) & (DEQ(xh1) = null))
    -> true
uses the following equation(s) for the induction hypothesis:

Induct.2: (DEQ(append(cons(c_xh, vi1), c_xh1)) = null)
           => ((DEQ(c_xh1) = null) & (DEQ(cons(c_xh, vi1)) = null))
          -> true

The system now contains 1 equation, 161 rewrite rules, and 12 deduction rules.

Ordered equation Induct.2 into the rewrite rule:
  ((DEQ(c_xh1) = null) & (DEQ(cons(c_xh, vi1)) = null))
   | ((DEQ(append(cons(c_xh, vi1), c_xh1)) = null) <=> false)
  -> true

The system now contains 162 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma3.1.2.2: (DEQ(append(cons(c_xh, vi1), cons(c_xh1, vi2))) = null)
               => ((DEQ(cons(c_xh, vi1)) = null)
                    & (DEQ(cons(c_xh1, vi2)) = null))

              -> true
                  which reduces to the equation
                  ((DEQ(cons(c_xh, vi1)) = null)
                    & (DEQ(cons(c_xh1, vi2)) = null))
                   | ((DEQ(cons(append(cons(c_xh, vi1), c_xh1), vi2)) = null)
                       <=> false)

                  -> true

Proof of Lemma lemma3.1.2.2 suspended.

-> resume by induction vi2 Ev

The basis step in an inductive proof of Lemma lemma3.1.2.2 for the induction
step in the proof of Lemma lemma3.1.2
    (DEQ(append(cons(c_xh, vi1), cons(c_xh1, vi2))) = null)
     => ((DEQ(cons(c_xh, vi1)) = null) & (DEQ(cons(c_xh1, vi2)) = null))
    -> true
involves proving the following lemma(s):

lemma3.1.2.2.1: (DEQ(append(cons(c_xh, vi1), cons(c_xh1, E(vi3)))) = null)
                 => ((DEQ(cons(c_xh, vi1)) = null)
                      & (DEQ(cons(c_xh1, E(vi3))) = null))

                -> true
                [] Proved by normalization
lemma3.1.2.2.2: (DEQ(append(cons(c_xh, vi1), cons(c_xh1, D(vi3)))) = null)
                 => ((DEQ(cons(c_xh, vi1)) = null)
                      & (DEQ(cons(c_xh1, D(vi3))) = null))

                -> true
                [] Proved by normalization

The induction step in an inductive proof of Lemma lemma3.1.2.2 for the
induction step in the proof of Lemma lemma3.1.2
    (DEQ(append(cons(c_xh, vi1), cons(c_xh1, vi2))) = null)
     => ((DEQ(cons(c_xh, vi1)) = null) & (DEQ(cons(c_xh1, vi2)) = null))
    -> true
is vacuous.

Lemma lemma3.1.2.2 for the induction step in the proof of Lemma lemma3.1.2
    (DEQ(append(cons(c_xh, vi1), cons(c_xh1, vi2))) = null)
     => ((DEQ(cons(c_xh, vi1)) = null) & (DEQ(cons(c_xh1, vi2)) = null))
```

```
    -> true
[] Proved by induction over 'vi2::Ev' of sort 'Ev'.

Lemma lemma3.1.2 for the induction step in the proof of Conjecture lemma3.1
    (DEQ(append(cons(c_xh, vil), xh1)) = null)
     => ((DEQ(cons(c_xh, vil)) = null) & (DEQ(xh1) = null))
    -> true
[] Proved by induction over 'xh1' of sort 'H'.

Conjecture lemma3.1
    (DEQ(append(xh, xh1)) = null) => ((DEQ(xh) = null) & (DEQ(xh1) = null))
    -> true
[] Proved by induction over 'xh::H' of sort 'H'.

The system now contains 1 equation, 160 rewrite rules, and 12 deduction rules.

Ordered equation lemma3.1 into the rewrite rule:
  ((DEQ(xh) = null) & (DEQ(xh1) = null))
   | ((DEQ(append(xh, xh1)) = null) <=> false)
  -> true

The system now contains 161 rewrite rules and 12 deduction rules.

-> prove
Please enter an equation to prove, terminated with a '..' line, or '?' for help:
((xh=append(cons(xh1, D(trip(element(xn), enqt(xn), xt))), xh2)) &
 (DEQ(xh1)=null:->Seq) & (DEQ(xh2)=null:->Seq) & in(append(xh1,xh2),af(xst)) &
 in(xn,enqd(xst)) & least(xn,enqd(xst))) => prefix(DEQ(xh),ENQ(xh))
..


Conjecture lemma3.2
    ((DEQ(xh1) = null)
      & (DEQ(xh2) = null)
      & (append(cons(xh1, D(trip(element(xn), enqt(xn), xt))), xh2) = xh)
      & in(append(xh1, xh2), af(xst))
      & in(xn, enqd(xst))
      & least(xn, enqd(xst)))
     => prefix(DEQ(xh), ENQ(xh))
    -> true
is NOT provable using the current partially completed system.  It reduces to
the equation
    ((DEQ(xh1) = null) <=> false)
     | ((DEQ(xh2) = null) <=> false)
     | ((append(cons(xh1, D(trip(element(xn), enqt(xn), xt))), xh2) = xh)
         <=> false)

     | (false <=> in(append(xh1, xh2), af(xst)))
     | (false <=> in(xn, enqd(xst)))
     | (false <=> least(xn, enqd(xst)))
     | prefix(DEQ(xh), ENQ(xh))
    -> true

Proof of Conjecture lemma3.2 suspended.

-> resume by case (append(cons(xh1,D(trip(element(xn),enqt(xn),xt))),xh2)=xh)
  &in(append(xh1,xh2),af(xst))&in(xn,enqd(xst))&least(xn,enqd(xst))&
  (DEQ(xh1)=null:->Seq)&(DEQ(xh2)=null:->Seq)

Case.1.1
    (DEQ(c_xh1) = null)
     & (DEQ(c_xh2) = null)
     & (append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))), c_xh2)
         = c_xh)

     & in(append(c_xh1, c_xh2), af(c_xst))
     & in(c_xn, enqd(c_xst))
     & least(c_xn, enqd(c_xst))
    == true
```

```
involves proving Lemma lemma3.2.1
     ((DEQ(c_xh1) = null)
       & (DEQ(c_xh2) = null)
       & (append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))), c_xh2)
           = c_xh)

       & in(append(c_xh1, c_xh2), af(c_xst))
       & in(c_xn, enqd(c_xst))
       & least(c_xn, enqd(c_xst)))
      => prefix(DEQ(c_xh), ENQ(c_xh))
     -> true

The case system now contains 1 equation.

Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation Case.1.1:
   (DEQ(c_xh1) = null)
    & (DEQ(c_xh2) = null)
    & (append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))), c_xh2)
        = c_xh)

    & in(append(c_xh1, c_xh2), af(c_xst))
    & in(c_xn, enqd(c_xst))
    & least(c_xn, enqd(c_xst))
   == true
to yield the following equations:
  Case.1.1.1: DEQ(c_xh1) = null == true
  Case.1.1.2: DEQ(c_xh2) = null == true
  Case.1.1.3: append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))),
                     c_xh2)
               = c_xh
              == true
  Case.1.1.4: in(append(c_xh1, c_xh2), af(c_xst)) == true
  Case.1.1.5: in(c_xn, enqd(c_xst)) == true
  Case.1.1.6: least(c_xn, enqd(c_xst)) == true

Ordered equation Case.1.1.6 into the rewrite rule:
  least(c_xn, enqd(c_xst)) -> true

Ordered equation Case.1.1.5 into the rewrite rule:
  in(c_xn, enqd(c_xst)) -> true

Ordered equation Case.1.1.4 into the rewrite rule:
  in(append(c_xh1, c_xh2), af(c_xst)) -> true

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.1.1.3:
  append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))), c_xh2) = c_xh
  == true
to yield the following equations:
  Case.1.1.3.1: append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))),
                       c_xh2)
                == c_xh

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.1.1.2:
  DEQ(c_xh2) = null == true
to yield the following equations:
  Case.1.1.2.1: DEQ(c_xh2) == null

Deduction rule equality.4:
  when  x = y == true
```

```
  yield x == y
has been applied to equation Case.1.1.1:
  DEQ(c_xh1) = null == true
to yield the following equations:
  Case.1.1.1.1: DEQ(c_xh1) == null

The case system now contains 3 equations and 3 rewrite rules.

Ordered equation Case.1.1.2.1 into the rewrite rule:
  DEQ(c_xh2) -> null

The case system now contains 2 equations and 4 rewrite rules.

Ordered equation Case.1.1.1.1 into the rewrite rule:
  DEQ(c_xh1) -> null

The case system now contains 1 equation and 5 rewrite rules.

Ordered equation Case.1.1.3.1 into the rewrite rule:
  append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))), c_xh2) -> c_xh

The case system now contains 6 rewrite rules.

The system now contains 1 equation, 162 rewrite rules, and 12 deduction rules.

Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation Case.1.1:
  (DEQ(c_xh1) = null)
   & (DEQ(c_xh2) = null)
   & (append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))), c_xh2)
       = c_xh)

   & in(append(c_xh1, c_xh2), af(c_xst))
   & in(c_xn, enqd(c_xst))
   & least(c_xn, enqd(c_xst))
  == true
to yield the following equations:
  Case.1.1.7: DEQ(c_xh1) = null == true
  Case.1.1.8: DEQ(c_xh2) = null == true
  Case.1.1.9: append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))),
                     c_xh2)
               = c_xh
              == true
  Case.1.1.10: in(append(c_xh1, c_xh2), af(c_xst)) == true
  Case.1.1.11: in(c_xn, enqd(c_xst)) == true
  Case.1.1.12: least(c_xn, enqd(c_xst)) == true

Ordered equation Case.1.1.12 into the rewrite rule:
  least(c_xn, enqd(c_xst)) -> true

Ordered equation Case.1.1.11 into the rewrite rule:
  in(c_xn, enqd(c_xst)) -> true

Ordered equation Case.1.1.10 into the rewrite rule:
  in(append(c_xh1, c_xh2), af(c_xst)) -> true

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.1.1.9:
  append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))), c_xh2) = c_xh
  == true
to yield the following equations:
  Case.1.1.9.1: append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))),
                       c_xh2)
                == c_xh
```

```
Ordered equation Case.1.1.9.1 into the rewrite rule:
  append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))), c_xh2) -> c_xh

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.1.1.8:
  DEQ(c_xh2) = null == true
to yield the following equations:
  Case.1.1.8.1: DEQ(c_xh2) == null

Ordered equation Case.1.1.8.1 into the rewrite rule:
  DEQ(c_xh2) -> null

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.1.1.7:
  DEQ(c_xh1) = null == true
to yield the following equations:
  Case.1.1.7.1: DEQ(c_xh1) == null

Ordered equation Case.1.1.7.1 into the rewrite rule:
  DEQ(c_xh1) -> null

The system now contains 168 rewrite rules and 12 deduction rules.

Lemma lemma3.2.1 in the proof by cases of Conjecture lemma3.2
    ((DEQ(c_xh1) = null)
      & (DEQ(c_xh2) = null)
      & (append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))), c_xh2)
          = c_xh)

      & in(append(c_xh1, c_xh2), af(c_xst))
      & in(c_xn, enqd(c_xst))
      & least(c_xn, enqd(c_xst)))
     => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true
    Case.1.1: (DEQ(c_xh1) = null)
              & (DEQ(c_xh2) = null)
              & (append(cons(c_xh1,
                             D(trip(element(c_xn), enqt(c_xn), c_xt1))),

                    c_xh2)
                  = c_xh)

              & in(append(c_xh1, c_xh2), af(c_xst))
              & in(c_xn, enqd(c_xst))
              & least(c_xn, enqd(c_xst))

is NOT provable using the current partially completed system.  It reduces to
the equation
    prefix(DEQ(c_xh), ENQ(c_xh)) -> true

Proof of Lemma lemma3.2.1 suspended.


-> crit case with lemma1.7

Critical pairs between rule Case.1.1.9.1:
  append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))), c_xh2) -> c_xh
and rule lemma1.7:
  ENQ(append(cons(x, D(y)), z)) -> ENQ(append(x, z))
  are as follows:
    ENQ(c_xh) == ENQ(append(c_xh1, c_xh2))

The system now contains 1 equation, 168 rewrite rules, and 12 deduction rules.
```

```
Ordered equation lemma3.3 into the rewrite rule:
  ENQ(append(c_xh1, c_xh2)) -> ENQ(c_xh)

The system now contains 169 rewrite rules and 12 deduction rules.

Computed 1 new critical pair.  Added 1 of them to the system.

-> crit case with lemma1.9

Critical pairs between rule Case.1.1.9.1:
  append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))), c_xh2) -> c_xh
and rule lemma1.9:
  DEQ(append(cons(x, D(y)), z)) -> append(cons(DEQ(x), what(y)), DEQ(z))
  are as follows:
    DEQ(c_xh) == cons(null, element(c_xn))

The system now contains 1 equation, 169 rewrite rules, and 12 deduction rules.

Ordered equation lemma3.4 into the rewrite rule:
  DEQ(c_xh) -> cons(null, element(c_xn))

The system now contains 170 rewrite rules and 12 deduction rules.

Lemma lemma3.2.1 in the proof by cases of Conjecture lemma3.2
    ((DEQ(c_xh1) = null)
      & (DEQ(c_xh2) = null)
      & (append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))), c_xh2)
          = c_xh)

      & in(append(c_xh1, c_xh2), af(c_xst))
      & in(c_xn, enqd(c_xst))
      & least(c_xn, enqd(c_xst)))
     => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true
    Case.1.1: (DEQ(c_xh1) = null)
               & (DEQ(c_xh2) = null)
               & (append(cons(c_xh1,
                              D(trip(element(c_xn), enqt(c_xn), c_xt1))),

                    c_xh2)
                   = c_xh)

               & in(append(c_xh1, c_xh2), af(c_xst))
               & in(c_xn, enqd(c_xst))
               & least(c_xn, enqd(c_xst))

is NOT provable using the current partially completed system.  It reduces to
the equation
    prefix(cons(null, element(c_xn)), ENQ(c_xh)) -> true

Proof of Lemma lemma3.2.1 suspended.

Computed 1 new critical pair.  Added 1 of them to the system.

-> crit case.1.1.11 with Abstraction.11

Critical pairs between rule Case.1.1.11:
  in(c_xn, enqd(c_xst)) -> true
and rule Abstraction.11:
  (false <=> in(xh, af(xst)))
   | (false <=> in(xn, enqd(xst)))
   | (false <=> least(xn, enqd(xst)))
   | (false <=> prefix(DEQ(xh), ENQ(xh)))
   | prefix(cons(DEQ(xh), element(xn)), ENQ(xh))
  -> true
  are as follows:
    (false <=> in(xh, af(c_xst)))
     | (false <=> prefix(DEQ(xh), ENQ(xh)))
     | prefix(cons(DEQ(xh), element(c_xn)), ENQ(xh))
```

```
    == true

The system now contains 1 equation, 170 rewrite rules, and 12 deduction rules.

Ordered equation lemma3.5 into the rewrite rule:
  (false <=> in(xh, af(c_xst)))
   | (false <=> prefix(DEQ(xh), ENQ(xh)))
   | prefix(cons(DEQ(xh), element(c_xn)), ENQ(xh))
  -> true

The system now contains 171 rewrite rules and 12 deduction rules.

Computed 1 new critical pair.  Added 1 of them to the system.

-> instantiate xh by append(c_xh1,c_xh2) in lemma3.5


Equation lemma3.5:
  (false <=> in(xh, af(c_xst)))
   | (false <=> prefix(DEQ(xh), ENQ(xh)))
   | prefix(cons(DEQ(xh), element(c_xn)), ENQ(xh))
  -> true
has been instantiated to equation lemma3.5.1:
  (false <=> prefix(DEQ(append(c_xh1, c_xh2)), ENQ(c_xh)))
   | prefix(cons(DEQ(append(c_xh1, c_xh2)), element(c_xn)), ENQ(c_xh))
  == true

Added 1 equation to the system.

Ordered equation lemma3.5.1 into the rewrite rule:
  (false <=> prefix(DEQ(append(c_xh1, c_xh2)), ENQ(c_xh)))
   | prefix(cons(DEQ(append(c_xh1, c_xh2)), element(c_xn)), ENQ(c_xh))
  -> true

The system now contains 172 rewrite rules and 12 deduction rules.

