```
  tcc_proofs.icalg_TCC1_PROOF
  tcc_proofs.icalg_accuracy_preservation_TCC1_PROOF
  tcc_proofs.icalg_accuracy_preservation_pr_TCC1_PROOF
  tcc_proofs.icalg_precision_enhancement_step_TCC1_PROOF
  tcc_proofs.prec_enh_step3_pr_TCC1_PROOF
  tcc_proofs.sigma_TCC2_PROOF
Total: 61
```

```
  ica4_tcc.icalg_accuracy_preservation_TCC1
  ica4_tcc.icalg_accuracy_preservation_pr_TCC1
  ica_tcc.icalg_TCC1
  ica_tcc.sigma_TCC1
  ica_tcc.sigma_TCC2
  multiplication.distrib
  multiplication.distrib_minus
  multiplication.mult_com
  multiplication.mult_gt
  multiplication.mult_ldistrib_minus
  multiplication.mult_leq_2
  multiplication.mult_lident
  multiplication.mult_rident
  multiplication.pos_product
Total: 60

The completed proofs are:
  absmod.abs_1_bnd_proof
  absmod.abs_2_bnd_proof
  absmod.abs_3_bnd_proof
  absmod.abs_drift_proof
  absmod.abs_leq_0_proof
  absmod.abs_plus_pr
  countmod_tcc.count_TCC1_PROOF
  countmod_tcc.count_TCC2_PROOF
  countmod_tcc.count_TCC3_PROOF
  division.abs_div_pr
  division.div_cancel_pr
  division.div_distrib_pr
  division.div_ineq_pr
  division.div_minus_1_pr
  division.div_minus_distrib_pr
  division.div_nonnegative_pr
  division.mult_div_pr
  division.mult_minus_pr
  division_tcc.ceil_mult_div_TCC1_PROOF
  division_tcc.div_cancel_TCC1_PROOF
  division_tcc.div_ineq_TCC1_PROOF
  division_tcc.div_minus_1_TCC1_PROOF
  division_tcc.div_nonnegative_TCC1_PROOF
  division_tcc.mult_div_1_TCC1_PROOF
  division_tcc.mult_div_TCC1_PROOF
  ica2.okay_Readpred_lr_pr
  ica2.sigma_diff_ind_pr
  ica2.sigma_diff_pr
  ica2.sigma_pos_ind_pr
  ica2.sigma_pos_pr
  ica2.sigma_split_ind_pr
  ica2.sigma_split_pr
  ica3.count_complement_pr
  ica3.mult_sum_ineq_pr
  ica4.acc_pres_sigma_neg_pr
  ica4.acc_pres_sigma_pos_pr
  ica4.acc_pres_step_pr
  ica4.icalg_accuracy_preservation_pr
  ica4.okay_Readpred_fix_diff2_pr
  ica4.okay_Readpred_fix_diff_pr
  ica4.sigma_abs_pr
  ica4.sigma_duplicate_pr
  ica_tcc.sigma_TCC1_PROOF
  multiplication.distrib_minus_pr
  multiplication.distrib_proof
  multiplication.mult_com_pr
  multiplication.mult_gt_pr
  multiplication.mult_ldistrib_minus_proof
  multiplication.mult_leq_2_pr
  multiplication.mult_lident_proof
  multiplication.mult_rident_proof
  multiplication.pos_product_pr
  tcc_proofs.countmod_TCC4_pr
  tcc_proofs.countmod_TCC5_pr
  tcc_proofs.icalg_Pi_TCC1_PROOF
```

```
  clockassumptions.N_maxfaults
  division.mult_div_1
  division.mult_div_2
  division.mult_div_3
  ica3.Delta_0
  multiplication.mult_l0
  multiplication.mult_non_neg
  multiplication.mult_pos
  readbounds.induction
Total: 10

The definitions and type-constraints are:
  absmod.abs
  clockassumptions.okay_Readpred
  clockassumptions.okay_pairs
  countmod.count
  countmod.countsize
  ica.fix
  ica.icalg
  ica.iconv
  ica.sigma
  ica.sigma_size
  multiplication.mult
Total: 11

The formulae used are:
  absmod.abs_1_bnd
  absmod.abs_2_bnd
  absmod.abs_3_bnd
  absmod.abs_drift
  absmod.abs_leq_0
  absmod.abs_plus
  countmod_tcc.count_TCC1
  countmod_tcc.count_TCC2
  countmod_tcc.count_TCC3
  countmod_tcc.count_TCC4
  countmod_tcc.count_TCC5
  division.abs_div
  division.div_cancel
  division.div_distrib
  division.div_ineq
  division.div_minus_1
  division.div_minus_distrib
  division.div_nonnegative
  division.mult_div
  division.mult_minus
  division_tcc.ceil_mult_div_TCC1
  division_tcc.div_cancel_TCC1
  division_tcc.div_ineq_TCC1
  division_tcc.div_minus_1_TCC1
  division_tcc.div_nonnegative_TCC1
  division_tcc.mult_div_1_TCC1
  division_tcc.mult_div_TCC1
  ica2.okay_Readpred_lr
  ica2.sigma_diff
  ica2.sigma_diff_ind
  ica2.sigma_pos
  ica2.sigma_pos_ind
  ica2.sigma_split
  ica2.sigma_split_ind
  ica3.count_complement
  ica3.mult_sum_ineq
  ica3_tcc.icalg_Pi_TCC1
  ica3_tcc.icalg_precision_enhancement_step_TCC1
  ica3_tcc.prec_enh_step3_pr_TCC1
  ica4.acc_pres_sigma_neg
  ica4.acc_pres_sigma_pos
  ica4.acc_pres_step
  ica4.okay_Readpred_fix_diff
  ica4.okay_Readpred_fix_diff2
  ica4.sigma_abs
  ica4.sigma_duplicate
```

