```
type msaux = {
  ts_zero: timestamp option
  ts_in: timestamp deque
  ts_out: timestamp deque
  beta_alphas: (timestamp * expl) deque
  beta_alphas_out: (timestamp * expl) deque
  alpha_betas: (timestamp * expl) deque
  alphas_out: (timestamp * vexpl) deque
  betas_suffix_in: (timestamp * vexpl) deque
  alphas_betas_out: (timestamp * vexpl option * vexpl option) deque
}

update_since (a, b) tp ts p1 p2 msaux =
  if (is_none(msaux.ts_zero) and ts − a < 0) or
     (is_some(msaux.ts_zero) and ts < ts_zero + a) then
    l, r = −1
    if is_none(msaux.ts_zero) then update_ts_zero a ts msaux
    else update_ts (l, r) a ts msaux
    update_since_aux (l, r) p1 p2 ts tp msaux
    (V (VSinceOutL tp), msaux)
  else
    l = max {0, (ts − b)}
    r = ts − a
    update_ts (l, r) a ts msaux
    (optimal_proof tp msaux, msaux)

optimal_proof tp msaux =
  if (is_not_empty(msaux.beta_alphas) then
    peek_front msaux.beta_alphas
  else
    p1 = if (is_not_empty(msaux.alpha_betas) then
           peek_front msaux.alpha_betas
    p2 = if (is_not_empty(msaux.alphas_out) then
           vp2 = peek_front msaux.alphas_out
           V (Vsince (tp, vp2, []))
    p3 = if len(msaux.betas_suffix_in) = len(msaux.ts_in) then
           V (VSinceInf (tp, betas_suffix_in))
    min [p1; p2; p3]
```

```
update_since_aux (l, r) tp ts p1 p2 msaux =
  match p1, p2 with
  | S sp1, S sp2 ->
    sp = S (SSince (sp2, []))
    append_to_beta_alphas msaux sp1
    append_to_beta_alphas_out msaux sp1
    enqueue_back msaux.beta_alphas_out (ts, sp)

  | S sp1, V vp2 ->
    append_to_beta_alphas msaux sp1
    append_to_beta_alphas_out msaux sp1
    enqueue_back msaux.alphas_betas_out (ts, None, Some(vp2))

  | V vp1, S sp2 ->
    sp = S (SSince (sp2, []))
    clear msaux.beta_alphas
    clear msaux.beta_alphas_out
    enqueue_back msaux.beta_alphas_out (ts, sp)
    add_alpha_v msaux (ts, V vp1)
    enqueue_back msaux.alphas_betas_out (ts, Some(vp1), None)

  | V vp1, V vp2 ->
    clear msaux.beta_alphas
    clear msaux.beta_alphas_out
    add_alpha_v msaux (ts, V vp1)
    enqueue_back msaux.alphas_betas_out (ts, Some(vp1), Some(vp2)) in

  new_in_sat = split_in_out_beta_alphas_out r msaux
  if is_not_empty(new_in_sat) then update_beta_alphas new_in_sat msaux
  remove_old_beta_alphas l msaux
  new_in_viol = split_in_out_alphas_betas_out r msaux
  if is_not_empty(new_in_viol) then
    update_betas_suffix_in new_in_viol msaux
    update_alpha_betas new_in_viol msaux
    add_new_ps_alpha_betas tp new_in_viol msaux
  remove_old_alpha_betas l msaux
  remove_old_betas_suffix_in l msaux
  remove_old_alphas_out r msaux
  remove_old_alphas_betas_out r msaux
```