-> prove ((DEQ(xh1)=null:->Seq)&(DEQ(xh2)=null:->Seq))=>(DEQ(append(xh1,xh2))=null:->Seq) by induction
xh1 H

The basis step in an inductive proof of Conjecture lemma3.6
    ((DEQ(xh1) = null) & (DEQ(xh2) = null)) => (DEQ(append(xh1, xh2)) = null)
    -> true
involves proving the following lemma(s):

lemma3.6.1: ((DEQ(null) = null) & (DEQ(xh2) = null))
             => (DEQ(append(null, xh2)) = null)
            -> true
            [] Proved by normalization

The induction step in an inductive proof of Conjecture lemma3.6
    ((DEQ(xh1) = null) & (DEQ(xh2) = null)) => (DEQ(append(xh1, xh2)) = null)
    -> true
uses the following equation(s) for the induction hypothesis:

Induct.1: ((DEQ(c_xh3) = null) & (DEQ(xh2) = null))
           => (DEQ(append(c_xh3, xh2)) = null)
          -> true

The system now contains 1 equation, 172 rewrite rules, and 12 deduction rules.

Ordered equation Induct.1 into the rewrite rule:
  ((DEQ(c_xh3) = null) <=> false)
   | ((DEQ(xh2) = null) <=> false)
   | (DEQ(append(c_xh3, xh2)) = null)
  -> true

The system now contains 173 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):
```

```
lemma3.6.2: ((DEQ(cons(c_xh3, vi1)) = null) & (DEQ(xh2) = null))
             => (DEQ(append(cons(c_xh3, vi1), xh2)) = null)
            -> true
                which reduces to the equation
                ((DEQ(cons(c_xh3, vi1)) = null) <=> false)
                 | ((DEQ(xh2) = null) <=> false)
                 | (DEQ(append(cons(c_xh3, vi1), xh2)) = null)
                -> true

Proof of Lemma lemma3.6.2 suspended.

-> resume by induction xh2 H

The basis step in an inductive proof of Lemma lemma3.6.2 for the induction step
in the proof of Conjecture lemma3.6
    ((DEQ(cons(c_xh3, vi1)) = null) & (DEQ(xh2) = null))
     => (DEQ(append(cons(c_xh3, vi1), xh2)) = null)
    -> true
involves proving the following lemma(s):

lemma3.6.2.1: ((DEQ(cons(c_xh3, vi1)) = null) & (DEQ(null) = null))
               => (DEQ(append(cons(c_xh3, vi1), null)) = null)
              -> true
              [] Proved by normalization

The induction step in an inductive proof of Lemma lemma3.6.2 for the induction
step in the proof of Conjecture lemma3.6
    ((DEQ(cons(c_xh3, vi1)) = null) & (DEQ(xh2) = null))
     => (DEQ(append(cons(c_xh3, vi1), xh2)) = null)
    -> true
uses the following equation(s) for the induction hypothesis:

Induct.2: ((DEQ(c_xh4) = null) & (DEQ(cons(c_xh3, vi1)) = null))
           => (DEQ(append(cons(c_xh3, vi1), c_xh4)) = null)
          -> true

The system now contains 1 equation, 173 rewrite rules, and 12 deduction rules.

Ordered equation Induct.2 into the rewrite rule:
  ((DEQ(c_xh4) = null) <=> false)
   | ((DEQ(cons(c_xh3, vi1)) = null) <=> false)
   | (DEQ(append(cons(c_xh3, vi1), c_xh4)) = null)
  -> true

The system now contains 174 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma3.6.2.2: ((DEQ(cons(c_xh3, vi1)) = null) & (DEQ(cons(c_xh4, vi2)) = null))
               => (DEQ(append(cons(c_xh3, vi1), cons(c_xh4, vi2))) = null)
              -> true
                  which reduces to the equation
                  ((DEQ(cons(c_xh3, vi1)) = null) <=> false)
                   | ((DEQ(cons(c_xh4, vi2)) = null) <=> false)
                   | (DEQ(cons(append(cons(c_xh3, vi1), c_xh4), vi2)) = null)
                  -> true

Proof of Lemma lemma3.6.2.2 suspended.

-> resume by induction vi2 Ev

The basis step in an inductive proof of Lemma lemma3.6.2.2 for the induction
step in the proof of Lemma lemma3.6.2
    ((DEQ(cons(c_xh3, vi1)) = null) & (DEQ(cons(c_xh4, vi2)) = null))
     => (DEQ(append(cons(c_xh3, vi1), cons(c_xh4, vi2))) = null)
    -> true
involves proving the following lemma(s):
```

```
lemma3.6.2.2.1: ((DEQ(cons(c_xh3, vi1)) = null)
                  & (DEQ(cons(c_xh4, E(vi3))) = null))
                 => (DEQ(append(cons(c_xh3, vi1), cons(c_xh4, E(vi3)))) = null)
                -> true
                [] Proved by normalization
lemma3.6.2.2.2: ((DEQ(cons(c_xh3, vi1)) = null)
                  & (DEQ(cons(c_xh4, D(vi3))) = null))
                 => (DEQ(append(cons(c_xh3, vi1), cons(c_xh4, D(vi3)))) = null)
                -> true
                [] Proved by normalization

The induction step in an inductive proof of Lemma lemma3.6.2.2 for the
induction step in the proof of Lemma lemma3.6.2
    ((DEQ(cons(c_xh3, vi1)) = null) & (DEQ(cons(c_xh4, vi2)) = null))
     => (DEQ(append(cons(c_xh3, vi1), cons(c_xh4, vi2))) = null)
    -> true
is vacuous.

Lemma lemma3.6.2.2 for the induction step in the proof of Lemma lemma3.6.2
    ((DEQ(cons(c_xh3, vi1)) = null) & (DEQ(cons(c_xh4, vi2)) = null))
     => (DEQ(append(cons(c_xh3, vi1), cons(c_xh4, vi2))) = null)
    -> true
[] Proved by induction over 'vi2::Ev' of sort 'Ev'.

Lemma lemma3.6.2 for the induction step in the proof of Conjecture lemma3.6
    ((DEQ(cons(c_xh3, vi1)) = null) & (DEQ(xh2) = null))
     => (DEQ(append(cons(c_xh3, vi1), xh2)) = null)
    -> true
[] Proved by induction over 'xh2' of sort 'H'.

Conjecture lemma3.6
    ((DEQ(xh1) = null) & (DEQ(xh2) = null)) => (DEQ(append(xh1, xh2)) = null)
    -> true
[] Proved by induction over 'xh1' of sort 'H'.

The system now contains 1 equation, 172 rewrite rules, and 12 deduction rules.

Ordered equation lemma3.6 into the rewrite rule:
  ((DEQ(xh1) = null) <=> false)
   | ((DEQ(xh2) = null) <=> false)
   | (DEQ(append(xh1, xh2)) = null)
  -> true

The system now contains 173 rewrite rules and 12 deduction rules.

Lemma lemma3.2.1 in the proof by cases of Conjecture lemma3.2
    ((DEQ(c_xh1) = null)
      & (DEQ(c_xh2) = null)
      & (append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))), c_xh2)
          = c_xh)

      & in(append(c_xh1, c_xh2), af(c_xst))
      & in(c_xn, enqd(c_xst))
      & least(c_xn, enqd(c_xst)))
     => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true
    Case.1.1: (DEQ(c_xh1) = null)
              & (DEQ(c_xh2) = null)
              & (append(cons(c_xh1,
                             D(trip(element(c_xn), enqt(c_xn), c_xt1))),

                  c_xh2)
                 = c_xh)

              & in(append(c_xh1, c_xh2), af(c_xst))
              & in(c_xn, enqd(c_xst))
              & least(c_xn, enqd(c_xst))

is NOT provable using the current partially completed system.  It reduces to
```

```
the equation
    prefix(cons(null, element(c_xn)), ENQ(c_xh)) -> true

Proof of Lemma lemma3.2.1 suspended.

-> crit case with lemma3.6

Critical pairs between rule Case.1.1.7.1:
  DEQ(c_xh1) -> null
and rule lemma3.6:
  ((DEQ(xh1) = null) <=> false)
   | ((DEQ(xh2) = null) <=> false)
   | (DEQ(append(xh1, xh2)) = null)
  -> true
  are as follows:
    ((DEQ(xh2) = null) <=> false) | (DEQ(append(c_xh1, xh2)) = null) == true
    ((DEQ(xh1) = null) <=> false) | (DEQ(append(xh1, c_xh1)) = null) == true

The system now contains 1 equation, 173 rewrite rules, and 12 deduction rules.

Ordered equation lemma3.7 into the rewrite rule:
  ((DEQ(xh2) = null) <=> false) | (DEQ(append(c_xh1, xh2)) = null) -> true

The system now contains 174 rewrite rules and 12 deduction rules.

The system now contains 1 equation, 174 rewrite rules, and 12 deduction rules.

Ordered equation lemma3.8 into the rewrite rule:
  ((DEQ(xh1) = null) <=> false) | (DEQ(append(xh1, c_xh1)) = null) -> true

The system now contains 175 rewrite rules and 12 deduction rules.

Critical pairs between rule Case.1.1.8.1:
  DEQ(c_xh2) -> null
and rule lemma3.6:
  ((DEQ(xh1) = null) <=> false)
   | ((DEQ(xh2) = null) <=> false)
   | (DEQ(append(xh1, xh2)) = null)
  -> true
  are as follows:
    ((DEQ(xh2) = null) <=> false) | (DEQ(append(c_xh2, xh2)) = null) == true
    ((DEQ(xh1) = null) <=> false) | (DEQ(append(xh1, c_xh2)) = null) == true

The system now contains 1 equation, 175 rewrite rules, and 12 deduction rules.

Ordered equation lemma3.9 into the rewrite rule:
  ((DEQ(xh2) = null) <=> false) | (DEQ(append(c_xh2, xh2)) = null) -> true

The system now contains 176 rewrite rules and 12 deduction rules.

The system now contains 1 equation, 176 rewrite rules, and 12 deduction rules.

Ordered equation lemma3.10 into the rewrite rule:
  ((DEQ(xh1) = null) <=> false) | (DEQ(append(xh1, c_xh2)) = null) -> true

The system now contains 177 rewrite rules and 12 deduction rules.

Computed 5 new critical pairs, 1 of which reduced to an identity.  Added 4 of
them to the system.


-> crit case.1.1.7.1 with lemma3.10

Critical pairs between rule Case.1.1.7.1:
  DEQ(c_xh1) -> null
and rule lemma3.10:
  ((DEQ(xh1) = null) <=> false) | (DEQ(append(xh1, c_xh2)) = null) -> true
  are as follows:
    DEQ(append(c_xh1, c_xh2)) = null == true
```

```
The system now contains 1 equation, 177 rewrite rules, and 12 deduction rules.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation lemma3.11:
  DEQ(append(c_xh1, c_xh2)) = null == true
to yield the following equations:
  lemma3.11.1: DEQ(append(c_xh1, c_xh2)) == null

Ordered equation lemma3.11.1 into the rewrite rule:
  DEQ(append(c_xh1, c_xh2)) -> null

    Left-hand side reduced:
    (false <=> prefix(DEQ(append(c_xh1, c_xh2)), ENQ(c_xh)))
     | prefix(cons(DEQ(append(c_xh1, c_xh2)), element(c_xn)), ENQ(c_xh))
    -> true
      became equation lemma3.5.1:
      (false <=> prefix(null, ENQ(c_xh)))
       | prefix(cons(DEQ(append(c_xh1, c_xh2)), element(c_xn)), ENQ(c_xh))
      == true

Ordered equation lemma3.5.1 into the rewrite rule:
  prefix(cons(null, element(c_xn)), ENQ(c_xh)) -> true

The system now contains 178 rewrite rules and 12 deduction rules.

Lemma lemma3.2.1 in the proof by cases of Conjecture lemma3.2
    ((DEQ(c_xh1) = null)
      & (DEQ(c_xh2) = null)
      & (append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xtl))), c_xh2)
          = c_xh)

      & in(append(c_xh1, c_xh2), af(c_xst))
      & in(c_xn, enqd(c_xst))
      & least(c_xn, enqd(c_xst)))
     => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true
    Case.1.1: (DEQ(c_xh1) = null)
              & (DEQ(c_xh2) = null)
              & (append(cons(c_xh1,
                             D(trip(element(c_xn), enqt(c_xn), c_xtl))),

                   c_xh2)
                  = c_xh)

              & in(append(c_xh1, c_xh2), af(c_xst))
              & in(c_xn, enqd(c_xst))
              & least(c_xn, enqd(c_xst))

[] Proved by rewriting.

Case.1.2
    not((DEQ(c_xh1) = null)
         & (DEQ(c_xh2) = null)
         & (append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xtl))),
                    c_xh2)
             = c_xh)

         & in(append(c_xh1, c_xh2), af(c_xst))
         & in(c_xn, enqd(c_xst))
         & least(c_xn, enqd(c_xst)))

    == true
involves proving Lemma lemma3.2.2
    ((DEQ(c_xh1) = null)
      & (DEQ(c_xh2) = null)
      & (append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xtl))), c_xh2)
```

```
          = c_xh)

      & in(append(c_xh1, c_xh2), af(c_xst))
      & in(c_xn, enqd(c_xst))
      & least(c_xn, enqd(c_xst)))
     => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true

The case system now contains 1 equation.

Ordered equation Case.1.2 into the rewrite rule:
   ((DEQ(c_xh1) = null) <=> false)
    | ((DEQ(c_xh2) = null) <=> false)
    | ((append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))), c_xh2)
         = c_xh)
        <=> false)

    | (false <=> in(append(c_xh1, c_xh2), af(c_xst)))
    | (false <=> in(c_xn, enqd(c_xst)))
    | (false <=> least(c_xn, enqd(c_xst)))
   -> true

The case system now contains 1 rewrite rule.

Lemma lemma3.2.2 in the proof by cases of Conjecture lemma3.2
    ((DEQ(c_xh1) = null)
      & (DEQ(c_xh2) = null)
      & (append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))), c_xh2)
          = c_xh)

      & in(append(c_xh1, c_xh2), af(c_xst))
      & in(c_xn, enqd(c_xst))
      & least(c_xn, enqd(c_xst)))
     => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true
    Case.1.2: not((DEQ(c_xh1) = null)
                  & (DEQ(c_xh2) = null)
                  & (append(cons(c_xh1,
                                 D(trip(element(c_xn), enqt(c_xn), c_xt1))),

                        c_xh2)
                      = c_xh)

                  & in(append(c_xh1, c_xh2), af(c_xst))
                  & in(c_xn, enqd(c_xst))
                  & least(c_xn, enqd(c_xst)))


[] Proved by rewriting (with unreduced rules).

Conjecture lemma3.2
    ((DEQ(xh1) = null)
      & (DEQ(xh2) = null)
      & (append(cons(xh1, D(trip(element(xn), enqt(xn), xt))), xh2) = xh)
      & in(append(xh1, xh2), af(xst))
      & in(xn, enqd(xst))
      & least(xn, enqd(xst)))
     => prefix(DEQ(xh), ENQ(xh))
    -> true
[] Proved by cases
    ((DEQ(xh1) = null)
      & (DEQ(xh2) = null)
      & (append(cons(xh1, D(trip(element(xn), enqt(xn), xt))), xh2) = xh)
      & in(append(xh1, xh2), af(xst))
      & in(xn, enqd(xst))
      & least(xn, enqd(xst)))
     | not((DEQ(xh1) = null)
           & (DEQ(xh2) = null)
           & (append(cons(xh1, D(trip(element(xn), enqt(xn), xt))), xh2) = xh)
```

```
            & in(append(xh1, xh2), af(xst))
            & in(xn, enqd(xst))
            & least(xn, enqd(xst)))



The system now contains 1 equation, 162 rewrite rules, and 12 deduction rules.

Ordered equation lemma3.2 into the rewrite rule:
  ((DEQ(xh1) = null) <=> false)
   | ((DEQ(xh2) = null) <=> false)
   | ((append(cons(xh1, D(trip(element(xn), enqt(xn), xt))), xh2) = xh)
       <=> false)

   | (false <=> in(append(xh1, xh2), af(xst)))
   | (false <=> in(xn, enqd(xst)))
   | (false <=> least(xn, enqd(xst)))
   | prefix(DEQ(xh), ENQ(xh))
  -> true

The system now contains 163 rewrite rules and 12 deduction rules.

Critical-pair computation abandoned because a theorem has been proved.

Computed 1 new critical pair.  Added 1 of them to the system.

-> prove
Please enter an equation to prove, terminated with a `..' line, or `?' for help:
((xh=append(cons(xh1, D(trip(element(xn), enqt(xn), xt))), xh2)) &
 in(append(xh1,xh2),af(xst)) & in(xn,enqd(xst)) & least(xn,enqd(xst)) &
 prefix(DEQ(append(xh1, xh2)), ENQ(append(xh1, xh2)))&(DEQ(xh2)=null:->Seq)&
 (enqr(top(deqd(xst))) < enqt(xn))) => prefix(DEQ(xh),ENQ(xh))
..