C.4 Proof Chain for ICA Accuracy Preservation

```
Terse proof chain for proof icalg_accuracy_preservation_pr in
module ica4

Use of the formula
  ica4.acc_pres_step
requires the following TCCs to be proven
  ica4_tcc.icalg_accuracy_preservation_TCC1
  ica4_tcc.icalg_accuracy_preservation_pr_TCC1

Use of the formula
  ica.sigma
requires the following TCCs to be proven
  ica_tcc.sigma_TCC1
  ica_tcc.sigma_TCC2
  ica_tcc.icalg_TCC1

Formula ica_tcc.sigma_TCC2 is a termination TCC for ica.sigma
Proof of
  ica_tcc.sigma_TCC2
must not use
  ica.sigma

Use of the formula
  countmod.count
requires the following TCCs to be proven
  countmod_tcc.count_TCC1
  countmod_tcc.count_TCC2
  countmod_tcc.count_TCC3
  countmod_tcc.count_TCC4
  countmod_tcc.count_TCC5

Formula countmod_tcc.count_TCC4 is a termination TCC for
countmod.count
Proof of
  countmod_tcc.count_TCC4
must not use
  countmod.count

Formula countmod_tcc.count_TCC5 is a termination TCC for
countmod.count
Proof of
  countmod_tcc.count_TCC5
must not use
  countmod.count

Use of the formula
  ica3.Delta_0
requires the following TCCs to be proven
  ica3_tcc.icalg_Pi_TCC1
  ica3_tcc.icalg_precision_enhancement_step_TCC1
  ica3_tcc.prec_enh_step3_pr_TCC1

Use of the formula
  division.abs_div
requires the following TCCs to be proven
  division_tcc.mult_div_1_TCC1
  division_tcc.mult_div_TCC1
  division_tcc.div_cancel_TCC1
  division_tcc.ceil_mult_div_TCC1
  division_tcc.div_nonnegative_TCC1
  division_tcc.div_ineq_TCC1
  division_tcc.div_minus_1_TCC1

   ================== SUMMARY ==================

The proof chain is complete

The axioms and assumptions at the base are:
  clockassumptions.N_0
```

```
absmod.abs_1_bnd_proof
absmod.abs_2_bnd_proof
absmod.abs_3_bnd_proof
absmod.abs_com_proof
absmod.abs_drift_proof
absmod.abs_leq_0_proof
countmod_tcc.count_TCC1_PROOF
countmod_tcc.count_TCC2_PROOF
countmod_tcc.count_TCC3_PROOF
division.div_distrib_pr
division.div_ineq_pr
division.div_minus_distrib_pr
division.mult_div_pr
division.mult_minus_pr
division_tcc.ceil_mult_div_TCC1_PROOF
division_tcc.div_cancel_TCC1_PROOF
division_tcc.div_ineq_TCC1_PROOF
division_tcc.div_minus_1_TCC1_PROOF
division_tcc.div_nonnegative_TCC1_PROOF
division_tcc.mult_div_1_TCC1_PROOF
division_tcc.mult_div_TCC1_PROOF
ica2.fix_diff1_pr
ica2.fix_diff2_pr
ica2.fix_diff3_pr
ica2.fix_diff_corr_pr
ica2.fix_diff_pr
ica2.iconv_sigma_diff_pr
ica2.okay_Readpred_lr_pr
ica2.okay_Readpred_pairs_pr
ica2.okay_pairs_fix_pr
ica2.okay_pairs_lr_pr
ica2.sigma_diff_ind_pr
ica2.sigma_diff_pr
ica2.sigma_neg_ind_pr
ica2.sigma_neg_ind_step_pr
ica2.sigma_neg_pr
  ica2.sigma_pos_ind_pr
  ica2.sigma_pos_neg_pr
  ica2.sigma_pos_pr
  ica2.sigma_split_ind_pr
  ica2.sigma_split_pr
  ica3.count_complement_pr
  ica3.icalg_precision_enhancement_pr
  ica3.icalg_precision_enhancement_step_pr
  ica3.mult_sum_ineq_pr
  ica3.prec_enh_step2_pr
  ica3.prec_enh_step3_pr
  ica3.prec_enh_step4_pr
  ica3.prec_enh_step_pr
  ica_tcc.sigma_TCC1_PROOF
  multiplication.distrib_minus_pr
  multiplication.distrib_proof
  multiplication.mult_com_pr
  multiplication.mult_gt_pr
  multiplication.mult_ldistrib_minus_proof
  multiplication.mult_leq_2_pr
  multiplication.mult_lident_proof
  multiplication.mult_rident_proof
  tcc_proofs.countmod_TCC4_pr
  tcc_proofs.countmod_TCC5_pr
  tcc_proofs.icalg_Pi_TCC1_PROOF
  tcc_proofs.icalg_TCC1_PROOF
  tcc_proofs.icalg_precision_enhancement_step_TCC1_PROOF
  tcc_proofs.prec_enh_step3_pr_TCC1_PROOF
  tcc_proofs.sigma_TCC2_PROOF
Total: 65
```

```
  ica3.icalg_Pi
  multiplication.mult
Total: 13

The formulae used are:
  absmod.abs_1_bnd
  absmod.abs_2_bnd
  absmod.abs_3_bnd
  absmod.abs_com
  absmod.abs_drift
  absmod.abs_leq_0
  countmod_tcc.count_TCC1
  countmod_tcc.count_TCC2
  countmod_tcc.count_TCC3
  countmod_tcc.count_TCC4
  countmod_tcc.count_TCC5
  division.div_distrib
  division.div_ineq
  division.div_minus_distrib
  division.mult_div
  division.mult_minus
  division_tcc.ceil_mult_div_TCC1
  division_tcc.div_cancel_TCC1
  division_tcc.div_ineq_TCC1
  division_tcc.div_minus_1_TCC1
  division_tcc.div_nonnegative_TCC1
  division_tcc.mult_div_1_TCC1
  division_tcc.mult_div_TCC1
  ica2.fix_diff
  ica2.fix_diff1
  ica2.fix_diff2
  ica2.fix_diff3
  ica2.fix_diff_corr
  ica2.iconv_sigma_diff
  ica2.okay_Readpred_lr
  ica2.okay_Readpred_pairs
  ica2.okay_pairs_fix
  ica2.okay_pairs_lr
  ica2.sigma_diff
  ica2.sigma_diff_ind
  ica2.sigma_neg
  ica2.sigma_neg_ind
  ica2.sigma_neg_ind_step
  ica2.sigma_pos
  ica2.sigma_pos_ind
  ica2.sigma_pos_neg
  ica2.sigma_split
  ica2.sigma_split_ind
  ica3.count_complement
  ica3.icalg_precision_enhancement_step
  ica3.mult_sum_ineq
  ica3.prec_enh_step
  ica3.prec_enh_step2
  ica3.prec_enh_step3
  ica3.prec_enh_step4
  ica3_tcc.icalg_Pi_TCC1
  ica3_tcc.icalg_precision_enhancement_step_TCC1
  ica3_tcc.prec_enh_step3_pr_TCC1
  ica_tcc.icalg_TCC1
  ica_tcc.sigma_TCC1
  ica_tcc.sigma_TCC2
  multiplication.distrib
  multiplication.distrib_minus
  multiplication.mult_com
  multiplication.mult_gt
  multiplication.mult_ldistrib_minus
  multiplication.mult_leq_2
  multiplication.mult_lident
  multiplication.mult_rident
Total: 64

The completed proofs are:
```

```
  countmod_tcc.count_TCC2
  countmod_tcc.count_TCC3
  countmod_tcc.count_TCC4
  countmod_tcc.count_TCC5

Formula countmod_tcc.count_TCC4 is a termination TCC for
countmod.count
Proof of
  countmod_tcc.count_TCC4
must not use
  countmod.count

Formula countmod_tcc.count_TCC5 is a termination TCC for
countmod.count
Proof of
  countmod_tcc.count_TCC5
must not use
  countmod.count

Use of the formula
  division.div_ineq
requires the following TCCs to be proven
  division_tcc.mult_div_1_TCC1
  division_tcc.mult_div_TCC1
  division_tcc.div_cancel_TCC1
  division_tcc.ceil_mult_div_TCC1
  division_tcc.div_nonnegative_TCC1
  division_tcc.div_ineq_TCC1
  division_tcc.div_minus_1_TCC1

Use of the formula
  ica.sigma
requires the following TCCs to be proven
  ica_tcc.sigma_TCC1
  ica_tcc.sigma_TCC2
  ica_tcc.icalg_TCC1

Formula ica_tcc.sigma_TCC2 is a termination TCC for ica.sigma
Proof of
  ica_tcc.sigma_TCC2
must not use
  ica.sigma

   ================== SUMMARY ==================

The proof chain is complete

The axioms and assumptions at the base are:
  clockassumptions.N_0
  clockassumptions.N_maxfaults
  division.mult_div_1
  division.mult_div_2
  division.mult_div_3
  ica3.Delta_0
  multiplication.mult_l0
  multiplication.mult_non_neg
  multiplication.mult_pos
  readbounds.induction
Total: 10

The definitions and type-constraints are:
  absmod.abs
  clockassumptions.okay_Readpred
  clockassumptions.okay_pairs
  countmod.count
  countmod.countsize
  ica.fix
  ica.icalg
  ica.iconv
  ica.sigma
  ica.sigma_size
  ica3.icalg_Mu
```

```
  ica.sigma
  ica.sigma_size
  multiplication.mult
Total: 6

The formulae used are:
  division.div_cancel
  division.div_distrib
  division_tcc.ceil_mult_div_TCC1
  division_tcc.div_cancel_TCC1
  division_tcc.div_ineq_TCC1
  division_tcc.div_minus_1_TCC1
  division_tcc.div_nonnegative_TCC1
  division_tcc.mult_div_1_TCC1
  division_tcc.mult_div_TCC1
  ica.fix_trans
  ica.ica_translation_invariance1
  ica.sigma_trans_inv
  ica.sigma_trans_inv_base
  ica.sigma_trans_inv_ind
  ica_tcc.icalg_TCC1
  ica_tcc.sigma_TCC1
  ica_tcc.sigma_TCC2
  multiplication.distrib
  multiplication.mult_lident
  multiplication.mult_rident
Total: 20

The completed proofs are:
  division.div_cancel_pr
  division.div_distrib_pr
  division_tcc.ceil_mult_div_TCC1_PROOF
  division_tcc.div_cancel_TCC1_PROOF
  division_tcc.div_ineq_TCC1_PROOF
  division_tcc.div_minus_1_TCC1_PROOF
  division_tcc.div_nonnegative_TCC1_PROOF
  division_tcc.mult_div_1_TCC1_PROOF
  division_tcc.mult_div_TCC1_PROOF
  ica.fix_trans_pr
  ica.ica_translation_invariance1_pr
  ica.ica_translation_invariance_pr
  ica.sigma_trans_inv_base_pr
  ica.sigma_trans_inv_ind_pr
  ica.sigma_trans_inv_pr
  ica_tcc.sigma_TCC1_PROOF
  multiplication.distrib_proof
  multiplication.mult_lident_proof
  multiplication.mult_rident_proof
  tcc_proofs.icalg_TCC1_PROOF
  tcc_proofs.sigma_TCC2_PROOF
Total: 21
```

C.3 Proof Chain for ICA Precision Enhancement

```
Terse proof chain for proof icalg_precision_enhancement_pr in
module ica3

Use of the formula
  ica3.prec_enh_step4
requires the following TCCs to be proven
  ica3_tcc.icalg_Pi_TCC1
  ica3_tcc.icalg_precision_enhancement_step_TCC1
  ica3_tcc.prec_enh_step3_pr_TCC1

Use of the formula
  countmod.count
requires the following TCCs to be proven
  countmod_tcc.count_TCC1
```

```
  multiplication.mult_rident_proof
  multiplication.pos_product_pr
  readbounds.Cfn_IClock1_proof
  readbounds.lemma2_abs_fact_proof
  readbounds.lemma_2_base_proof
  readbounds.lemma_2_ind1_proof
  readbounds.lemma_2_ind3_proof
  readbounds.lemma_2_ind_proof
  readbounds.lemma_2_ind_step_proof
  readbounds.lemma_2_proof
  readbounds.okay_Reading_plus_proof
  readbounds.okay_Reading_shift2_proof
  readbounds.okaymaxsync_defn_lr_pr
  readbounds.okaymaxsync_defn_rl_pr
  tcc_proofs.agreement_proof_TCC1_PROOF
  tcc_proofs.agreement_proof_TCC2_PROOF
  tcc_proofs.posnumber_TCC1_PROOF
  tcc_proofs.synctime_multiples_bnd_TCC1_PROOF
  tcc_proofs.synctime_multiples_bnd_TCC2_PROOF
Total: 99
```