Conjecture lemma3.12
     ((enqr(top(deqd(xst))) < enqt(xn))
       & (DEQ(xh2) = null)
       & (append(cons(xh1, D(trip(element(xn), enqt(xn), xt))), xh2) = xh)
       & in(append(xh1, xh2), af(xst))
       & in(xn, enqd(xst))
       & least(xn, enqd(xst))
       & prefix(DEQ(append(xh1, xh2)), ENQ(append(xh1, xh2))))
      => prefix(DEQ(xh), ENQ(xh))
     -> true
is NOT provable using the current partially completed system.  It reduces to
the equation
     ((enqr(top(deqd(xst))) < enqt(xn)) <=> false)
      | ((DEQ(xh2) = null) <=> false)
      | ((append(cons(xh1, D(trip(element(xn), enqt(xn), xt))), xh2) = xh)
          <=> false)

      | (false <=> in(append(xh1, xh2), af(xst)))
      | (false <=> in(xn, enqd(xst)))
      | (false <=> least(xn, enqd(xst)))
      | (false <=> prefix(DEQ(append(xh1, xh2)), ENQ(append(xh1, xh2))))
      | prefix(DEQ(xh), ENQ(xh))
     -> true

Proof of Conjecture lemma3.12 suspended.

-> resume by case (enqr(top(deqd(xst)))<enqt(xn))&(DEQ(xh2) = null:->Seq)&
   (append(cons(xh1,D(trip(element(xn),enqt(xn),xt))),xh2)=xh)&
  in(append(xh1,xh2),af(xst))&in(xn,enqd(xst))&least(xn,enqd(xst))&
  prefix(DEQ(append(xh1, xh2)), ENQ(append(xh1, xh2)))

Case.2.1
     (enqr(top(deqd(c_xst))) < enqt(c_xn))
```

```
     & (DEQ(c_xh2) = null)
     & (append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))), c_xh2)
          = c_xh)

     & in(append(c_xh1, c_xh2), af(c_xst))
     & in(c_xn, enqd(c_xst))
     & least(c_xn, enqd(c_xst))
     & prefix(DEQ(append(c_xh1, c_xh2)), ENQ(append(c_xh1, c_xh2)))
    == true
involves proving Lemma lemma3.12.1
    ((enqr(top(deqd(c_xst))) < enqt(c_xn))
      & (DEQ(c_xh2) = null)
      & (append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))), c_xh2)
           = c_xh)

      & in(append(c_xh1, c_xh2), af(c_xst))
      & in(c_xn, enqd(c_xst))
      & least(c_xn, enqd(c_xst))
      & prefix(DEQ(append(c_xh1, c_xh2)), ENQ(append(c_xh1, c_xh2))))
     => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true

The case system now contains 1 equation.

Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation Case.2.1:
  (enqr(top(deqd(c_xst))) < enqt(c_xn))
   & (DEQ(c_xh2) = null)
   & (append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))), c_xh2)
       = c_xh)

   & in(append(c_xh1, c_xh2), af(c_xst))
   & in(c_xn, enqd(c_xst))
   & least(c_xn, enqd(c_xst))
   & prefix(DEQ(append(c_xh1, c_xh2)), ENQ(append(c_xh1, c_xh2)))
  == true
to yield the following equations:
  Case.2.1.1: enqr(top(deqd(c_xst))) < enqt(c_xn) == true
  Case.2.1.2: DEQ(c_xh2) = null == true
  Case.2.1.3: append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))),
                     c_xh2)
               = c_xh
              == true
  Case.2.1.4: in(append(c_xh1, c_xh2), af(c_xst)) == true
  Case.2.1.5: in(c_xn, enqd(c_xst)) == true
  Case.2.1.6: least(c_xn, enqd(c_xst)) == true
  Case.2.1.7: prefix(DEQ(append(c_xh1, c_xh2)), ENQ(append(c_xh1, c_xh2)))
              == true

Ordered equation Case.2.1.7 into the rewrite rule:
  prefix(DEQ(append(c_xh1, c_xh2)), ENQ(append(c_xh1, c_xh2))) -> true

Ordered equation Case.2.1.6 into the rewrite rule:
  least(c_xn, enqd(c_xst)) -> true

Ordered equation Case.2.1.5 into the rewrite rule:
  in(c_xn, enqd(c_xst)) -> true

Ordered equation Case.2.1.4 into the rewrite rule:
  in(append(c_xh1, c_xh2), af(c_xst)) -> true

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.2.1.3:
  append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))), c_xh2) = c_xh
```

```
  == true
to yield the following equations:
  Case.2.1.3.1: append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))),
                       c_xh2)
                == c_xh

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.2.1.2:
  DEQ(c_xh2) = null == true
to yield the following equations:
  Case.2.1.2.1: DEQ(c_xh2) == null

Ordered equation Case.2.1.1 into the rewrite rule:
  enqr(top(deqd(c_xst))) < enqt(c_xn) -> true

The case system now contains 2 equations and 5 rewrite rules.

Ordered equation Case.2.1.2.1 into the rewrite rule:
  DEQ(c_xh2) -> null

The case system now contains 1 equation and 6 rewrite rules.

Ordered equation Case.2.1.3.1 into the rewrite rule:
  append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))), c_xh2) -> c_xh

The case system now contains 7 rewrite rules.

The system now contains 1 equation, 163 rewrite rules, and 12 deduction rules.

Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation Case.2.1:
  (enqr(top(deqd(c_xst))) < enqt(c_xn))
   & (DEQ(c_xh2) = null)
   & (append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))), c_xh2)
       = c_xh)

   & in(append(c_xh1, c_xh2), af(c_xst))
   & in(c_xn, enqd(c_xst))
   & least(c_xn, enqd(c_xst))
   & prefix(DEQ(append(c_xh1, c_xh2)), ENQ(append(c_xh1, c_xh2)))
  == true
to yield the following equations:
  Case.2.1.8: enqr(top(deqd(c_xst))) < enqt(c_xn) == true
  Case.2.1.9: DEQ(c_xh2) = null == true
  Case.2.1.10: append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))),
                      c_xh2)
               = c_xh
              == true
  Case.2.1.11: in(append(c_xh1, c_xh2), af(c_xst)) == true
  Case.2.1.12: in(c_xn, enqd(c_xst)) == true
  Case.2.1.13: least(c_xn, enqd(c_xst)) == true
  Case.2.1.14: prefix(DEQ(append(c_xh1, c_xh2)), ENQ(append(c_xh1, c_xh2)))
              == true

Ordered equation Case.2.1.14 into the rewrite rule:
  prefix(DEQ(append(c_xh1, c_xh2)), ENQ(append(c_xh1, c_xh2))) -> true

Ordered equation Case.2.1.13 into the rewrite rule:
  least(c_xn, enqd(c_xst)) -> true

Ordered equation Case.2.1.12 into the rewrite rule:
  in(c_xn, enqd(c_xst)) -> true

Ordered equation Case.2.1.11 into the rewrite rule:
```

```
  in(append(c_xh1, c_xh2), af(c_xst)) -> true

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.2.1.10:
  append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))), c_xh2) = c_xh
  == true
to yield the following equations:
  Case.2.1.10.1: append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))),
                        c_xh2)
                 == c_xh

Ordered equation Case.2.1.10.1 into the rewrite rule:
  append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))), c_xh2) -> c_xh

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.2.1.9:
  DEQ(c_xh2) = null == true
to yield the following equations:
  Case.2.1.9.1: DEQ(c_xh2) == null

Ordered equation Case.2.1.9.1 into the rewrite rule:
  DEQ(c_xh2) -> null

Ordered equation Case.2.1.8 into the rewrite rule:
  enqr(top(deqd(c_xst))) < enqt(c_xn) -> true

The system now contains 170 rewrite rules and 12 deduction rules.

Lemma lemma3.12.1 in the proof by cases of Conjecture lemma3.12
    ((enqr(top(deqd(c_xst))) < enqt(c_xn))
      & (DEQ(c_xh2) = null)
      & (append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))), c_xh2)
          = c_xh)

      & in(append(c_xh1, c_xh2), af(c_xst))
      & in(c_xn, enqd(c_xst))
      & least(c_xn, enqd(c_xst))
      & prefix(DEQ(append(c_xh1, c_xh2)), ENQ(append(c_xh1, c_xh2))))
     => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true
    Case.2.1:  (enqr(top(deqd(c_xst))) < enqt(c_xn))
               & (DEQ(c_xh2) = null)
               & (append(cons(c_xh1,
                              D(trip(element(c_xn), enqt(c_xn), c_xt1))),

                    c_xh2)
                   = c_xh)

               & in(append(c_xh1, c_xh2), af(c_xst))
               & in(c_xn, enqd(c_xst))
               & least(c_xn, enqd(c_xst))
               & prefix(DEQ(append(c_xh1, c_xh2)), ENQ(append(c_xh1, c_xh2)))

is NOT provable using the current partially completed system.  It reduces to
the equation
    prefix(DEQ(c_xh), ENQ(c_xh)) -> true

Proof of Lemma lemma3.12.1 suspended.

-> crit case with lemma1.7

Critical pairs between rule Case.2.1.10.1:
  append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))), c_xh2) -> c_xh
and rule lemma1.7:
  ENQ(append(cons(x, D(y)), z)) -> ENQ(append(x, z))
```

```
  are as follows:
    ENQ(c_xh) == ENQ(append(c_xh1, c_xh2))

The system now contains 1 equation, 170 rewrite rules, and 12 deduction rules.

Ordered equation lemma3.13 into the rewrite rule:
  ENQ(append(c_xh1, c_xh2)) -> ENQ(c_xh)

    Left-hand side reduced:
    prefix(DEQ(append(c_xh1, c_xh2)), ENQ(append(c_xh1, c_xh2))) -> true
      became equation Case.2.1.14:
      prefix(DEQ(append(c_xh1, c_xh2)), ENQ(c_xh)) == true

Ordered equation Case.2.1.14 into the rewrite rule:
  prefix(DEQ(append(c_xh1, c_xh2)), ENQ(c_xh)) -> true

The system now contains 171 rewrite rules and 12 deduction rules.

Computed 1 new critical pair.  Added 1 of them to the system.

-> crit case with lemma1.9

Critical pairs between rule Case.2.1.10.1:
  append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))), c_xh2) -> c_xh
and rule lemma1.9:
  DEQ(append(cons(x, D(y)), z)) -> append(cons(DEQ(x), what(y)), DEQ(z))
  are as follows:
    DEQ(c_xh) == cons(DEQ(c_xh1), element(c_xn))

The system now contains 1 equation, 171 rewrite rules, and 12 deduction rules.

Ordered equation lemma3.14 into the rewrite rule:
  cons(DEQ(c_xh1), element(c_xn)) -> DEQ(c_xh)

The system now contains 172 rewrite rules and 12 deduction rules.

Computed 1 new critical pair.  Added 1 of them to the system.

-> crit case.2.1.12 with Abstraction.11

Critical pairs between rule Case.2.1.12:
  in(c_xn, enqd(c_xst)) -> true
and rule Abstraction.11:
  (false <=> in(xh, af(xst)))
   | (false <=> in(xn, enqd(xst)))
   | (false <=> least(xn, enqd(xst)))
   | (false <=> prefix(DEQ(xh), ENQ(xh)))
   | prefix(cons(DEQ(xh), element(xn)), ENQ(xh))
  -> true
  are as follows:
    (false <=> in(xh, af(c_xst)))
     | (false <=> prefix(DEQ(xh), ENQ(xh)))
     | prefix(cons(DEQ(xh), element(c_xn)), ENQ(xh))
    == true

The system now contains 1 equation, 172 rewrite rules, and 12 deduction rules.

Ordered equation lemma3.15 into the rewrite rule:
  (false <=> in(xh, af(c_xst)))
   | (false <=> prefix(DEQ(xh), ENQ(xh)))
   | prefix(cons(DEQ(xh), element(c_xn)), ENQ(xh))
  -> true

The system now contains 173 rewrite rules and 12 deduction rules.

Computed 1 new critical pair.  Added 1 of them to the system.

-> instantiate xh by append(c_xh1,c_xh2) in lemma3.15
```

```
Equation lemma3.15:
  (false <=> in(xh, af(c_xst)))
   | (false <=> prefix(DEQ(xh), ENQ(xh)))
   | prefix(cons(DEQ(xh), element(c_xn)), ENQ(xh))
  -> true
has been instantiated to equation lemma3.15.1:
  prefix(cons(DEQ(append(c_xh1, c_xh2)), element(c_xn)), ENQ(c_xh)) == true

Added 1 equation to the system.

Ordered equation lemma3.15.1 into the rewrite rule:
  prefix(cons(DEQ(append(c_xh1, c_xh2)), element(c_xn)), ENQ(c_xh)) -> true

The system now contains 174 rewrite rules and 12 deduction rules.

-> prove (DEQ(xh2)=null:->Seq)=>(DEQ(append(xh1,xh2))=DEQ(xh1)) by induction xh2 H

The basis step in an inductive proof of Conjecture lemma3.16
    (DEQ(xh2) = null) => (DEQ(append(xh1, xh2)) = DEQ(xh1)) -> true
involves proving the following lemma(s):

lemma3.16.1: (DEQ(null) = null) => (DEQ(append(xh1, null)) = DEQ(xh1)) -> true
             [] Proved by normalization

The induction step in an inductive proof of Conjecture lemma3.16
    (DEQ(xh2) = null) => (DEQ(append(xh1, xh2)) = DEQ(xh1)) -> true
uses the following equation(s) for the induction hypothesis:

Induct.3: (DEQ(c_xh3) = null) => (DEQ(append(xh1, c_xh3)) = DEQ(xh1)) -> true

The system now contains 1 equation, 174 rewrite rules, and 12 deduction rules.

Ordered equation Induct.3 into the rewrite rule:
  ((DEQ(c_xh3) = null) <=> false) | (DEQ(append(xh1, c_xh3)) = DEQ(xh1))
  -> true

The system now contains 175 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

lemma3.16.2: (DEQ(cons(c_xh3, vi1)) = null)
              => (DEQ(append(xh1, cons(c_xh3, vi1))) = DEQ(xh1))
             -> true
                 which reduces to the equation
                 ((DEQ(cons(c_xh3, vi1)) = null) <=> false)
                  | (DEQ(cons(append(xh1, c_xh3), vi1)) = DEQ(xh1))
                 -> true

Proof of Lemma lemma3.16.2 suspended.

-> resume by induction vi1 Ev

The basis step in an inductive proof of Lemma lemma3.16.2 for the induction
step in the proof of Conjecture lemma3.16
    (DEQ(cons(c_xh3, vi1)) = null)
     => (DEQ(append(xh1, cons(c_xh3, vi1))) = DEQ(xh1))
    -> true
involves proving the following lemma(s):

lemma3.16.2.1: (DEQ(cons(c_xh3, E(vi2))) = null)
                => (DEQ(append(xh1, cons(c_xh3, E(vi2)))) = DEQ(xh1))
               -> true
               [] Proved by normalization
lemma3.16.2.2: (DEQ(cons(c_xh3, D(vi2))) = null)
                => (DEQ(append(xh1, cons(c_xh3, D(vi2)))) = DEQ(xh1))
               -> true
               [] Proved by normalization
```

```
The induction step in an inductive proof of Lemma lemma3.16.2 for the induction
step in the proof of Conjecture lemma3.16
    (DEQ(cons(c_xh3, vil)) = null)
     => (DEQ(append(xh1, cons(c_xh3, vil))) = DEQ(xh1))
    -> true
is vacuous.

Lemma lemma3.16.2 for the induction step in the proof of Conjecture lemma3.16
    (DEQ(cons(c_xh3, vil)) = null)
     => (DEQ(append(xh1, cons(c_xh3, vil))) = DEQ(xh1))
    -> true
[] Proved by induction over 'vil::Ev' of sort 'Ev'.

Conjecture lemma3.16
    (DEQ(xh2) = null) => (DEQ(append(xh1, xh2)) = DEQ(xh1)) -> true
[] Proved by induction over 'xh2' of sort 'H'.

The system now contains 1 equation, 174 rewrite rules, and 12 deduction rules.

Ordered equation lemma3.16 into the rewrite rule:
  ((DEQ(xh2) = null) <=> false) | (DEQ(append(xh1, xh2)) = DEQ(xh1)) -> true

The system now contains 175 rewrite rules and 12 deduction rules.

Lemma lemma3.12.1 in the proof by cases of Conjecture lemma3.12
    ((enqr(top(deqd(c_xst))) < enqt(c_xn))
      & (DEQ(c_xh2) = null)
      & (append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xtl))), c_xh2)
          = c_xh)

      & in(append(c_xh1, c_xh2), af(c_xst))
      & in(c_xn, enqd(c_xst))
      & least(c_xn, enqd(c_xst))
      & prefix(DEQ(append(c_xh1, c_xh2)), ENQ(append(c_xh1, c_xh2))))
     => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true
    Case.2.1: (enqr(top(deqd(c_xst))) < enqt(c_xn))
              & (DEQ(c_xh2) = null)
              & (append(cons(c_xh1,
                             D(trip(element(c_xn), enqt(c_xn), c_xtl))),

                    c_xh2)
                   = c_xh)

              & in(append(c_xh1, c_xh2), af(c_xst))
              & in(c_xn, enqd(c_xst))
              & least(c_xn, enqd(c_xst))
              & prefix(DEQ(append(c_xh1, c_xh2)), ENQ(append(c_xh1, c_xh2)))

is NOT provable using the current partially completed system.  It reduces to
the equation
    prefix(DEQ(c_xh), ENQ(c_xh)) -> true

Proof of Lemma lemma3.12.1 suspended.