C.2 Proof Chain for ICA Translation Invariance

```
Terse proof chain for proof ica_translation_invariance_pr in
module ica

Use of the formula
  ica.ica_translation_invariance1
requires the following TCCs to be proven
  ica_tcc.sigma_TCC1
  ica_tcc.sigma_TCC2
  ica_tcc.icalg_TCC1

Formula ica_tcc.sigma_TCC2 is a termination TCC for ica.sigma
Proof of
  ica_tcc.sigma_TCC2
must not use
  ica.sigma

Use of the formula
  division.div_distrib
requires the following TCCs to be proven
  division_tcc.mult_div_1_TCC1
  division_tcc.mult_div_TCC1
  division_tcc.div_cancel_TCC1
  division_tcc.ceil_mult_div_TCC1
  division_tcc.div_nonnegative_TCC1
  division_tcc.div_ineq_TCC1
  division_tcc.div_minus_1_TCC1

   ================== SUMMARY ==================

The proof chain is complete

The axioms and assumptions at the base are:
  clockassumptions.N_0
  division.mult_div_1
  division.mult_div_2
  division.mult_div_3
  ica.fun_extensionality
  multiplication.mult_10
  readbounds.induction
Total: 7

The definitions and type-constraints are:
  ica.fix
  ica.icalg
  ica.iconv
```

```
basics.abs_shift_proof
basics.lemma_1_1_proof
basics.lemma_1_2_proof
basics.lemma_1_proof
basics.lemma_2_0_proof
basics.lemma_2_1_proof
basics.lemma_2_2a_proof
basics.lemma_2_2b_proof
basics.maxsync_correct_pr
basics.minsync_correct_pr
basics.minsync_maxsync_pr
basics.okay_Reading_shift1_proof
basics.okay_Readvars_shift11_proof
basics.okay_Readvars_shift12_proof
basics.okay_Readvars_shift1_proof
basics.okay_Readvars_shift_proof
basics.okay_Readvars_shift_step2_proof
basics.okay_Readvars_shift_stepb_proof
clockassumptions.okay_Reading_defn_lr_pr
clockassumptions.okay_Reading_defn_rl_pr
clockassumptions.okay_Readpred_Reading_pr
clockassumptions.okay_Readvars_defn_rl_pr
clockassumptions.okay_pairs_Readvars_pr
clockassumptions.precision_enhancement_pr
clockassumptions.rts_0_proof
clockassumptions.rts_1_proof
division.ceil_mult_div_proof
division.ceil_plus_mult_div_proof
division.div_nonnegative_pr
division.mult_div_pr
division_tcc.ceil_mult_div_TCC1_PROOF
division_tcc.div_cancel_TCC1_PROOF
division_tcc.div_ineq_TCC1_PROOF
division_tcc.div_minus_1_TCC1_PROOF
division_tcc.div_nonnegative_TCC1_PROOF
division_tcc.mult_div_1_TCC1_PROOF
division_tcc.mult_div_TCC1_PROOF
lemma3.abs_diff_2_pr
lemma3.accuracy_pres_step0_pr
lemma3.accuracy_pres_step1_pr
lemma3.accuracy_pres_step2_pr
lemma3.accuracy_preservation_pr
lemma3.drift_bnd_proof
lemma3.lemma3_1_1proof
lemma3.lemma3_1_proof
lemma3.lemma3_2_0_proof
lemma3.lemma3_2_1_proof
lemma3.lemma3_2_proof
lemma3.lemma3_2_step1_proof
lemma3.lemma3_2_step2_proof
lemma3.lemma3_2_step3_proof
lemma3.lemma3_2_step_proof
lemma3.lemma3_3_0_proof
lemma3.lemma3_3_ind_proof
lemma3.lemma3_3_proof
lemma3.maxmax_gap_proof
lemma3.maxsync_max_proof
lemma3.minmax_gap_proof
lemma3.minsync_min_proof
lemma3.okayClocks_defn_lr_pr
lemma3.okayClocks_defn_rl_pr
lemma_final.agreement_proof
lemma_final.synctime_multiples_bnd_proof
lemma_final.synctime_multiples_proof
lemma_final.synctime_multiples_step_proof
multiplication.distrib_minus_pr
multiplication.distrib_proof
multiplication.mult_com_pr
multiplication.mult_ldistrib_minus_proof
multiplication.mult_ldistrib_proof
multiplication.mult_leq_pr
multiplication.mult_lident_proof
```

```
division.mult_div
division_tcc.ceil_mult_div_TCC1
division_tcc.div_cancel_TCC1
division_tcc.div_ineq_TCC1
division_tcc.div_minus_1_TCC1
division_tcc.div_nonnegative_TCC1
division_tcc.mult_div_1_TCC1
division_tcc.mult_div_TCC1
lemma3.abs_diff_2
lemma3.accuracy_pres_step0
lemma3.accuracy_pres_step1
lemma3.accuracy_pres_step2
lemma3.accuracy_preservation
lemma3.drift_bnd
lemma3.lemma3_1
lemma3.lemma3_1_1
lemma3.lemma3_2
lemma3.lemma3_2_0
lemma3.lemma3_2_1
lemma3.lemma3_2_step
lemma3.lemma3_2_step1
lemma3.lemma3_2_step2
lemma3.lemma3_2_step3
lemma3.lemma3_3
lemma3.lemma3_3_0
lemma3.lemma3_3_ind
lemma3.maxmax_gap
lemma3.maxsync_max
lemma3.minmax_gap
lemma3.minsync_min
lemma3.okayClocks_defn_lr
lemma3.okayClocks_defn_rl
lemma_final.synctime_multiples
lemma_final.synctime_multiples_bnd
lemma_final.synctime_multiples_step
lemma_final_tcc.agreement_proof_TCC1
  lemma_final_tcc.agreement_proof_TCC2
  lemma_final_tcc.posnumber_TCC1
  lemma_final_tcc.synctime_multiples_bnd_TCC1
  lemma_final_tcc.synctime_multiples_bnd_TCC2
  multiplication.distrib
  multiplication.distrib_minus
  multiplication.mult_com
  multiplication.mult_ldistrib
  multiplication.mult_ldistrib_minus
  multiplication.mult_leq
  multiplication.mult_lident
  multiplication.mult_rident
  multiplication.pos_product
  readbounds.Cfn_IClock1
  readbounds.lemma2_abs_fact
  readbounds.lemma_2
  readbounds.lemma_2_base
  readbounds.lemma_2_ind
  readbounds.lemma_2_ind1
  readbounds.lemma_2_ind3
  readbounds.lemma_2_ind_step
  readbounds.okay_Reading_plus
  readbounds.okay_Reading_shift2
  readbounds.okaymaxsync_defn_lr
  readbounds.okaymaxsync_defn_rl
Total: 98

The completed proofs are:
  absmod.abs_bnd_proof
  absmod.abs_com_proof
  absmod.abs_diff_3_pr
  basics.ReadClock_bnd11_proof
  basics.ReadClock_bnd12_proof
  basics.ReadClock_bnd1_proof
  basics.ReadClock_bnd2_proof
  basics.ReadClock_bnd_proof
```

```
  clockassumptions.rho_0
  clockassumptions.rho_1
  clockassumptions.rmax_0
  clockassumptions.rmin_0
  clockassumptions.rts0
  clockassumptions.rts1
  clockassumptions.rts2
  clockassumptions.rts_2
  clockassumptions.synctime_0
  clockassumptions.translation_invariance
  division.ceil_defn
  division.mult_div_1
  division.mult_div_2
  division.mult_div_3
  multiplication.mult_l0
  multiplication.mult_non_neg
  readbounds.induction
Total: 29

The definitions and type-constraints are:
  absmod.abs
  basics.maxsync
  basics.maxsynctime
  basics.minsync
  clockassumptions.Adj
  clockassumptions.okay_Reading
  clockassumptions.okay_Readpred
  clockassumptions.okay_Readvars
  clockassumptions.okay_pairs
  lemma3.okayClocks
  multiplication.mult
  readbounds.okaymaxsync
Total: 12

The formulae used are:
  absmod.abs_bnd
  absmod.abs_com
  absmod.abs_diff_3
  basics.ReadClock_bnd
  basics.ReadClock_bnd1
  basics.ReadClock_bnd11
  basics.ReadClock_bnd12
  basics.ReadClock_bnd2
  basics.abs_shift
  basics.lemma_1
  basics.lemma_1_1
  basics.lemma_1_2
  basics.lemma_2_0
  basics.lemma_2_1
  basics.lemma_2_2a
  basics.lemma_2_2b
  basics.maxsync_correct
  basics.minsync_correct
  basics.minsync_maxsync
  basics.okay_Reading_shift1
  basics.okay_Readvars_shift
  basics.okay_Readvars_shift1
  basics.okay_Readvars_shift11
  basics.okay_Readvars_shift12
  basics.okay_Readvars_shift_step2
  basics.okay_Readvars_shift_stepb
  clockassumptions.okay_Reading_defn_lr
  clockassumptions.okay_Reading_defn_rl
  clockassumptions.okay_Readpred_Reading
  clockassumptions.okay_Readvars_defn_rl
  clockassumptions.okay_pairs_Readvars
  clockassumptions.precision_enhancement
  clockassumptions.rts_0
  clockassumptions.rts_1
  division.ceil_mult_div
  division.ceil_plus_mult_div
  division.div_nonnegative
```