-> instantiate xh1 by c_xh1, xh2 by c_xh2 in lemma3.16


Equation lemma3.16:
  ((DEQ(xh2) = null) <=> false) | (DEQ(append(xh1, xh2)) = DEQ(xh1)) -> true
has been instantiated to equation lemma3.16.3:
  DEQ(append(c_xh1, c_xh2)) = DEQ(c_xh1) -> true

Added 1 equation to the system.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation lemma3.16.3:
```

```
  DEQ(append(c_xh1, c_xh2)) = DEQ(c_xh1) -> true
to yield the following equations:
  lemma3.16.3.1: DEQ(append(c_xh1, c_xh2)) == DEQ(c_xh1)

Ordered equation lemma3.16.3.1 into the rewrite rule:
  DEQ(append(c_xh1, c_xh2)) -> DEQ(c_xh1)

    Following 2 left-hand sides reduced:
    prefix(DEQ(append(c_xh1, c_xh2)), ENQ(c_xh)) -> true
      became equation Case.2.1.14:
      prefix(DEQ(c_xh1), ENQ(c_xh)) == true
    prefix(cons(DEQ(append(c_xh1, c_xh2)), element(c_xn)), ENQ(c_xh)) -> true
      became equation lemma3.15.1:
      prefix(cons(DEQ(c_xh1), element(c_xn)), ENQ(c_xh)) == true

Ordered equation Case.2.1.14 into the rewrite rule:
  prefix(DEQ(c_xh1), ENQ(c_xh)) -> true

Ordered equation lemma3.15.1 into the rewrite rule:
  prefix(DEQ(c_xh), ENQ(c_xh)) -> true

The system now contains 176 rewrite rules and 12 deduction rules.

Lemma lemma3.12.1 in the proof by cases of Conjecture lemma3.12
     ((enqr(top(deqd(c_xst))) < enqt(c_xn))
       & (DEQ(c_xh2) = null)
       & (append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))), c_xh2)
           = c_xh)

       & in(append(c_xh1, c_xh2), af(c_xst))
       & in(c_xn, enqd(c_xst))
       & least(c_xn, enqd(c_xst))
       & prefix(DEQ(append(c_xh1, c_xh2)), ENQ(append(c_xh1, c_xh2))))
      => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true
    Case.2.1: (enqr(top(deqd(c_xst))) < enqt(c_xn))
              & (DEQ(c_xh2) = null)
              & (append(cons(c_xh1,
                             D(trip(element(c_xn), enqt(c_xn), c_xt1))),

                    c_xh2)
                   = c_xh)

              & in(append(c_xh1, c_xh2), af(c_xst))
              & in(c_xn, enqd(c_xst))
              & least(c_xn, enqd(c_xst))
              & prefix(DEQ(append(c_xh1, c_xh2)), ENQ(append(c_xh1, c_xh2)))

[] Proved by rewriting.

Case.2.2
    not((enqr(top(deqd(c_xst))) < enqt(c_xn))
         & (DEQ(c_xh2) = null)
         & (append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))),
                  c_xh2)
              = c_xh)

         & in(append(c_xh1, c_xh2), af(c_xst))
         & in(c_xn, enqd(c_xst))
         & least(c_xn, enqd(c_xst))
         & prefix(DEQ(append(c_xh1, c_xh2)), ENQ(append(c_xh1, c_xh2))))

    == true
involves proving Lemma lemma3.12.2
     ((enqr(top(deqd(c_xst))) < enqt(c_xn))
       & (DEQ(c_xh2) = null)
       & (append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))), c_xh2)
           = c_xh)
```

```
      & in(append(c_xh1, c_xh2), af(c_xst))
      & in(c_xn, enqd(c_xst))
      & least(c_xn, enqd(c_xst))
      & prefix(DEQ(append(c_xh1, c_xh2)), ENQ(append(c_xh1, c_xh2))))
     => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true

The case system now contains 1 equation.

Ordered equation Case.2.2 into the rewrite rule:
  ((enqr(top(deqd(c_xst))) < enqt(c_xn)) <=> false)
   | ((DEQ(c_xh2) = null) <=> false)
   | ((append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))), c_xh2)
        = c_xh)
       <=> false)

   | (false <=> in(append(c_xh1, c_xh2), af(c_xst)))
   | (false <=> in(c_xn, enqd(c_xst)))
   | (false <=> least(c_xn, enqd(c_xst)))
   | (false <=> prefix(DEQ(append(c_xh1, c_xh2)), ENQ(append(c_xh1, c_xh2))))
  -> true

The case system now contains 1 rewrite rule.

Lemma lemma3.12.2 in the proof by cases of Conjecture lemma3.12
    ((enqr(top(deqd(c_xst))) < enqt(c_xn))
      & (DEQ(c_xh2) = null)
      & (append(cons(c_xh1, D(trip(element(c_xn), enqt(c_xn), c_xt1))), c_xh2)
          = c_xh)

      & in(append(c_xh1, c_xh2), af(c_xst))
      & in(c_xn, enqd(c_xst))
      & least(c_xn, enqd(c_xst))
      & prefix(DEQ(append(c_xh1, c_xh2)), ENQ(append(c_xh1, c_xh2))))
     => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true
    Case.2.2: not((enqr(top(deqd(c_xst))) < enqt(c_xn))
                  & (DEQ(c_xh2) = null)
                  & (append(cons(c_xh1,
                                 D(trip(element(c_xn), enqt(c_xn), c_xt1))),

                         c_xh2)
                        = c_xh)

                  & in(append(c_xh1, c_xh2), af(c_xst))
                  & in(c_xn, enqd(c_xst))
                  & least(c_xn, enqd(c_xst))
                  & prefix(DEQ(append(c_xh1, c_xh2)),
                           ENQ(append(c_xh1, c_xh2))))


[] Proved by rewriting (with unreduced rules).

Conjecture lemma3.12
    ((enqr(top(deqd(xst))) < enqt(xn))
      & (DEQ(xh2) = null)
      & (append(cons(xh1, D(trip(element(xn), enqt(xn), xt))), xh2) = xh)
      & in(append(xh1, xh2), af(xst))
      & in(xn, enqd(xst))
      & least(xn, enqd(xst))
      & prefix(DEQ(append(xh1, xh2)), ENQ(append(xh1, xh2))))
     => prefix(DEQ(xh), ENQ(xh))
    -> true
[] Proved by cases
    ((enqr(top(deqd(xst))) < enqt(xn))
      & (DEQ(xh2) = null)
      & (append(cons(xh1, D(trip(element(xn), enqt(xn), xt))), xh2) = xh)
      & in(append(xh1, xh2), af(xst))
```

```
      & in(xn, enqd(xst))
      & least(xn, enqd(xst))
      & prefix(DEQ(append(xh1, xh2)), ENQ(append(xh1, xh2))))
     | not((enqr(top(deqd(xst))) < enqt(xn))
            & (DEQ(xh2) = null)
            & (append(cons(xh1, D(trip(element(xn), enqt(xn), xt))), xh2) = xh)
            & in(append(xh1, xh2), af(xst))
            & in(xn, enqd(xst))
            & least(xn, enqd(xst))
            & prefix(DEQ(append(xh1, xh2)), ENQ(append(xh1, xh2))))



The system now contains 1 equation, 163 rewrite rules, and 12 deduction rules.

Ordered equation lemma3.12 into the rewrite rule:
   ((enqr(top(deqd(xst))) < enqt(xn)) <=> false)
    | ((DEQ(xh2) = null) <=> false)
    | ((append(cons(xh1, D(trip(element(xn), enqt(xn), xt))), xh2) = xh)
        <=> false)

    | (false <=> in(append(xh1, xh2), af(xst)))
    | (false <=> in(xn, enqd(xst)))
    | (false <=> least(xn, enqd(xst)))
    | (false <=> prefix(DEQ(append(xh1, xh2)), ENQ(append(xh1, xh2))))
    | prefix(DEQ(xh), ENQ(xh))
   -> true

The system now contains 164 rewrite rules and 12 deduction rules.

-> qed

All conjectures have been proved.

-> freeze theory3
```

6. LP Proof of Correctness Condition

The prefix property is stated in the fourth line below.

```
-> thaw theory3

System thawed from 'theory3.frz'.

-> set name sync

The name prefix is now 'sync'.

-> prove in(xh, af(xst))=>prefix(DEQ(xh),ENQ(xh)) by induction xst St

The basis step in an inductive proof of Conjecture sync.1
    in(xh, af(xst)) => prefix(DEQ(xh), ENQ(xh)) -> true
involves proving the following lemma(s):

sync.1.1: in(xh, af(init)) => prefix(DEQ(xh), ENQ(xh)) -> true
              which reduces to the equation
              (false <=> in(xh, af(init))) | prefix(DEQ(xh), ENQ(xh)) -> true

Proof of Lemma sync.1.1 suspended.

-> resume by case in(xh,af(init))

Case.1.1
    in(c_xh, af(init)) == true
involves proving Lemma sync.1.1.1
    in(c_xh, af(init)) => prefix(DEQ(c_xh), ENQ(c_xh)) -> true

The case system now contains 1 equation.

Ordered equation Case.1.1 into the rewrite rule:
  in(c_xh, af(init)) -> true

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 164 rewrite rules, and 12 deduction rules.

Ordered equation Case.1.1 into the rewrite rule:
  in(c_xh, af(init)) -> true

The system now contains 165 rewrite rules and 12 deduction rules.

Lemma sync.1.1.1 in the proof by cases of Lemma sync.1.1
    in(c_xh, af(init)) => prefix(DEQ(c_xh), ENQ(c_xh)) -> true
    Case.1.1: in(c_xh, af(init))
is NOT provable using the current partially completed system.  It reduces to
the equation
    prefix(DEQ(c_xh), ENQ(c_xh)) -> true

Proof of Lemma sync.1.1.1 suspended.

-> crit case with Abstraction

Critical pairs between rule Case.1.1:
  in(c_xh, af(init)) -> true
and rule Abstraction.5:
  (in_state(xh, xst) & ordered(xh)) | (false <=> in(xh, af(xst))) -> true
  are as follows:
    in_state(c_xh, init) & ordered(c_xh) == true

The system now contains 1 equation, 165 rewrite rules, and 12 deduction rules.

Deduction rule boolean.3:
```

```
  when  x & y == true
  yield x == true
        y == true
has been applied to equation sync.2:
  in_state(c_xh, init) & ordered(c_xh) == true
to yield the following equations:
  sync.2.1: in_state(c_xh, init) == true
  sync.2.2: ordered(c_xh) == true

Ordered equation sync.2.2 into the rewrite rule:
  ordered(c_xh) -> true

Ordered equation sync.2.1 into the rewrite rule:
  in_state(c_xh, init) -> true

The system now contains 167 rewrite rules and 12 deduction rules.

Computed 2 new critical pairs, 1 of which reduced to an identity.  Added 1 of
them to the system.

-> crit sync with lemma1.12

Critical pairs between rule sync.2.1:
  in_state(c_xh, init) -> true
and rule lemma1.12:
  (false <=> in_state(x, init)) | (null = x) -> true
  are as follows:
    c_xh = null == true

The system now contains 1 equation, 167 rewrite rules, and 12 deduction rules.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation sync.3:
  c_xh = null == true
to yield the following equations:
  sync.3.1: c_xh == null

Ordered equation sync.3.1 into the rewrite rule:
  c_xh -> null

    Following 3 left-hand sides reduced:
    in(c_xh, af(init)) -> true
      became equation Case.1.1:
      in(null, af(init)) == true
    ordered(c_xh) -> true
      became equation sync.2.2:
      ordered(null) == true
    in_state(c_xh, init) -> true
      became equation sync.2.1:
      in_state(null, init) == true

The system now contains 3 equations, 165 rewrite rules, and 12 deduction rules.

Ordered equation Case.1.1 into the rewrite rule:
  in(null, af(init)) -> true

The system now contains 166 rewrite rules and 12 deduction rules.

Lemma sync.1.1.1 in the proof by cases of Lemma sync.1.1
    in(c_xh, af(init)) => prefix(DEQ(c_xh), ENQ(c_xh)) -> true
    Case.1.1: in(c_xh, af(init))
[] Proved by rewriting.

Case.1.2
    not(in(c_xh, af(init))) == true
involves proving Lemma sync.1.1.2
    in(c_xh, af(init)) => prefix(DEQ(c_xh), ENQ(c_xh)) -> true
```

```
The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.1.2:
  false <=> in(c_xh, af(init)) == true
to yield the following equations:
  Case.1.2.1: false == in(c_xh, af(init))

Ordered equation Case.1.2.1 into the rewrite rule:
  in(c_xh, af(init)) -> false

The case system now contains 1 rewrite rule.

Lemma sync.1.1.2 in the proof by cases of Lemma sync.1.1
    in(c_xh, af(init)) => prefix(DEQ(c_xh), ENQ(c_xh)) -> true
    Case.1.2: not(in(c_xh, af(init)))
[] Proved by rewriting (with unreduced rules).

Lemma sync.1.1 for the basis step in the proof of Conjecture sync.1
    in(xh, af(init)) => prefix(DEQ(xh), ENQ(xh)) -> true
[] Proved by cases
    in(xh, af(init)) | not(in(xh, af(init)))

The induction step in an inductive proof of Conjecture sync.1
    in(xh, af(xst)) => prefix(DEQ(xh), ENQ(xh)) -> true
uses the following equation(s) for the induction hypothesis:

Induct.1: in(xh, af(c_xst)) => prefix(DEQ(xh), ENQ(xh)) -> true

The system now contains 1 equation, 164 rewrite rules, and 12 deduction rules.

Ordered equation Induct.1 into the rewrite rule:
  (false <=> in(xh, af(c_xst))) | prefix(DEQ(xh), ENQ(xh)) -> true

The system now contains 165 rewrite rules and 12 deduction rules.

The induction step involves proving the following lemma(s):

sync.1.2: in(xh, af(deq(c_xst, vi1, vi2))) => prefix(DEQ(xh), ENQ(xh)) -> true
              which reduces to the equation
              (false <=> in(xh, af(deq(c_xst, vi1, vi2))))
               | prefix(DEQ(xh), ENQ(xh))
              -> true
sync.1.3: in(xh, af(enq(c_xst, vi1, vi2))) => prefix(DEQ(xh), ENQ(xh)) -> true
              which reduces to the equation
              (false <=> in(xh, af(enq(c_xst, vi1, vi2))))
               | prefix(DEQ(xh), ENQ(xh))
              -> true
sync.1.4: in(xh, af(commit(c_xst, vi1))) => prefix(DEQ(xh), ENQ(xh)) -> true
              which reduces to the equation
              (false <=> in(xh, af(commit(c_xst, vi1))))
               | prefix(DEQ(xh), ENQ(xh))
              -> true
sync.1.5: in(xh, af(abort(c_xst, vi1))) => prefix(DEQ(xh), ENQ(xh)) -> true
              which reduces to the equation
              (false <=> in(xh, af(abort(c_xst, vi1))))
               | prefix(DEQ(xh), ENQ(xh))
              -> true

Proof of Lemma sync.1.5 suspended.

Critical-pair computation abandoned because a theorem has been proved.

Computed 1 new critical pair.  Added 1 of them to the system.

-> resume by case in(xh,af(abort(c_xst,vi1)))
```

```
Case.2.1
    in(c_xh, af(abort(c_xst, c_vil))) == true
involves proving Lemma sync.1.5.1
    in(c_xh, af(abort(c_xst, c_vil))) => prefix(DEQ(c_xh), ENQ(c_xh)) -> true

The case system now contains 1 equation.

Ordered equation Case.2.1 into the rewrite rule:
  in(c_xh, af(abort(c_xst, c_vil))) -> true

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 165 rewrite rules, and 12 deduction rules.

Ordered equation Case.2.1 into the rewrite rule:
  in(c_xh, af(abort(c_xst, c_vil))) -> true

The system now contains 166 rewrite rules and 12 deduction rules.

Lemma sync.1.5.1 in the proof by cases of Lemma sync.1.5
    in(c_xh, af(abort(c_xst, c_vil))) => prefix(DEQ(c_xh), ENQ(c_xh)) -> true
    Case.2.1: in(c_xh, af(abort(c_xst, c_vil)))
is NOT provable using the current partially completed system.  It reduces to
the equation
    prefix(DEQ(c_xh), ENQ(c_xh)) -> true

Proof of Lemma sync.1.5.1 suspended.