Appendix C

Proof Chain Analysis

The dependency analysis automatically establishes that there are no unproved statements in the proof that are not axioms or definitions.

C.1 Proof Chain for Agreement

```
Terse proof chain for proof agreement_proof in module
lemma_final

Use of the formula
  lemma_final.synctime_multiples_bnd
requires the following TCCs to be proven
  lemma_final_tcc.posnumber_TCC1
  lemma_final_tcc.synctime_multiples_bnd_TCC1
  lemma_final_tcc.synctime_multiples_bnd_TCC2
  lemma_final_tcc.agreement_proof_TCC1
  lemma_final_tcc.agreement_proof_TCC2

Use of the formula
  division.div_nonnegative
requires the following TCCs to be proven
  division_tcc.mult_div_1_TCC1
  division_tcc.mult_div_TCC1
  division_tcc.div_cancel_TCC1
  division_tcc.ceil_mult_div_TCC1
  division_tcc.div_nonnegative_TCC1
  division_tcc.div_ineq_TCC1
  division_tcc.div_minus_1_TCC1

 ================== SUMMARY ==================

The proof chain is complete

The axioms and assumptions at the base are:
  clockassumptions.IClock_defn
  clockassumptions.Readerror
  clockassumptions.VClock_defn
  clockassumptions.accuracy_preservation_ax
  clockassumptions.beta_0
  clockassumptions.correct_closed
  clockassumptions.correct_count
  clockassumptions.init
  clockassumptions.mu_0
  clockassumptions.precision_enhancement_ax
  clockassumptions.rate_1
  clockassumptions.rate_2
```

top: **Module**

Using arith, lemma_final, ica4, tcc_proofs, tcc_proofs_tcc, division_tcc

Theory

Proof

synctime_multiples_bnd_TCC2_PROOF_TCC1: **Prove**
synctime_multiples_bnd_TCC2_PROOF_TCC1 **from** rmin_0

End top

icalg_precision_enhancement_step_TCC1_PROOF: **Prove**
icalg_precision_enhancement_step_TCC1 **from** N_0

prec_enh_step3_pr_TCC1_PROOF: **Prove**
prec_enh_step3_pr_TCC1 **from** N_maxfaults

icalg_accuracy_preservation_TCC1_PROOF: **Prove**
icalg_accuracy_preservation_TCC1 **from** N_0

icalg_accuracy_preservation_pr_TCC1_PROOF: **Prove**
icalg_accuracy_preservation_pr_TCC1 **from** N_maxfaults

End tcc_proofs

tcc_proofs_tcc: **Module**

Using tcc_proofs

Exporting all with tcc_proofs

Theory

t: **Var** lemma_final.posnumber
i: **Var** naturalnumber

countmod_TCC4_pr_TCC1: **Formula** ((**if** $i > 0$ **then** $i-1$ **else** i **end if**) $\geq$ 0)

synctime_multiples_bnd_TCC2_PROOF_TCC1: **Formula** ($r_{min} \neq 0$)

sigma_TCC2_PROOF_TCC1: **Formula** ((**if** $i > 0$ **then** $i-1$ **else** 0 **end if**) $\geq$ 0)

Proof

countmod_TCC4_pr_TCC1_PROOF: **Prove** countmod_TCC4_pr_TCC1

synctime_multiples_bnd_TCC2_PROOF_TCC1_PROOF: **Prove**
synctime_multiples_bnd_TCC2_PROOF_TCC1

sigma_TCC2_PROOF_TCC1_PROOF: **Prove** sigma_TCC2_PROOF_TCC1

End tcc_proofs_tcc

ica3_tcc: **Module**

Using ica3

Exporting all with ica3

Theory

p: **Var** naturalnumber
q: **Var** naturalnumber
X: **Var** number
Z: **Var** number
fun1: **Var** function[naturalnumber $\rightarrow$ number]
fun2: **Var** function[naturalnumber $\rightarrow$ number]
ppred: **Var** function[naturalnumber $\rightarrow$ boolean]
j: **Var** naturalnumber

icalg_Pi_TCC1: **Formula** $(N \neq 0)$

icalg_precision_enhancement_step_TCC1: **Formula**
$(\text{ppred}(p) \wedge \text{ppred}(q) \wedge \text{count}(\text{ppred}, N) \geq N - \text{maxfaults} \wedge \text{okay\_pairs}(\text{fun1}, \text{fun2}, X, \text{ppred}) \wedge \text{okay\_Readpred}(\text{fun1}, Z, \text{ppred}) \wedge \text{okay\_Readpred}(\text{fun2}, Z, \text{ppred})) \supset (N \neq 0)$

prec_enh_step3_pr_TCC1: **Formula** $(N - \text{maxfaults} \geq 0)$

Proof

icalg_Pi_TCC1_PROOF: **Prove** icalg_Pi_TCC1

icalg_precision_enhancement_step_TCC1_PROOF: **Prove** icalg_precision_enhancement_step_TCC1

prec_enh_step3_pr_TCC1_PROOF: **Prove** prec_enh_step3_pr_TCC1

End ica3_tcc

tcc_proofs: **Module**

Using countmod_tcc, lemma_final_tcc, division, clockassumptions, ica_tcc, ica4_tcc, ica3_tcc

Exporting all with countmod_tcc, lemma_final_tcc, division, clockassumptions, ica_tcc, ica4_tcc, ica3_tcc

Proof

countmod_TCC4_pr: **Prove** count_TCC4 **from**
countsize, countsize $\{i \leftarrow (\textbf{ if } i > 0 \textbf{ then } i - 1 \textbf{ else } i \textbf{ end if})\}$

countmod_TCC5_pr: **Prove** count_TCC5 **from**
countsize, countsize $\{i \leftarrow (\textbf{ if } i > 0 \textbf{ then } i - 1 \textbf{ else } i \textbf{ end if})\}$

posnumber_TCC1_PROOF: **Prove** posnumber_TCC1 $\{x \leftarrow 0\}$

synctime_multiples_bnd_TCC1_PROOF: **Prove** synctime_multiples_bnd_TCC1 **from**
rmin_0

synctime_multiples_bnd_TCC2_PROOF: **Prove** synctime_multiples_bnd_TCC2 **from**
div_nonnegative $\{x \leftarrow t,\ y \leftarrow r_{min}\}$, rmin_0, ceil_defn $\{x \leftarrow t/r_{min}\}$

agreement_proof_TCC1_PROOF: **Prove** agreement_proof_TCC1 **from** rmin_0

agreement_proof_TCC2_PROOF: **Prove** agreement_proof_TCC2 **from**
div_nonnegative $\{x \leftarrow t,\ y \leftarrow r_{min}\}$, rmin_0, ceil_defn $\{x \leftarrow t/r_{min}\}$

sigma_TCC2_PROOF: **Prove** sigma_TCC2 **from**
sigma_size, sigma_size $\{i \leftarrow (\textbf{ if } i > 0 \textbf{ then } i - 1 \textbf{ else } 0 \textbf{ end if})\}$

icalg_TCC1_PROOF: **Prove** icalg_TCC1 **from** N_0

icalg_Pi_TCC1_PROOF:
Prove icalg_Pi_TCC1 **from** N_0

ica_tcc: **Module**

Using ica

Exporting all with ica

Theory

i: **Var** naturalnumber
fun: **Var** function[naturalnumber → number]
j: **Var** naturalnumber
l: **Var** naturalnumber

sigma_TCC1: **Formula** $(i > 0) \supset (i - 1 \geq 0)$

sigma_TCC2: **Formula** $(i > 0)$
$\supset$ sigma_size(fun, i) $>$ sigma_size(fun, $i - 1$)

icalg_TCC1: **Formula** $(N \neq 0)$

Proof

sigma_TCC1_PROOF: **Prove** sigma_TCC1

sigma_TCC2_PROOF: **Prove** sigma_TCC2

icalg_TCC1_PROOF: **Prove** icalg_TCC1

End ica_tcc

ica4_tcc: **Module**

Using ica4

Exporting all with ica4

Theory

p: **Var** naturalnumber
q: **Var** naturalnumber
X: **Var** number
fun: **Var** function[naturalnumber → number]
ppred: **Var** function[naturalnumber → boolean]
p_3: **Var** naturalnumber
j: **Var** naturalnumber

icalg_accuracy_preservation_TCC1: **Formula**
(ppred(p) $\wedge$ ppred(q)
$\wedge$count(ppred, N) $\geq N -$maxfaults$\wedge$okay_Readpred(fun, X, ppred))
$\supset (N \neq 0)$

icalg_accuracy_preservation_pr_TCC1: **Formula** $(N -$ maxfaults $\geq 0)$

Proof

icalg_accuracy_preservation_TCC1_PROOF: **Prove**
icalg_accuracy_preservation_TCC1

icalg_accuracy_preservation_pr_TCC1_PROOF: **Prove**
icalg_accuracy_preservation_pr_TCC1