-> crit case with Abstraction

Critical pairs between rule Case.2.1:
  in(c_xh, af(abort(c_xst, c_vil))) -> true
and rule Abstraction.5:
  (in_state(xh, xst) & ordered(xh)) | (false <=> in(xh, af(xst))) -> true
  are as follows:
    in_state(c_xh, abort(c_xst, c_vil)) & ordered(c_xh) == true

The system now contains 1 equation, 166 rewrite rules, and 12 deduction rules.

Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation sync.4:
  in_state(c_xh, abort(c_xst, c_vil)) & ordered(c_xh) == true
to yield the following equations:
  sync.4.1: in_state(c_xh, abort(c_xst, c_vil)) == true
  sync.4.2: ordered(c_xh) == true

Ordered equation sync.4.2 into the rewrite rule:
  ordered(c_xh) -> true

Ordered equation sync.4.1 into the rewrite rule:
  in_state(c_xh, abort(c_xst, c_vil)) -> true

The system now contains 168 rewrite rules and 12 deduction rules.

Critical pairs between rule Case.2.1:
  in(c_xh, af(abort(c_xst, c_vil))) -> true
and rule Abstraction.9:
  ((discard(xt, c_hl) = xh) & in(c_hl, af(xst)))
   | (false <=> in(xh, af(abort(xst, xt))))
  -> true
  are as follows:
    (c_xh = discard(c_vil, c_hl)) & in(c_hl, af(c_xst)) == true

The system now contains 1 equation, 168 rewrite rules, and 12 deduction rules.
```

```
Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation sync.5:
  (c_xh = discard(c_vil, c_hl)) & in(c_hl, af(c_xst)) == true
to yield the following equations:
  sync.5.1: c_xh = discard(c_vil, c_hl) == true
  sync.5.2: in(c_hl, af(c_xst)) == true

Ordered equation sync.5.2 into the rewrite rule:
  in(c_hl, af(c_xst)) -> true

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation sync.5.1:
  c_xh = discard(c_vil, c_hl) == true
to yield the following equations:
  sync.5.1.1: c_xh == discard(c_vil, c_hl)

The system now contains 1 equation, 169 rewrite rules, and 12 deduction rules.

Ordered equation sync.5.1.1 into the rewrite rule:
  c_xh -> discard(c_vil, c_hl)

    Following 3 left-hand sides reduced:
    in(c_xh, af(abort(c_xst, c_vil))) -> true
      became equation Case.2.1:
      in(discard(c_vil, c_hl), af(abort(c_xst, c_vil))) == true
    ordered(c_xh) -> true
      became equation sync.4.2:
      ordered(discard(c_vil, c_hl)) == true
    in_state(c_xh, abort(c_xst, c_vil)) -> true
      became equation sync.4.1:
      in_state(discard(c_vil, c_hl), abort(c_xst, c_vil)) == true

The system now contains 3 equations, 167 rewrite rules, and 12 deduction rules.

Ordered equation Case.2.1 into the rewrite rule:
  in(discard(c_vil, c_hl), af(abort(c_xst, c_vil))) -> true

Ordered equation sync.4.2 into the rewrite rule:
  ordered(discard(c_vil, c_hl)) -> true

Ordered equation sync.4.1 into the rewrite rule:
  in_state(discard(c_vil, c_hl), abort(c_xst, c_vil)) -> true

The system now contains 170 rewrite rules and 12 deduction rules.

Lemma sync.1.5.1 in the proof by cases of Lemma sync.1.5
    in(c_xh, af(abort(c_xst, c_vil))) => prefix(DEQ(c_xh), ENQ(c_xh)) -> true
    Case.2.1: in(c_xh, af(abort(c_xst, c_vil)))
is NOT provable using the current partially completed system.  It reduces to
the equation
    prefix(DEQ(discard(c_vil, c_hl)), ENQ(discard(c_vil, c_hl))) -> true

Proof of Lemma sync.1.5.1 suspended.

Critical pairs between rule Case.2.1:
  in(c_xh, af(abort(c_xst, c_vil))) -> true
and rule Abstraction.11:
   (false <=> in(xh, af(xst)))
    | (false <=> in(xn, enqd(xst)))
    | (false <=> least(xn, enqd(xst)))
    | (false <=> prefix(DEQ(xh), ENQ(xh)))
    | prefix(cons(DEQ(xh), element(xn)), ENQ(xh))
   -> true
   are as follows:
```

```
    (false <=> in(xn, enqd(abort(c_xst, c_vil))))
     | (false <=> least(xn, enqd(abort(c_xst, c_vil))))
     | (false <=> prefix(DEQ(discard(c_vil, c_h1)), ENQ(discard(c_vil, c_h1))))
     | prefix(cons(DEQ(discard(c_vil, c_h1)), element(xn)),
              ENQ(discard(c_vil, c_h1)))

    == true

The system now contains 1 equation, 170 rewrite rules, and 12 deduction rules.

Ordered equation sync.6 into the rewrite rule:
  (false <=> in(xn, enqd(abort(c_xst, c_vil))))
   | (false <=> least(xn, enqd(abort(c_xst, c_vil))))
   | (false <=> prefix(DEQ(discard(c_vil, c_h1)), ENQ(discard(c_vil, c_h1))))
   | prefix(cons(DEQ(discard(c_vil, c_h1)), element(xn)),
            ENQ(discard(c_vil, c_h1)))

  -> true

The system now contains 171 rewrite rules and 12 deduction rules.

Computed 3 new critical pairs.  Added 3 of them to the system.

-> crit induct with sync

Critical pairs between rule Induct.1:
  (false <=> in(xh, af(c_xst))) | prefix(DEQ(xh), ENQ(xh)) -> true
and rule sync.5.2:
  in(c_h1, af(c_xst)) -> true
  are as follows:
    prefix(DEQ(c_h1), ENQ(c_h1)) == true

The system now contains 1 equation, 171 rewrite rules, and 12 deduction rules.

Ordered equation sync.7 into the rewrite rule:
  prefix(DEQ(c_h1), ENQ(c_h1)) -> true

The system now contains 172 rewrite rules and 12 deduction rules.

Computed 3 new critical pairs, 2 of which reduced to an identity.  Added 1 of
them to the system.

-> crit sync with lemma1.17

Critical pairs between rule sync.7:
  prefix(DEQ(c_h1), ENQ(c_h1)) -> true
and rule lemma1.17:
  (false <=> in_state(xh, xst))
   | (false <=> prefix(DEQ(xh), ENQ(xh)))
   | prefix(DEQ(discard(xt, xh)), ENQ(discard(xt, xh)))
  -> true
  are as follows:
    (false <=> in_state(c_h1, xst))
     | prefix(DEQ(discard(xt, c_h1)), ENQ(discard(xt, c_h1)))
    == true

The system now contains 1 equation, 172 rewrite rules, and 12 deduction rules.

Ordered equation sync.8 into the rewrite rule:
  (false <=> in_state(c_h1, xst))
   | prefix(DEQ(discard(xt, c_h1)), ENQ(discard(xt, c_h1)))
  -> true

The system now contains 173 rewrite rules and 12 deduction rules.

Critical pairs between rule sync.4.1:
  in_state(discard(c_vil, c_h1), abort(c_xst, c_vil)) -> true
and rule lemma1.17:
  (false <=> in_state(xh, xst))
```

```
    | (false <=> prefix(DEQ(xh), ENQ(xh)))
    | prefix(DEQ(discard(xt, xh)), ENQ(discard(xt, xh)))
  -> true
  are as follows:
    (false <=> prefix(DEQ(discard(c_vil, c_h1)), ENQ(discard(c_vil, c_h1))))
     | prefix(DEQ(discard(xt, discard(c_vil, c_h1))),
              ENQ(discard(xt, discard(c_vil, c_h1))))

    == true

The system now contains 1 equation, 173 rewrite rules, and 12 deduction rules.

Ordered equation sync.9 into the rewrite rule:
  (false <=> prefix(DEQ(discard(c_vil, c_h1)), ENQ(discard(c_vil, c_h1))))
   | prefix(DEQ(discard(xt, discard(c_vil, c_h1))),
            ENQ(discard(xt, discard(c_vil, c_h1))))

  -> true

The system now contains 174 rewrite rules and 12 deduction rules.

Computed 6 new critical pairs, 4 of which reduced to an identity.  Added 2 of
them to the system.

-> crit sync.5.2 with Abstraction.5

Critical pairs between rule sync.5.2:
  in(c_h1, af(c_xst)) -> true
and rule Abstraction.5:
  (in_state(xh, xst) & ordered(xh)) | (false <=> in(xh, af(xst))) -> true
  are as follows:
    in_state(c_h1, c_xst) & ordered(c_h1) == true

The system now contains 1 equation, 174 rewrite rules, and 12 deduction rules.

Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation sync.10:
  in_state(c_h1, c_xst) & ordered(c_h1) == true
to yield the following equations:
  sync.10.1: in_state(c_h1, c_xst) == true
  sync.10.2: ordered(c_h1) == true

Ordered equation sync.10.2 into the rewrite rule:
  ordered(c_h1) -> true

Ordered equation sync.10.1 into the rewrite rule:
  in_state(c_h1, c_xst) -> true

The system now contains 176 rewrite rules and 12 deduction rules.

Computed 1 new critical pair.  Added 1 of them to the system.

-> crit sync with sync

Critical pairs between rule sync.10.1:
  in_state(c_h1, c_xst) -> true
and rule sync.8:
  (false <=> in_state(c_h1, xst))
   | prefix(DEQ(discard(xt, c_h1)), ENQ(discard(xt, c_h1)))
  -> true
  are as follows:
    prefix(DEQ(discard(xt, c_h1)), ENQ(discard(xt, c_h1))) == true

The system now contains 1 equation, 176 rewrite rules, and 12 deduction rules.

Ordered equation sync.11 into the rewrite rule:
```

```
  prefix(DEQ(discard(xt, c_hl)), ENQ(discard(xt, c_hl))) -> true

    Following 3 left-hand sides reduced:
    (false <=> in(xn, enqd(abort(c_xst, c_vil))))
     | (false <=> least(xn, enqd(abort(c_xst, c_vil))))
     | (false <=> prefix(DEQ(discard(c_vil, c_hl)), ENQ(discard(c_vil, c_hl))))
     | prefix(cons(DEQ(discard(c_vil, c_hl)), element(xn)),
              ENQ(discard(c_vil, c_hl)))

    -> true
      became equation sync.6:
      (false <=> in(xn, enqd(abort(c_xst, c_vil))))
       | (false <=> least(xn, enqd(abort(c_xst, c_vil))))
       | (false <=> true)
       | prefix(cons(DEQ(discard(c_vil, c_hl)), element(xn)),
                ENQ(discard(c_vil, c_hl)))

      == true
    (false <=> in_state(c_hl, xst))
     | prefix(DEQ(discard(xt, c_hl)), ENQ(discard(xt, c_hl)))
    -> true
      became equation sync.8:
      (false <=> in_state(c_hl, xst)) | true == true
    (false <=> prefix(DEQ(discard(c_vil, c_hl)), ENQ(discard(c_vil, c_hl))))
     | prefix(DEQ(discard(xt, discard(c_vil, c_hl))),
              ENQ(discard(xt, discard(c_vil, c_hl))))

    -> true
      became equation sync.9:
      (false <=> true)
       | prefix(DEQ(discard(xt, discard(c_vil, c_hl))),
                ENQ(discard(xt, discard(c_vil, c_hl))))

      == true

Ordered equation sync.6 into the rewrite rule:
  (false <=> in(xn, enqd(abort(c_xst, c_vil))))
   | (false <=> least(xn, enqd(abort(c_xst, c_vil))))
   | prefix(cons(DEQ(discard(c_vil, c_hl)), element(xn)),
            ENQ(discard(c_vil, c_hl)))

  -> true

Ordered equation sync.9 into the rewrite rule:
  prefix(DEQ(discard(xt, discard(c_vil, c_hl))),
         ENQ(discard(xt, discard(c_vil, c_hl))))
  -> true

The system now contains 176 rewrite rules and 12 deduction rules.

Lemma sync.1.5.1 in the proof by cases of Lemma sync.1.5
    in(c_xh, af(abort(c_xst, c_vil))) => prefix(DEQ(c_xh), ENQ(c_xh)) -> true
    Case.2.1: in(c_xh, af(abort(c_xst, c_vil)))
[] Proved by rewriting.

Case.2.2
    not(in(c_xh, af(abort(c_xst, c_vil)))) == true
involves proving Lemma sync.1.5.2
    in(c_xh, af(abort(c_xst, c_vil))) => prefix(DEQ(c_xh), ENQ(c_xh)) -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.2.2:
  false <=> in(c_xh, af(abort(c_xst, c_vil))) == true
to yield the following equations:
  Case.2.2.1: false == in(c_xh, af(abort(c_xst, c_vil)))
```

```
Ordered equation Case.2.2.1 into the rewrite rule:
  in(c_xh, af(abort(c_xst, c_vil))) -> false

The case system now contains 1 rewrite rule.

Lemma sync.1.5.2 in the proof by cases of Lemma sync.1.5
    in(c_xh, af(abort(c_xst, c_vil))) => prefix(DEQ(c_xh), ENQ(c_xh)) -> true
    Case.2.2: not(in(c_xh, af(abort(c_xst, c_vil))))
[] Proved by rewriting (with unreduced rules).

Lemma sync.1.5 for the induction step in the proof of Conjecture sync.1
    in(xh, af(abort(c_xst, vil))) => prefix(DEQ(xh), ENQ(xh)) -> true
[] Proved by cases
    in(xh, af(abort(c_xst, vil))) | not(in(xh, af(abort(c_xst, vil))))

Lemma sync.1.4 for the induction step in the proof of Conjecture sync.1
    in(xh, af(commit(c_xst, vil))) => prefix(DEQ(xh), ENQ(xh)) -> true
is NOT provable using the current partially completed system.  It reduces to
the equation
    (false <=> in(xh, af(commit(c_xst, vil)))) | prefix(DEQ(xh), ENQ(xh))
    -> true

Proof of Lemma sync.1.4 suspended.

Critical-pair computation abandoned because a theorem has been proved.

Computed 4 new critical pairs, 3 of which reduced to an identity.  Added 1 of
them to the system.

-> resume by case in(xh,af(commit(c_xst,vil)))

Case.3.1
    in(c_xh, af(commit(c_xst, c_vil))) == true
involves proving Lemma sync.1.4.1
    in(c_xh, af(commit(c_xst, c_vil))) => prefix(DEQ(c_xh), ENQ(c_xh)) -> true

The case system now contains 1 equation.

Ordered equation Case.3.1 into the rewrite rule:
  in(c_xh, af(commit(c_xst, c_vil))) -> true

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 165 rewrite rules, and 12 deduction rules.

Ordered equation Case.3.1 into the rewrite rule:
  in(c_xh, af(commit(c_xst, c_vil))) -> true

The system now contains 166 rewrite rules and 12 deduction rules.

Lemma sync.1.4.1 in the proof by cases of Lemma sync.1.4
    in(c_xh, af(commit(c_xst, c_vil))) => prefix(DEQ(c_xh), ENQ(c_xh)) -> true
    Case.3.1: in(c_xh, af(commit(c_xst, c_vil)))
is NOT provable using the current partially completed system.  It reduces to
the equation
    prefix(DEQ(c_xh), ENQ(c_xh)) -> true

Proof of Lemma sync.1.4.1 suspended.

-> crit case with Abstraction

Critical pairs between rule Case.3.1:
  in(c_xh, af(commit(c_xst, c_vil))) -> true
and rule Abstraction.5:
  (in_state(xh, xst) & ordered(xh)) | (false <=> in(xh, af(xst))) -> true
  are as follows:
    in_state(c_xh, commit(c_xst, c_vil)) & ordered(c_xh) == true
```

```
The system now contains 1 equation, 166 rewrite rules, and 12 deduction rules.

Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation sync.12:
  in_state(c_xh, commit(c_xst, c_vil)) & ordered(c_xh) == true
to yield the following equations:
  sync.12.1: in_state(c_xh, commit(c_xst, c_vil)) == true
  sync.12.2: ordered(c_xh) == true

Ordered equation sync.12.2 into the rewrite rule:
  ordered(c_xh) -> true

Ordered equation sync.12.1 into the rewrite rule:
  in_state(c_xh, commit(c_xst, c_vil)) -> true

The system now contains 168 rewrite rules and 12 deduction rules.

Critical pairs between rule Case.3.1:
  in(c_xh, af(commit(c_xst, c_vil))) -> true
and rule Abstraction.8:
  (false <=> in(xh, af(commit(xst, xt)))) | (DEQ(xh) = null) -> true
  are as follows:
    DEQ(c_xh) = null == true

The system now contains 1 equation, 168 rewrite rules, and 12 deduction rules.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation sync.13:
  DEQ(c_xh) = null == true
to yield the following equations:
  sync.13.1: DEQ(c_xh) == null

Ordered equation sync.13.1 into the rewrite rule:
  DEQ(c_xh) -> null

The system now contains 169 rewrite rules and 12 deduction rules.

Lemma sync.1.4.1 in the proof by cases of Lemma sync.1.4
    in(c_xh, af(commit(c_xst, c_vil))) => prefix(DEQ(c_xh), ENQ(c_xh)) -> true
    Case.3.1: in(c_xh, af(commit(c_xst, c_vil)))
[] Proved by rewriting.

Case.3.2
    not(in(c_xh, af(commit(c_xst, c_vil)))) == true
involves proving Lemma sync.1.4.2
    in(c_xh, af(commit(c_xst, c_vil))) => prefix(DEQ(c_xh), ENQ(c_xh)) -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.3.2:
  false <=> in(c_xh, af(commit(c_xst, c_vil))) == true
to yield the following equations:
  Case.3.2.1: false == in(c_xh, af(commit(c_xst, c_vil)))

Ordered equation Case.3.2.1 into the rewrite rule:
  in(c_xh, af(commit(c_xst, c_vil))) -> false

The case system now contains 1 rewrite rule.

Lemma sync.1.4.2 in the proof by cases of Lemma sync.1.4
    in(c_xh, af(commit(c_xst, c_vil))) => prefix(DEQ(c_xh), ENQ(c_xh)) -> true
```

```
    Case.3.2: not(in(c_xh, af(commit(c_xst, c_vi1))))
[] Proved by rewriting (with unreduced rules).

Lemma sync.1.4 for the induction step in the proof of Conjecture sync.1
    in(xh, af(commit(c_xst, vi1))) => prefix(DEQ(xh), ENQ(xh)) -> true
[] Proved by cases
    in(xh, af(commit(c_xst, vi1))) | not(in(xh, af(commit(c_xst, vi1))))

Lemma sync.1.3 for the induction step in the proof of Conjecture sync.1
    in(xh, af(enq(c_xst, vi1, vi2))) => prefix(DEQ(xh), ENQ(xh)) -> true
is NOT provable using the current partially completed system.  It reduces to
the equation
    (false <=> in(xh, af(enq(c_xst, vi1, vi2)))) | prefix(DEQ(xh), ENQ(xh))
    -> true

Proof of Lemma sync.1.3 suspended.

Critical-pair computation abandoned because a theorem has been proved.

Computed 2 new critical pairs.  Added 2 of them to the system.

-> resume by case in(xh,af(enq(c_xst,vi1,vi2::EL)))

Case.4.1
    in(c_xh, af(enq(c_xst, c_vi1, c_vi2))) == true
involves proving Lemma sync.1.3.1
    in(c_xh, af(enq(c_xst, c_vi1, c_vi2))) => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true

The case system now contains 1 equation.

Ordered equation Case.4.1 into the rewrite rule:
  in(c_xh, af(enq(c_xst, c_vi1, c_vi2))) -> true

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 165 rewrite rules, and 12 deduction rules.

Ordered equation Case.4.1 into the rewrite rule:
  in(c_xh, af(enq(c_xst, c_vi1, c_vi2))) -> true

The system now contains 166 rewrite rules and 12 deduction rules.

Lemma sync.1.3.1 in the proof by cases of Lemma sync.1.3
    in(c_xh, af(enq(c_xst, c_vi1, c_vi2))) => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true
    Case.4.1: in(c_xh, af(enq(c_xst, c_vi1, c_vi2)))
is NOT provable using the current partially completed system.  It reduces to
the equation
    prefix(DEQ(c_xh), ENQ(c_xh)) -> true

Proof of Lemma sync.1.3.1 suspended.

-> add when_enq(c_xst,z,w,c_vi1,c_vi2)

Added 1 equation to the system.

Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation sync.14:
  ((enqr(top(deqd(c_xst))) < c_vi1) | (deqd(c_xst) = new))
   & (((c_vi2 = element(z)) <=> false) | (false <=> in(z, enqd(c_xst))))
   & (((c_vi2 = what(w)) <=> false) | (false <=> in_stack(w, deqd(c_xst))))
  -> true
to yield the following equations:
  sync.14.1: (enqr(top(deqd(c_xst))) < c_vi1) | (deqd(c_xst) = new) == true
  sync.14.2: ((c_vi2 = element(z)) <=> false) | (false <=> in(z, enqd(c_xst)))
```

```
            == true
  sync.14.3: ((c_vi2 = what(w)) <=> false)
              | (false <=> in_stack(w, deqd(c_xst)))
             == true

Ordered equation sync.14.3 into the rewrite rule:
  ((c_vi2 = what(w)) <=> false) | (false <=> in_stack(w, deqd(c_xst))) -> true

Ordered equation sync.14.2 into the rewrite rule:
  ((c_vi2 = element(z)) <=> false) | (false <=> in(z, enqd(c_xst))) -> true

Ordered equation sync.14.1 into the rewrite rule:
  (enqr(top(deqd(c_xst))) < c_vi1) | (deqd(c_xst) = new) -> true

The system now contains 169 rewrite rules and 12 deduction rules.

-> crit case with Abstraction

Critical pairs between rule Case.4.1:
  in(c_xh, af(enq(c_xst, c_vi1, c_vi2))) -> true
and rule Abstraction.5:
  (in_state(xh, xst) & ordered(xh)) | (false <=> in(xh, af(xst))) -> true
  are as follows:
    in_state(c_xh, enq(c_xst, c_vi1, c_vi2)) & ordered(c_xh) == true

The system now contains 1 equation, 169 rewrite rules, and 12 deduction rules.

Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation sync.15:
  in_state(c_xh, enq(c_xst, c_vi1, c_vi2)) & ordered(c_xh) == true
to yield the following equations:
  sync.15.1: in_state(c_xh, enq(c_xst, c_vi1, c_vi2)) == true
  sync.15.2: ordered(c_xh) == true

Ordered equation sync.15.2 into the rewrite rule:
  ordered(c_xh) -> true

Ordered equation sync.15.1 into the rewrite rule:
  in_state(c_xh, enq(c_xst, c_vi1, c_vi2)) -> true

The system now contains 171 rewrite rules and 12 deduction rules.

Critical pairs between rule Case.4.1:
  in(c_xh, af(enq(c_xst, c_vi1, c_vi2))) -> true
and rule Abstraction.6:
  ((append(cons(c_h1, E(pair(xe, xt))), c_h2) = xh)
    & in(append(c_h1, c_h2), af(xst)))
   | (false <=> in(xh, af(enq(xst, xt, xe))))
  -> true
  are as follows:
    (append(cons(c_h1, E(pair(c_vi2, c_vi1))), c_h2) = c_xh)
     & in(append(c_h1, c_h2), af(c_xst))
    == true

The system now contains 1 equation, 171 rewrite rules, and 12 deduction rules.

Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation sync.16:
  (append(cons(c_h1, E(pair(c_vi2, c_vi1))), c_h2) = c_xh)
   & in(append(c_h1, c_h2), af(c_xst))
  == true
to yield the following equations:
  sync.16.1: append(cons(c_h1, E(pair(c_vi2, c_vi1))), c_h2) = c_xh == true
```

```
  sync.16.2: in(append(c_h1, c_h2), af(c_xst)) == true

Ordered equation sync.16.2 into the rewrite rule:
  in(append(c_h1, c_h2), af(c_xst)) -> true

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation sync.16.1:
  append(cons(c_h1, E(pair(c_vi2, c_vi1))), c_h2) = c_xh == true
to yield the following equations:
  sync.16.1.1: append(cons(c_h1, E(pair(c_vi2, c_vi1))), c_h2) == c_xh

The system now contains 1 equation, 172 rewrite rules, and 12 deduction rules.

Ordered equation sync.16.1.1 into the rewrite rule:
  append(cons(c_h1, E(pair(c_vi2, c_vi1))), c_h2) -> c_xh

The system now contains 173 rewrite rules and 12 deduction rules.

Critical pairs between rule Case.4.1:
  in(c_xh, af(enq(c_xst, c_vi1, c_vi2))) -> true
and rule Abstraction.11:
  (false <=> in(xh, af(xst)))
   | (false <=> in(xn, enqd(xst)))
   | (false <=> least(xn, enqd(xst)))
   | (false <=> prefix(DEQ(xh), ENQ(xh)))
   | prefix(cons(DEQ(xh), element(xn)), ENQ(xh))
  -> true
  are as follows:
    (((pair(c_vi2, c_vi1) = xn) <=> false) & (false <=> in(xn, enqd(c_xst))))
     | ((enqt(xn) < c_vi1) <=> false)
     | (false <=> least(xn, enqd(c_xst)))
     | (false <=> prefix(DEQ(c_xh), ENQ(c_xh)))
     | prefix(cons(DEQ(c_xh), element(xn)), ENQ(c_xh))
    == true

The system now contains 1 equation, 173 rewrite rules, and 12 deduction rules.

Ordered equation sync.17 into the rewrite rule:
  (((pair(c_vi2, c_vi1) = xn) <=> false) & (false <=> in(xn, enqd(c_xst))))
   | ((enqt(xn) < c_vi1) <=> false)
   | (false <=> least(xn, enqd(c_xst)))
   | (false <=> prefix(DEQ(c_xh), ENQ(c_xh)))
   | prefix(cons(DEQ(c_xh), element(xn)), ENQ(c_xh))
  -> true

The system now contains 174 rewrite rules and 12 deduction rules.

Computed 3 new critical pairs.  Added 3 of them to the system.


-> resume by case deqd(c_xst)=new

Case.5.1
    deqd(c_xst) = new == true
involves proving Lemma sync.1.3.1.1
    in(c_xh, af(enq(c_xst, c_vi1, c_vi2))) => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true

The case system now contains 1 equation.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.5.1:
  deqd(c_xst) = new == true
to yield the following equations:
  Case.5.1.1: deqd(c_xst) == new
```

```
Ordered equation Case.5.1.1 into the rewrite rule:
  deqd(c_xst) -> new

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 174 rewrite rules, and 12 deduction rules.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.5.1:
  deqd(c_xst) = new == true
to yield the following equations:
  Case.5.1.2: deqd(c_xst) == new

Ordered equation Case.5.1.2 into the rewrite rule:
  deqd(c_xst) -> new

    Following 2 left-hand sides reduced:
    ((c_vi2 = what(w)) <=> false) | (false <=> in_stack(w, deqd(c_xst)))
    -> true
      became equation sync.14.3:
      ((c_vi2 = what(w)) <=> false) | (false <=> in_stack(w, new)) == true
    (enqr(top(deqd(c_xst))) < c_vi1) | (deqd(c_xst) = new) -> true
      became equation sync.14.1:
      (enqr(top(new)) < c_vi1) | (deqd(c_xst) = new) == true

The system now contains 173 rewrite rules and 12 deduction rules.

Lemma sync.1.3.1.1 in the proof by cases of Lemma sync.1.3.1
    in(c_xh, af(enq(c_xst, c_vi1, c_vi2))) => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true
    Case.5.1: deqd(c_xst) = new
is NOT provable using the current partially completed system.  It reduces to
the equation
    prefix(DEQ(c_xh), ENQ(c_xh)) -> true

Proof of Lemma sync.1.3.1.1 suspended.

-> crit case with lemma2.1

Critical pairs between rule Case.5.1.2:
  deqd(c_xst) -> new
and rule lemma2.1:
  ((deqd(xst) = new) <=> false)
   | (false <=> in_state(xh, xst))
   | (DEQ(xh) = null)
  -> true
  are as follows:
    (false <=> in_state(xh, c_xst)) | (DEQ(xh) = null) == true

The system now contains 1 equation, 173 rewrite rules, and 12 deduction rules.

Ordered equation sync.18 into the rewrite rule:
  (false <=> in_state(xh, c_xst)) | (DEQ(xh) = null) -> true

The system now contains 174 rewrite rules and 12 deduction rules.

Computed 1 new critical pair.  Added 1 of them to the system.

-> crit sync with lemma1.8

Critical pairs between rule sync.16.1.1:
  append(cons(c_h1, E(pair(c_vi2, c_vi1))), c_h2) -> c_xh
and rule lemma1.8:
  DEQ(append(cons(x, E(y)), z)) -> DEQ(append(x, z))
  are as follows:
    DEQ(c_xh) == DEQ(append(c_h1, c_h2))
```

```
The system now contains 1 equation, 174 rewrite rules, and 12 deduction rules.

Ordered equation sync.19 into the rewrite rule:
  DEQ(append(c_h1, c_h2)) -> DEQ(c_xh)

The system now contains 175 rewrite rules and 12 deduction rules.

Critical pairs between rule sync.18:
  (false <=> in_state(xh, c_xst)) | (DEQ(xh) = null) -> true
and rule lemmal.8:
  DEQ(append(cons(x, E(y)), z)) -> DEQ(append(x, z))
  are as follows:
    (false <=> in_state(append(cons(x, E(y)), z), c_xst))
     | (DEQ(append(x, z)) = null)
    == true

The system now contains 1 equation, 175 rewrite rules, and 12 deduction rules.

Ordered equation sync.20 into the rewrite rule:
  (false <=> in_state(append(cons(x, E(y)), z), c_xst))
   | (DEQ(append(x, z)) = null)
  -> true

The system now contains 176 rewrite rules and 12 deduction rules.

Computed 2 new critical pairs.  Added 2 of them to the system.

-> crit sync.16.2 with Abstraction.5

Critical pairs between rule sync.16.2:
  in(append(c_h1, c_h2), af(c_xst)) -> true
and rule Abstraction.5:
  (in_state(xh, xst) & ordered(xh)) | (false <=> in(xh, af(xst))) -> true
  are as follows:
    in_state(append(c_h1, c_h2), c_xst) & ordered(append(c_h1, c_h2)) == true

The system now contains 1 equation, 176 rewrite rules, and 12 deduction rules.

Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation sync.21:
  in_state(append(c_h1, c_h2), c_xst) & ordered(append(c_h1, c_h2)) == true
to yield the following equations:
  sync.21.1: in_state(append(c_h1, c_h2), c_xst) == true
  sync.21.2: ordered(append(c_h1, c_h2)) == true

Ordered equation sync.21.2 into the rewrite rule:
  ordered(append(c_h1, c_h2)) -> true

Ordered equation sync.21.1 into the rewrite rule:
  in_state(append(c_h1, c_h2), c_xst) -> true

The system now contains 178 rewrite rules and 12 deduction rules.

Computed 1 new critical pair.  Added 1 of them to the system.

-> crit sync with sync

Critical pairs between rule sync.19:
  DEQ(append(c_h1, c_h2)) -> DEQ(c_xh)
and rule sync.18:
  (false <=> in_state(xh, c_xst)) | (DEQ(xh) = null) -> true
  are as follows:
    DEQ(c_xh) = null == true

The system now contains 1 equation, 178 rewrite rules, and 12 deduction rules.
```

```
Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation sync.22:
  DEQ(c_xh) = null == true
to yield the following equations:
  sync.22.1: DEQ(c_xh) == null

Ordered equation sync.22.1 into the rewrite rule:
  DEQ(c_xh) -> null

    Left-hand side reduced:
    (((pair(c_vi2, c_vi1) = xn) <=> false) & (false <=> in(xn, enqd(c_xst))))
     | ((enqt(xn) < c_vi1) <=> false)
     | (false <=> least(xn, enqd(c_xst)))
     | (false <=> prefix(DEQ(c_xh), ENQ(c_xh)))
     | prefix(cons(DEQ(c_xh), element(xn)), ENQ(c_xh))
    -> true
      became equation sync.17:
      (((pair(c_vi2, c_vi1) = xn) <=> false) & (false <=> in(xn, enqd(c_xst))))
       | ((enqt(xn) < c_vi1) <=> false)
       | (false <=> least(xn, enqd(c_xst)))
       | (false <=> prefix(null, ENQ(c_xh)))
       | prefix(cons(DEQ(c_xh), element(xn)), ENQ(c_xh))
      == true

Ordered equation sync.17 into the rewrite rule:
  (((pair(c_vi2, c_vi1) = xn) <=> false) & (false <=> in(xn, enqd(c_xst))))
   | ((enqt(xn) < c_vi1) <=> false)
   | (false <=> least(xn, enqd(c_xst)))
   | prefix(cons(null, element(xn)), ENQ(c_xh))
  -> true

The system now contains 179 rewrite rules and 12 deduction rules.

Lemma sync.1.3.1.1 in the proof by cases of Lemma sync.1.3.1
    in(c_xh, af(enq(c_xst, c_vi1, c_vi2))) => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true
    Case.5.1: deqd(c_xst) = new
[] Proved by rewriting.