End ica4_tcc

sigma_abs_pr: **Prove** sigma_abs **from**
induction $\{\text{prop} \leftarrow (\lambda\, i{:}\, |\text{sigma}(\text{fun}, i)| \leq \text{sigma}((\lambda\, p{:}\, |\text{fun}(p)|), i))\}$,
sigma $\{i \leftarrow 0\}$,
$|\star 1|$ $\{x \leftarrow 0\}$,
sigma $\{i \leftarrow 0,\ \text{fun} \leftarrow (\lambda\, p{:}\, |\text{fun}(p)|)\}$,
sigma $\{i \leftarrow j@P1S + 1\}$,
sigma $\{i \leftarrow j@P1S + 1,\ \text{fun} \leftarrow (\lambda\, p{:}\, |\text{fun}(p)|)\}$,
abs_plus $\{x \leftarrow \text{sigma}(\text{fun}, j@P1S),\ y \leftarrow \text{fun}(j@P1S)\}$

acc_pres_sigma_neg_pr: **Prove** acc_pres_sigma_neg **from**
sigma_pos
$\{i \leftarrow N,$
$\text{fun1} \leftarrow (\lambda\, p_1{:}\, |\text{fix}(Y, \text{fun}(p_1), \text{fun}(p)) - \text{fun}(q)|),$
$\text{fun2} \leftarrow (\lambda\, p_1 \rightarrow \text{number}{:}\, 0),$
$\text{ppred} \leftarrow (\lambda\, p_1{:}\, \neg\text{ppred}(p_1)),$
$X \leftarrow X + Y\}$,
okay_pairs
$\{\gamma \leftarrow (\lambda\, p_1{:}\, |\text{fix}(Y, \text{fun}(p_1), \text{fun}(p)) - \text{fun}(q)|),$
$\theta \leftarrow (\lambda\, p_1 \rightarrow \text{number}{:}\, 0),$
$X \leftarrow X + Y,$
$\text{ppred} \leftarrow (\lambda\, p_1{:}\, \neg\text{ppred}(p_1))\}$,
okay_Readpred_fix_diff2 $\{p_1 \leftarrow p_3@\text{P2S}\}$,
$|\star 1|$ $\{x \leftarrow |\text{fix}(Y, \text{fun}(p_3@\text{P2S}), \text{fun}(p)) - \text{fun}(q)|\}$,
$|\star 1|$ $\{x \leftarrow \text{fix}(Y, \text{fun}(p_3@\text{P2S}), \text{fun}(p)) - \text{fun}(q)\}$

acc_pres_sigma_pos_pr: **Prove** acc_pres_sigma_pos **from**
sigma_pos
$\{i \leftarrow N,$
$\text{fun1} \leftarrow (\lambda\, p_1{:}\, |\text{fix}(Y, \text{fun}(p_1), \text{fun}(p)) - \text{fun}(q)|),$
$\text{fun2} \leftarrow (\lambda\, p_1 \rightarrow \text{number}{:}\, 0)\}$,
okay_pairs
$\{\gamma \leftarrow (\lambda\, p_1{:}\, |\text{fix}(Y, \text{fun}(p_1), \text{fun}(p)) - \text{fun}(q)|),$
$\theta \leftarrow (\lambda\, p_1 \rightarrow \text{number}{:}\, 0)\}$,
okay_Readpred_fix_diff $\{p_1 \leftarrow p_3@\text{P2S}\}$,
$|\star 1|$ $\{x \leftarrow |\text{fix}(Y, \text{fun}(p_3@\text{P2S}), \text{fun}(p)) - \text{fun}(q)|\}$,
$|\star 1|$ $\{x \leftarrow \text{fix}(Y, \text{fun}(p_3@\text{P2S}), \text{fun}(p)) - \text{fun}(q)\}$

okay_Readpred_fix_diff2_pr: **Prove** okay_Readpred_fix_diff2 **from**
okay_Readpred_lr $\{\text{fun1} \leftarrow \text{fun},\ Z \leftarrow X\}$,
fix $\{X \leftarrow Y,\ Y \leftarrow \text{fun}(p_1),\ Z \leftarrow \text{fun}(p)\}$,
abs_drift
$\{x_1 \leftarrow \text{fun}(p_1),$
$y \leftarrow \text{fun}(q),$
$x \leftarrow \text{fun}(p),$
$z \leftarrow X,$
$z_1 \leftarrow Y\}$

okay_Readpred_fix_diff_pr: **Prove** okay_Readpred_fix_diff **from**
okay_Readpred_lr $\{\text{fun1} \leftarrow \text{fun},\ Z \leftarrow X\}$,
okay_Readpred_lr $\{\text{fun1} \leftarrow \text{fun},\ p \leftarrow p_1,\ Z \leftarrow X\}$,
fix $\{X \leftarrow Y,\ Y \leftarrow \text{fun}(p_1),\ Z \leftarrow \text{fun}(p)\}$

sigma_duplicate_pr: **Prove** sigma_duplicate **from**
induction $\{\text{prop} \leftarrow (\lambda\, i{:}\, \text{sigma}((\lambda\, i{:}\, x), i) = i \star x)\}$,
sigma $\{i \leftarrow 0,\ \text{fun} \leftarrow (\lambda\, i{:}\, x)\}$,
$\star 1 \star \star 2$ $\{x \leftarrow 0,\ y \leftarrow x\}$,
sigma $\{i \leftarrow j@P1S,\ \text{fun} \leftarrow (\lambda\, i{:}\, x)\}$,
sigma $\{i \leftarrow j@P1S + 1,\ \text{fun} \leftarrow (\lambda\, i{:}\, x)\}$,
distrib $\{x \leftarrow j@P1S,\ y \leftarrow 1,\ z \leftarrow x\}$,
$\star 1 \star \star 2$ $\{x \leftarrow 1,\ y \leftarrow x\}$