Case.5.2
    not(deqd(c_xst) = new) == true
involves proving Lemma sync.1.3.1.2
    in(c_xh, af(enq(c_xst, c_vi1, c_vi2))) => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.5.2:
  (deqd(c_xst) = new) <=> false == true
to yield the following equations:
  Case.5.2.1: deqd(c_xst) = new == false

Ordered equation Case.5.2.1 into the rewrite rule:
  deqd(c_xst) = new -> false

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 174 rewrite rules, and 12 deduction rules.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.5.2:
```

```
  (deqd(c_xst) = new) <=> false == true
to yield the following equations:
  Case.5.2.2: deqd(c_xst) = new == false

Ordered equation Case.5.2.2 into the rewrite rule:
  deqd(c_xst) = new -> false

    Left-hand side reduced:
    (enqr(top(deqd(c_xst))) < c_vil) | (deqd(c_xst) = new) -> true
      became equation sync.14.1:
      (enqr(top(deqd(c_xst))) < c_vil) | false == true

Ordered equation sync.14.1 into the rewrite rule:
  enqr(top(deqd(c_xst))) < c_vil -> true

The system now contains 175 rewrite rules and 12 deduction rules.

Lemma sync.1.3.1.2 in the proof by cases of Lemma sync.1.3.1
    in(c_xh, af(enq(c_xst, c_vil, c_vi2))) => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true
    Case.5.2: not(deqd(c_xst) = new)
is NOT provable using the current partially completed system.  It reduces to
the equation
    prefix(DEQ(c_xh), ENQ(c_xh)) -> true

Proof of Lemma sync.1.3.1.2 suspended.

Critical-pair computation abandoned because a theorem has been proved.

Computed 3 new critical pairs, 2 of which reduced to an identity.  Added 1 of
them to the system.

-> crit induct with sync.16.2

Critical pairs between rule Induct.1:
  (false <=> in(xh, af(c_xst))) | prefix(DEQ(xh), ENQ(xh)) -> true
and rule sync.16.2:
  in(append(c_h1, c_h2), af(c_xst)) -> true
  are as follows:
    prefix(DEQ(append(c_h1, c_h2)), ENQ(append(c_h1, c_h2))) == true

The system now contains 1 equation, 175 rewrite rules, and 12 deduction rules.

Ordered equation sync.23 into the rewrite rule:
  prefix(DEQ(append(c_h1, c_h2)), ENQ(append(c_h1, c_h2))) -> true

The system now contains 176 rewrite rules and 12 deduction rules.

Computed 1 new critical pair.  Added 1 of them to the system.

-> instantiate xh1 by c_h1,xh2 by c_h2,xh by c_xh,xe by c_vi2,xt by c_vil,xst by c_xst in lemma2.3


Equation lemma2.3:
  ((enqr(top(deqd(xst))) < xt) <=> false)
   | ((append(cons(xh1, E(pair(xe, xt))), xh2) = xh) <=> false)
   | (false <=> in(append(xh1, xh2), af(xst)))
   | (false <=> ordered(xh))
   | (false <=> prefix(DEQ(append(xh1, xh2)), ENQ(append(xh1, xh2))))
   | prefix(DEQ(xh), ENQ(xh))
  -> true
has been instantiated to equation lemma2.3.1:
  prefix(DEQ(c_xh), ENQ(c_xh)) -> true

Added 1 equation to the system.

Ordered equation lemma2.3.1 into the rewrite rule:
  prefix(DEQ(c_xh), ENQ(c_xh)) -> true
```

```
    Left-hand side reduced:
    (((pair(c_vi2, c_vi1) = xn) <=> false) & (false <=> in(xn, enqd(c_xst))))
     | ((enqt(xn) < c_vi1) <=> false)
     | (false <=> least(xn, enqd(c_xst)))
     | (false <=> prefix(DEQ(c_xh), ENQ(c_xh)))
     | prefix(cons(DEQ(c_xh), element(xn)), ENQ(c_xh))
    -> true
      became equation sync.17:
      (((pair(c_vi2, c_vi1) = xn) <=> false) & (false <=> in(xn, enqd(c_xst))))
       | ((enqt(xn) < c_vi1) <=> false)
       | (false <=> least(xn, enqd(c_xst)))
       | (false <=> true)
       | prefix(cons(DEQ(c_xh), element(xn)), ENQ(c_xh))
      == true

Ordered equation sync.17 into the rewrite rule:
  (((pair(c_vi2, c_vi1) = xn) <=> false) & (false <=> in(xn, enqd(c_xst))))
   | ((enqt(xn) < c_vi1) <=> false)
   | (false <=> least(xn, enqd(c_xst)))
   | prefix(cons(DEQ(c_xh), element(xn)), ENQ(c_xh))
  -> true

The system now contains 177 rewrite rules and 12 deduction rules.

Lemma sync.1.3.1.2 in the proof by cases of Lemma sync.1.3.1
    in(c_xh, af(enq(c_xst, c_vi1, c_vi2))) => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true
    Case.5.2: not(deqd(c_xst) = new)
[] Proved by rewriting.

Lemma sync.1.3.1 in the proof by cases of Lemma sync.1.3
    in(c_xh, af(enq(c_xst, c_vi1, c_vi2))) => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true
    Case.4.1: in(c_xh, af(enq(c_xst, c_vi1, c_vi2)))
[] Proved by cases
     (deqd(c_xst) = new) | not(deqd(c_xst) = new)

Case.4.2
    not(in(c_xh, af(enq(c_xst, c_vi1, c_vi2)))) == true
involves proving Lemma sync.1.3.2
    in(c_xh, af(enq(c_xst, c_vi1, c_vi2))) => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.4.2:
  false <=> in(c_xh, af(enq(c_xst, c_vi1, c_vi2))) == true
to yield the following equations:
  Case.4.2.1: false == in(c_xh, af(enq(c_xst, c_vi1, c_vi2)))

Ordered equation Case.4.2.1 into the rewrite rule:
  in(c_xh, af(enq(c_xst, c_vi1, c_vi2))) -> false

The case system now contains 1 rewrite rule.

Lemma sync.1.3.2 in the proof by cases of Lemma sync.1.3
    in(c_xh, af(enq(c_xst, c_vi1, c_vi2))) => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true
    Case.4.2: not(in(c_xh, af(enq(c_xst, c_vi1, c_vi2))))
[] Proved by rewriting (with unreduced rules).

Lemma sync.1.3 for the induction step in the proof of Conjecture sync.1
    in(xh, af(enq(c_xst, vi1, vi2))) => prefix(DEQ(xh), ENQ(xh)) -> true
[] Proved by cases
    in(xh, af(enq(c_xst, vi1, vi2))) | not(in(xh, af(enq(c_xst, vi1, vi2))))
```

```
Lemma sync.1.2 for the induction step in the proof of Conjecture sync.1
    in(xh, af(deq(c_xst, vi1, vi2))) => prefix(DEQ(xh), ENQ(xh)) -> true
is NOT provable using the current partially completed system.  It reduces to
the equation
    (false <=> in(xh, af(deq(c_xst, vi1, vi2)))) | prefix(DEQ(xh), ENQ(xh))
    -> true

Proof of Lemma sync.1.2 suspended.

-> resume by case in(xh,af(deq(c_xst,vi1,vi2::enq_rec)))

Case.6.1
    in(c_xh, af(deq(c_xst, c_vi1, c_vi2))) == true
involves proving Lemma sync.1.2.1
    in(c_xh, af(deq(c_xst, c_vi1, c_vi2))) => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true

The case system now contains 1 equation.

Ordered equation Case.6.1 into the rewrite rule:
  in(c_xh, af(deq(c_xst, c_vi1, c_vi2))) -> true

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 165 rewrite rules, and 12 deduction rules.

Ordered equation Case.6.1 into the rewrite rule:
  in(c_xh, af(deq(c_xst, c_vi1, c_vi2))) -> true

The system now contains 166 rewrite rules and 12 deduction rules.

Lemma sync.1.2.1 in the proof by cases of Lemma sync.1.2
    in(c_xh, af(deq(c_xst, c_vi1, c_vi2))) => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true
    Case.6.1: in(c_xh, af(deq(c_xst, c_vi1, c_vi2)))
is NOT provable using the current partially completed system.  It reduces to
the equation
    prefix(DEQ(c_xh), ENQ(c_xh)) -> true

Proof of Lemma sync.1.2.1 suspended.

-> crit case with Abstraction

Critical pairs between rule Case.6.1:
  in(c_xh, af(deq(c_xst, c_vi1, c_vi2))) -> true
and rule Abstraction.5:
  (in_state(xh, xst) & ordered(xh)) | (false <=> in(xh, af(xst))) -> true
  are as follows:
    in_state(c_xh, deq(c_xst, c_vi1, c_vi2)) & ordered(c_xh) == true

The system now contains 1 equation, 166 rewrite rules, and 12 deduction rules.

Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation sync.24:
  in_state(c_xh, deq(c_xst, c_vi1, c_vi2)) & ordered(c_xh) == true
to yield the following equations:
  sync.24.1: in_state(c_xh, deq(c_xst, c_vi1, c_vi2)) == true
  sync.24.2: ordered(c_xh) == true

Ordered equation sync.24.2 into the rewrite rule:
  ordered(c_xh) -> true

Ordered equation sync.24.1 into the rewrite rule:
  in_state(c_xh, deq(c_xst, c_vi1, c_vi2)) -> true

The system now contains 168 rewrite rules and 12 deduction rules.
```

```
Critical pairs between rule Case.6.1:
  in(c_xh, af(deq(c_xst, c_vi1, c_vi2))) -> true
and rule Abstraction.7:
  ((DEQ(c_h2) = null)
    & (append(cons(c_h1, D(trip(element(xn), enqt(xn), xt))), c_h2) = xh)
    & in(append(c_h1, c_h2), af(xst)))
   | (false <=> in(xh, af(deq(xst, xt, xn))))
  -> true
  are as follows:
    (DEQ(c_h2) = null)
     & (append(cons(c_h1, D(trip(element(c_vi2), enqt(c_vi2), c_vi1))), c_h2)
         = c_xh)

     & in(append(c_h1, c_h2), af(c_xst))
    == true

The system now contains 1 equation, 168 rewrite rules, and 12 deduction rules.

Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation sync.25:
  (DEQ(c_h2) = null)
   & (append(cons(c_h1, D(trip(element(c_vi2), enqt(c_vi2), c_vi1))), c_h2)
       = c_xh)

   & in(append(c_h1, c_h2), af(c_xst))
  == true
to yield the following equations:
  sync.25.1: DEQ(c_h2) = null == true
  sync.25.2: append(cons(c_h1, D(trip(element(c_vi2), enqt(c_vi2), c_vi1))),
                    c_h2)
              = c_xh
             == true
  sync.25.3: in(append(c_h1, c_h2), af(c_xst)) == true

Ordered equation sync.25.3 into the rewrite rule:
  in(append(c_h1, c_h2), af(c_xst)) -> true

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation sync.25.2:
  append(cons(c_h1, D(trip(element(c_vi2), enqt(c_vi2), c_vi1))), c_h2) = c_xh
  == true
to yield the following equations:
  sync.25.2.1: append(cons(c_h1, D(trip(element(c_vi2), enqt(c_vi2), c_vi1))),
                      c_h2)
               == c_xh

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation sync.25.1:
  DEQ(c_h2) = null == true
to yield the following equations:
  sync.25.1.1: DEQ(c_h2) == null

The system now contains 2 equations, 169 rewrite rules, and 12 deduction rules.

Ordered equation sync.25.1.1 into the rewrite rule:
  DEQ(c_h2) -> null

    Left-hand side reduced:
    ((DEQ(c_h2) = null)
      & (append(cons(c_h1, D(trip(element(xn), enqt(xn), xt))), c_h2) = xh)
      & in(append(c_h1, c_h2), af(xst)))
```

```
     | (false <=> in(xh, af(deq(xst, xt, xn))))
    -> true
      became equation Abstraction.7:
      ((append(cons(c_h1, D(trip(element(xn), enqt(xn), xt))), c_h2) = xh)
        & (null = null)
        & in(append(c_h1, c_h2), af(xst)))
       | (false <=> in(xh, af(deq(xst, xt, xn))))
      -> true

Ordered equation Abstraction.7 into the rewrite rule:
  ((append(cons(c_h1, D(trip(element(xn), enqt(xn), xt))), c_h2) = xh)
    & in(append(c_h1, c_h2), af(xst)))
   | (false <=> in(xh, af(deq(xst, xt, xn))))
  -> true

The system now contains 1 equation, 170 rewrite rules, and 12 deduction rules.

Ordered equation sync.25.2.1 into the rewrite rule:
  append(cons(c_h1, D(trip(element(c_vi2), enqt(c_vi2), c_vi1))), c_h2) -> c_xh

The system now contains 171 rewrite rules and 12 deduction rules.

Critical pairs between rule Case.6.1:
  in(c_xh, af(deq(c_xst, c_vi1, c_vi2))) -> true
and rule Abstraction.11:
  (false <=> in(xh, af(xst)))
   | (false <=> in(xn, enqd(xst)))
   | (false <=> least(xn, enqd(xst)))
   | (false <=> prefix(DEQ(xh), ENQ(xh)))
   | prefix(cons(DEQ(xh), element(xn)), ENQ(xh))
  -> true
  are as follows:
    (false <=> in(xn, enqd(c_xst)))
     | (false <=> least(xn, delete(enqd(c_xst), c_vi2)))
     | (false <=> prefix(DEQ(c_xh), ENQ(c_xh)))
     | (c_vi2 = xn)
     | prefix(cons(DEQ(c_xh), element(xn)), ENQ(c_xh))
    == true

The system now contains 1 equation, 171 rewrite rules, and 12 deduction rules.

Ordered equation sync.26 into the rewrite rule:
  (false <=> in(xn, enqd(c_xst)))
   | (false <=> least(xn, delete(enqd(c_xst), c_vi2)))
   | (false <=> prefix(DEQ(c_xh), ENQ(c_xh)))
   | (c_vi2 = xn)
   | prefix(cons(DEQ(c_xh), element(xn)), ENQ(c_xh))
  -> true

The system now contains 172 rewrite rules and 12 deduction rules.

Computed 3 new critical pairs.  Added 3 of them to the system.

-> add when_deq(c_xst,x,c_vi1,c_vi2)

Added 1 equation to the system.

Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation sync.27:
  (enqt(c_vi2) < c_vi1)
   & in(c_vi2, enqd(c_xst))
   & least(c_vi2, enqd(c_xst))
   & (((deqr(top(deqd(c_xst))) < c_vi1)
        & (enqr(top(deqd(c_xst))) < enqt(c_vi2)))
       | (deqd(c_xst) = new))
```

```
  & (((element(c_vi2) = what(x)) <=> false)
      | (false <=> in_stack(x, deqd(c_xst))))

  -> true
to yield the following equations:
  sync.27.1: enqt(c_vi2) < c_vi1 == true
  sync.27.2: in(c_vi2, enqd(c_xst)) == true
  sync.27.3: least(c_vi2, enqd(c_xst)) == true
  sync.27.4: ((deqr(top(deqd(c_xst))) < c_vi1)
               & (enqr(top(deqd(c_xst))) < enqt(c_vi2)))
              | (deqd(c_xst) = new)
             == true
  sync.27.5: ((element(c_vi2) = what(x)) <=> false)
              | (false <=> in_stack(x, deqd(c_xst)))
             == true

Ordered equation sync.27.5 into the rewrite rule:
  ((element(c_vi2) = what(x)) <=> false) | (false <=> in_stack(x, deqd(c_xst)))
  -> true

Ordered equation sync.27.4 into the rewrite rule:
  ((deqr(top(deqd(c_xst))) < c_vi1) & (enqr(top(deqd(c_xst))) < enqt(c_vi2)))
   | (deqd(c_xst) = new)
  -> true

Ordered equation sync.27.3 into the rewrite rule:
  least(c_vi2, enqd(c_xst)) -> true

Ordered equation sync.27.2 into the rewrite rule:
  in(c_vi2, enqd(c_xst)) -> true

Ordered equation sync.27.1 into the rewrite rule:
  enqt(c_vi2) < c_vi1 -> true

The system now contains 177 rewrite rules and 12 deduction rules.

-> crit induct with sync

Critical pairs between rule Induct.1:
  (false <=> in(xh, af(c_xst))) | prefix(DEQ(xh), ENQ(xh)) -> true
and rule sync.25.3:
  in(append(c_h1, c_h2), af(c_xst)) -> true
  are as follows:
    prefix(DEQ(append(c_h1, c_h2)), ENQ(append(c_h1, c_h2))) == true

The system now contains 1 equation, 177 rewrite rules, and 12 deduction rules.

Ordered equation sync.28 into the rewrite rule:
  prefix(DEQ(append(c_h1, c_h2)), ENQ(append(c_h1, c_h2))) -> true

The system now contains 178 rewrite rules and 12 deduction rules.

Computed 8 new critical pairs, 7 of which reduced to an identity.  Added 1 of
them to the system.

-> resume by case deqd(c_xst)=new

Case.7.1
    deqd(c_xst) = new == true
involves proving Lemma sync.1.2.1.1
    in(c_xh, af(deq(c_xst, c_vi1, c_vi2))) => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true

The case system now contains 1 equation.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.7.1:
```

```
  deqd(c_xst) = new == true
to yield the following equations:
  Case.7.1.1: deqd(c_xst) == new

Ordered equation Case.7.1.1 into the rewrite rule:
  deqd(c_xst) -> new

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 178 rewrite rules, and 12 deduction rules.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation Case.7.1:
  deqd(c_xst) = new == true
to yield the following equations:
  Case.7.1.2: deqd(c_xst) == new

Ordered equation Case.7.1.2 into the rewrite rule:
  deqd(c_xst) -> new

    Following 2 left-hand sides reduced:
    ((element(c_vi2) = what(x)) <=> false)
     | (false <=> in_stack(x, deqd(c_xst)))
    -> true
      became equation sync.27.5:
      ((element(c_vi2) = what(x)) <=> false) | (false <=> in_stack(x, new))
      == true
    ((deqr(top(deqd(c_xst))) < c_vi1) & (enqr(top(deqd(c_xst))) < enqt(c_vi2)))
     | (deqd(c_xst) = new)
    -> true
      became equation sync.27.4:
      ((deqr(top(new)) < c_vi1) & (enqr(top(deqd(c_xst))) < enqt(c_vi2)))
       | (deqd(c_xst) = new)
      == true

The system now contains 177 rewrite rules and 12 deduction rules.

Lemma sync.1.2.1.1 in the proof by cases of Lemma sync.1.2.1
    in(c_xh, af(deq(c_xst, c_vi1, c_vi2))) => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true
    Case.7.1: deqd(c_xst) = new
is NOT provable using the current partially completed system.  It reduces to
the equation
    prefix(DEQ(c_xh), ENQ(c_xh)) -> true

Proof of Lemma sync.1.2.1.1 suspended.

-> crit case with lemma2.1

Critical pairs between rule Case.7.1.2:
  deqd(c_xst) -> new
and rule lemma2.1:
  ((deqd(xst) = new) <=> false)
   | (false <=> in_state(xh, xst))
   | (DEQ(xh) = null)
  -> true
  are as follows:
    (false <=> in_state(xh, c_xst)) | (DEQ(xh) = null) == true

The system now contains 1 equation, 177 rewrite rules, and 12 deduction rules.

Ordered equation sync.29 into the rewrite rule:
  (false <=> in_state(xh, c_xst)) | (DEQ(xh) = null) -> true

The system now contains 178 rewrite rules and 12 deduction rules.

Computed 1 new critical pair.  Added 1 of them to the system.
```

```
-> crit sync.25.3 with Abstraction.5

Critical pairs between rule sync.25.3:
  in(append(c_h1, c_h2), af(c_xst)) -> true
and rule Abstraction.5:
  (in_state(xh, xst) & ordered(xh)) | (false <=> in(xh, af(xst))) -> true
  are as follows:
    in_state(append(c_h1, c_h2), c_xst) & ordered(append(c_h1, c_h2)) == true

The system now contains 1 equation, 178 rewrite rules, and 12 deduction rules.

Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation sync.30:
  in_state(append(c_h1, c_h2), c_xst) & ordered(append(c_h1, c_h2)) == true
to yield the following equations:
  sync.30.1: in_state(append(c_h1, c_h2), c_xst) == true
  sync.30.2: ordered(append(c_h1, c_h2)) == true

Ordered equation sync.30.2 into the rewrite rule:
  ordered(append(c_h1, c_h2)) -> true

Ordered equation sync.30.1 into the rewrite rule:
  in_state(append(c_h1, c_h2), c_xst) -> true

The system now contains 180 rewrite rules and 12 deduction rules.

Computed 1 new critical pair.  Added 1 of them to the system.

-> crit sync.30.1 with sync.29

Critical pairs between rule sync.30.1:
  in_state(append(c_h1, c_h2), c_xst) -> true
and rule sync.29:
  (false <=> in_state(xh, c_xst)) | (DEQ(xh) = null) -> true
  are as follows:
    DEQ(append(c_h1, c_h2)) = null == true

The system now contains 1 equation, 180 rewrite rules, and 12 deduction rules.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation sync.31:
  DEQ(append(c_h1, c_h2)) = null == true
to yield the following equations:
  sync.31.1: DEQ(append(c_h1, c_h2)) == null

Ordered equation sync.31.1 into the rewrite rule:
  DEQ(append(c_h1, c_h2)) -> null

    Left-hand side reduced:
    prefix(DEQ(append(c_h1, c_h2)), ENQ(append(c_h1, c_h2))) -> true
      became equation sync.28:
      prefix(null, ENQ(append(c_h1, c_h2))) == true

The system now contains 180 rewrite rules and 12 deduction rules.

Computed 1 new critical pair.  Added 1 of them to the system.

-> crit sync.31.1 with lemma3.1

Critical pairs between rule sync.31.1:
  DEQ(append(c_h1, c_h2)) -> null
and rule lemma3.1:
  ((DEQ(xh) = null) & (DEQ(xh1) = null))
```

```
  | ((DEQ(append(xh, xh1)) = null) <=> false)
  -> true
  are as follows:
    ((DEQ(append(c_h1, append(c_h2, xh1))) = null) <=> false)
     | (DEQ(xh1) = null)
    == true
    ((DEQ(append(xh, append(c_h1, c_h2))) = null) <=> false) | (DEQ(xh) = null)
    == true
    DEQ(c_h1) = null == true

The system now contains 1 equation, 180 rewrite rules, and 12 deduction rules.

Ordered equation sync.32 into the rewrite rule:
  ((DEQ(append(c_h1, append(c_h2, xh1))) = null) <=> false) | (DEQ(xh1) = null)
  -> true

The system now contains 181 rewrite rules and 12 deduction rules.

The system now contains 1 equation, 181 rewrite rules, and 12 deduction rules.

Ordered equation sync.33 into the rewrite rule:
  ((DEQ(append(xh, append(c_h1, c_h2))) = null) <=> false) | (DEQ(xh) = null)
  -> true

The system now contains 182 rewrite rules and 12 deduction rules.

The system now contains 1 equation, 182 rewrite rules, and 12 deduction rules.

Deduction rule equality.4:
  when  x = y == true
  yield x == y
has been applied to equation sync.34:
  DEQ(c_h1) = null == true
to yield the following equations:
  sync.34.1: DEQ(c_h1) == null

Ordered equation sync.34.1 into the rewrite rule:
  DEQ(c_h1) -> null

The system now contains 183 rewrite rules and 12 deduction rules.

Computed 3 new critical pairs.  Added 3 of them to the system.


-> instantiate xh1 by c_h1,xh2 by c_h2,xn by c_vi2,xt by c_vi1,xh by c_xh,xst by c_xst in lemma3.2


Equation lemma3.2:
  ((DEQ(xh1) = null) <=> false)
   | ((DEQ(xh2) = null) <=> false)
   | ((append(cons(xh1, D(trip(element(xn), enqt(xn), xt))), xh2) = xh)
      <=> false)

   | (false <=> in(append(xh1, xh2), af(xst)))
   | (false <=> in(xn, enqd(xst)))
   | (false <=> least(xn, enqd(xst)))
   | prefix(DEQ(xh), ENQ(xh))
  -> true
has been instantiated to equation lemma3.2.1:
  prefix(DEQ(c_xh), ENQ(c_xh)) -> true

Added 1 equation to the system.

Ordered equation lemma3.2.1 into the rewrite rule:
  prefix(DEQ(c_xh), ENQ(c_xh)) -> true

    Left-hand side reduced:
    (false <=> in(xn, enqd(c_xst)))
     | (false <=> least(xn, delete(enqd(c_xst), c_vi2)))
```

```
      | (false <=> prefix(DEQ(c_xh), ENQ(c_xh)))
      | (c_vi2 = xn)
      | prefix(cons(DEQ(c_xh), element(xn)), ENQ(c_xh))
    -> true
       became equation sync.26:
       (false <=> in(xn, enqd(c_xst)))
        | (false <=> least(xn, delete(enqd(c_xst), c_vi2)))
        | (false <=> true)
        | (c_vi2 = xn)
        | prefix(cons(DEQ(c_xh), element(xn)), ENQ(c_xh))
       == true

Ordered equation sync.26 into the rewrite rule:
  (false <=> in(xn, enqd(c_xst)))
   | (false <=> least(xn, delete(enqd(c_xst), c_vi2)))
   | (c_vi2 = xn)
   | prefix(cons(DEQ(c_xh), element(xn)), ENQ(c_xh))
  -> true

The system now contains 184 rewrite rules and 12 deduction rules.

Lemma sync.1.2.1.1 in the proof by cases of Lemma sync.1.2.1
    in(c_xh, af(deq(c_xst, c_vi1, c_vi2))) => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true
    Case.7.1: deqd(c_xst) = new
[] Proved by rewriting.

Case.7.2
    not(deqd(c_xst) = new) == true
involves proving Lemma sync.1.2.1.2
    in(c_xh, af(deq(c_xst, c_vi1, c_vi2))) => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.7.2:
  (deqd(c_xst) = new) <=> false == true
to yield the following equations:
  Case.7.2.1: deqd(c_xst) = new == false

Ordered equation Case.7.2.1 into the rewrite rule:
  deqd(c_xst) = new -> false

The case system now contains 1 rewrite rule.

The system now contains 1 equation, 178 rewrite rules, and 12 deduction rules.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.7.2:
  (deqd(c_xst) = new) <=> false == true
to yield the following equations:
  Case.7.2.2: deqd(c_xst) = new == false

Ordered equation Case.7.2.2 into the rewrite rule:
  deqd(c_xst) = new -> false

    Left-hand side reduced:
    ((deqr(top(deqd(c_xst))) < c_vi1) & (enqr(top(deqd(c_xst))) < enqt(c_vi2)))
     | (deqd(c_xst) = new)
    -> true
      became equation sync.27.4:
      ((deqr(top(deqd(c_xst))) < c_vi1)
        & (enqr(top(deqd(c_xst))) < enqt(c_vi2)))
       | false
```

```
      == true

Deduction rule boolean.3:
  when  x & y == true
  yield x == true
        y == true
has been applied to equation sync.27.4:
  (deqr(top(deqd(c_xst))) < c_vi1) & (enqr(top(deqd(c_xst))) < enqt(c_vi2))
  == true
to yield the following equations:
  sync.27.4.1: deqr(top(deqd(c_xst))) < c_vi1 == true
  sync.27.4.2: enqr(top(deqd(c_xst))) < enqt(c_vi2) == true

Ordered equation sync.27.4.2 into the rewrite rule:
  enqr(top(deqd(c_xst))) < enqt(c_vi2) -> true

Ordered equation sync.27.4.1 into the rewrite rule:
  deqr(top(deqd(c_xst))) < c_vi1 -> true

The system now contains 180 rewrite rules and 12 deduction rules.

Lemma sync.1.2.1.2 in the proof by cases of Lemma sync.1.2.1
    in(c_xh, af(deq(c_xst, c_vi1, c_vi2))) => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true
    Case.7.2: not(deqd(c_xst) = new)
is NOT provable using the current partially completed system.  It reduces to
the equation
    prefix(DEQ(c_xh), ENQ(c_xh)) -> true

Proof of Lemma sync.1.2.1.2 suspended.

-> instantiate xst by c_xst, xh by c_xh,xh1 by c_h1,xh2 by c_h2,xn by c_vi2,xt by c_vi1 in lemma3.3


Equation lemma3.3:
  ((enqr(top(deqd(xst))) < enqt(xn)) <=> false)
   | ((DEQ(xh2) = null) <=> false)
   | ((append(cons(xh1, D(trip(element(xn), enqt(xn), xt))), xh2) = xh)
      <=> false)

   | (false <=> in(append(xh1, xh2), af(xst)))
   | (false <=> in(xn, enqd(xst)))
   | (false <=> least(xn, enqd(xst)))
   | (false <=> prefix(DEQ(append(xh1, xh2)), ENQ(append(xh1, xh2))))
   | prefix(DEQ(xh), ENQ(xh))
  -> true
has been instantiated to equation lemma3.3.1:
  prefix(DEQ(c_xh), ENQ(c_xh)) -> true

Added 1 equation to the system.

Ordered equation lemma3.3.1 into the rewrite rule:
  prefix(DEQ(c_xh), ENQ(c_xh)) -> true

    Left-hand side reduced:
    (false <=> in(xn, enqd(c_xst)))
     | (false <=> least(xn, delete(enqd(c_xst), c_vi2)))
     | (false <=> prefix(DEQ(c_xh), ENQ(c_xh)))
     | (c_vi2 = xn)
     | prefix(cons(DEQ(c_xh), element(xn)), ENQ(c_xh))
    -> true
      became equation sync.26:
      (false <=> in(xn, enqd(c_xst)))
       | (false <=> least(xn, delete(enqd(c_xst), c_vi2)))
       | (false <=> true)
       | (c_vi2 = xn)
       | prefix(cons(DEQ(c_xh), element(xn)), ENQ(c_xh))
      == true
```

```
Ordered equation sync.26 into the rewrite rule:
  (false <=> in(xn, enqd(c_xst)))
   | (false <=> least(xn, delete(enqd(c_xst), c_vi2)))
   | (c_vi2 = xn)
   | prefix(cons(DEQ(c_xh), element(xn)), ENQ(c_xh))
  -> true

The system now contains 181 rewrite rules and 12 deduction rules.

Lemma sync.1.2.1.2 in the proof by cases of Lemma sync.1.2.1
    in(c_xh, af(deq(c_xst, c_vi1, c_vi2))) => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true
    Case.7.2: not(deqd(c_xst) = new)
[] Proved by rewriting.

Lemma sync.1.2.1 in the proof by cases of Lemma sync.1.2
    in(c_xh, af(deq(c_xst, c_vi1, c_vi2))) => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true
    Case.6.1: in(c_xh, af(deq(c_xst, c_vi1, c_vi2)))
[] Proved by cases
     (deqd(c_xst) = new) | not(deqd(c_xst) = new)

Case.6.2
    not(in(c_xh, af(deq(c_xst, c_vi1, c_vi2)))) == true
involves proving Lemma sync.1.2.2
    in(c_xh, af(deq(c_xst, c_vi1, c_vi2))) => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true

The case system now contains 1 equation.

Deduction rule equality.3:
  when  x <=> y == true
  yield x == y
has been applied to equation Case.6.2:
  false <=> in(c_xh, af(deq(c_xst, c_vi1, c_vi2))) == true
to yield the following equations:
  Case.6.2.1: false == in(c_xh, af(deq(c_xst, c_vi1, c_vi2)))

Ordered equation Case.6.2.1 into the rewrite rule:
  in(c_xh, af(deq(c_xst, c_vi1, c_vi2))) -> false

The case system now contains 1 rewrite rule.

Lemma sync.1.2.2 in the proof by cases of Lemma sync.1.2
    in(c_xh, af(deq(c_xst, c_vi1, c_vi2))) => prefix(DEQ(c_xh), ENQ(c_xh))
    -> true
    Case.6.2: not(in(c_xh, af(deq(c_xst, c_vi1, c_vi2))))
[] Proved by rewriting (with unreduced rules).

Lemma sync.1.2 for the induction step in the proof of Conjecture sync.1
    in(xh, af(deq(c_xst, vi1, vi2))) => prefix(DEQ(xh), ENQ(xh)) -> true
[] Proved by cases
    in(xh, af(deq(c_xst, vi1, vi2))) | not(in(xh, af(deq(c_xst, vi1, vi2))))

Conjecture sync.1
    in(xh, af(xst)) => prefix(DEQ(xh), ENQ(xh)) -> true
[] Proved by induction over `xst::St' of sort `St'.

The system now contains 1 equation, 164 rewrite rules, and 12 deduction rules.

Ordered equation sync.1 into the rewrite rule:
  (false <=> in(xh, af(xst))) | prefix(DEQ(xh), ENQ(xh)) -> true

The system now contains 165 rewrite rules and 12 deduction rules.

-> q
```

References

See the bibliography of [5] for a more extensive list of references.

[1] J.M. Wing, M.P. Herlihy, S.M. Clamen, D.L. Detlefs, K. Kietzke, R.A. Lerner, S.-Y. Ling.
The Avalon/C++ Programming Language(Version 0)
CMU-CS-88-209R

[2] M.P. Herlihy, J. M. Wing.
Reasoning About Atomic Objects
Symposium on Formal Techniques in Real-time and Fault-tolerant Systems,
22-23 September 1988, Warwick, U.K., Lecture Notes in Computer Science 331,
Springer-Verlag, pp. 193-208, also CMU-CS-87-176

[3] J.V.Guttag, J.J.Horning, J.M.Wing
Larch in Five Easy Pieces
IEEE Software 2(5):24-36, September, 1985

[4] S.J.Garland, J.V.Guttag
LP: The Larch Prover
Proceedings of the 9th International Conference on Automatic Deduction, LNCS 310

[5] J.M. Wing and C. Gong
Machine-Assisted Proofs of Properties of Avalon Programs
CMU-CS-89-171