End ica4

acc_pres_sigma_neg: **Lemma**

$$\begin{array}{l}
\text{ppred}(p) \wedge \text{ppred}(q) \wedge \text{okay_Readpred}(\text{fun}, X, \text{ppred}) \wedge Y \geq 0 \\
\quad \supset \text{sigma}((\,\lambda\, p_1:\\
\qquad\qquad (\ \textbf{if}\ \neg\text{ppred}(p_1)\\
\qquad\qquad\quad \textbf{then}\ |\text{fix}(Y, \text{fun}(p_1), \text{fun}(p)) - \text{fun}(q)|\\
\qquad\qquad\quad \textbf{else}\ 0\\
\qquad\qquad\quad \textbf{end if})),\\
\qquad N) \leq \text{count}((\,\lambda\, p_1: \neg\text{ppred}(p_1)), N) \star (X + Y)
\end{array}$$

sigma_abs: **Lemma** $|\text{sigma}(\text{fun}, i)| \leq \text{sigma}((\,\lambda\, p: |\text{fun}(p)|), i)$

acc_pres_step: **Lemma**

$$\begin{array}{l}
\text{ppred}(p) \wedge \text{ppred}(q) \wedge \text{okay_Readpred}(\text{fun}, X, \text{ppred})\\
\quad \supset |\text{iconv}(p, \text{fun}, \Delta) - N \star \text{fun}(q)|\\
\qquad \leq \text{count}(\text{ppred}, N) \star X + \text{count}((\,\lambda\, p: \neg\text{ppred}(p)), N) \star (X + \Delta)
\end{array}$$

icalg_accuracy_preservation: **Lemma**

$$\begin{array}{l}
\text{ppred}(p) \wedge \text{ppred}(q)\\
\qquad \wedge \text{count}(\text{ppred}, N) \geq N - \text{maxfaults} \wedge \text{okay_Readpred}(\text{fun}, X, \text{ppred})\\
\quad \supset |\text{icalg}(p, \text{fun}, \Delta) - \text{fun}(q)|\\
\qquad \leq ((N - \text{maxfaults}) \star X + \text{maxfaults} \star (X + \Delta))/N
\end{array}$$

Proof

icalg_accuracy_preservation_pr: **Prove** icalg_accuracy_preservation **from**

$$\begin{array}{l}
\text{acc_pres_step},\\
\text{N_0},\\
\text{abs_div}\ \{x \leftarrow \text{iconv}(p, \text{fun}, \Delta) - N \star \text{fun}(q),\ y \leftarrow N\},\\
\text{icalg}\ \{Y \leftarrow \Delta\},\\
\text{div_cancel}\ \{x \leftarrow N,\ y \leftarrow \text{fun}(q)\},\\
\text{mult_sum_ineq}\\
\quad \{m \leftarrow \text{count}(\text{ppred}, N),\\
\quad\ n \leftarrow \text{count}((\,\lambda\, p: \neg\text{ppred}(p)), N),\\
\quad\ p \leftarrow N - \text{maxfaults},\\
\quad\ q \leftarrow \text{maxfaults},\\
\quad\ x \leftarrow X,\\
\quad\ y \leftarrow X + \Delta\},\\
\text{Delta_0},\\
\text{count_complement}\ \{n \leftarrow N\},\\
\text{div_minus_distrib}\ \{z \leftarrow N,\ x \leftarrow \text{iconv}(p, \text{fun}, \Delta),\ y \leftarrow N \star \text{fun}(q)\},\\
\text{div_ineq}\\
\quad \{z \leftarrow N,\\
\quad\ x \leftarrow |\text{iconv}(p, \text{fun}, \Delta) - N \star \text{fun}(q)|,\\
\quad\ y \leftarrow (N - \text{maxfaults}) \star X + \text{maxfaults} \star (X + \Delta)\}
\end{array}$$

acc_pres_step_pr: **Prove** acc_pres_step **from**

$$\begin{array}{l}
\text{sigma_split}\\
\quad \{\text{fun} \leftarrow (\,\lambda\, p_1: |\text{fix}(\Delta, \text{fun}(p_1), \text{fun}(p)) - \text{fun}(q)|),\\
\quad\ i \leftarrow N\},\\
\text{sigma_abs}\ \{\text{fun} \leftarrow (\,\lambda\, p_1: \text{fix}(\Delta, \text{fun}(p_1), \text{fun}(p)) - \text{fun}(q)),\ i \leftarrow N\},\\
\text{sigma_diff}\\
\quad \{\text{fun1} \leftarrow (\,\lambda\, p_1: \text{fix}(\Delta, \text{fun}(p_1), \text{fun}(p))),\\
\quad\ \text{fun2} \leftarrow (\,\lambda\, p_1: \text{fun}(q)),\\
\quad\ i \leftarrow N\},\\
\text{acc_pres_sigma_neg}\ \{Y \leftarrow \Delta\},\\
\text{acc_pres_sigma_pos}\ \{Y \leftarrow \Delta\},\\
\text{iconv}\ \{Y \leftarrow \Delta\},\\
\text{sigma_duplicate}\ \{x \leftarrow \text{fun}(q),\ i \leftarrow N\},\\
\text{Delta_0}
\end{array}$$

count_complement_pr: **Prove** count_complement **from**
induction
$\{\text{prop} \leftarrow (\lambda\, n\colon \text{count}((\lambda\, q\colon \neg\text{ppred}(q)), n) = n - \text{count}(\text{ppred}, n)),$
$i \leftarrow n\},$
count $\{\text{ppred} \leftarrow (\lambda\, q\colon \neg\text{ppred}(q)),\ i \leftarrow 0\},$
count $\{i \leftarrow 0\},$
count $\{\text{ppred} \leftarrow (\lambda\, q\colon \neg\text{ppred}(q)),\ i \leftarrow j@P1S + 1\},$
count $\{i \leftarrow j@P1S + 1\}$

mult_sum_ineq_pr: **Prove** mult_sum_ineq **from**
distrib $\{x \leftarrow n,\ y \leftarrow q - n,\ z \leftarrow y\},$
distrib $\{x \leftarrow p,\ y \leftarrow m - p,\ z \leftarrow x\},$
mult_leq_2 $\{z \leftarrow q - n,\ x \leftarrow y,\ y \leftarrow x\}$

prec_enh_step_pr: **Prove** prec_enh_step **from**
okay_Readpred_lr, $|\star 1|$ $\{x \leftarrow \text{fun1}(p) - \text{fun1}(q)\}$

End ica3

ica4: **Module**

Using arith, countmod, clockassumptions, readbounds, ica, ica2, ica3

Exporting all with clockassumptions, ica3

Theory

process: **Type is** nat
event: **Type is** nat
time: **Type is** number
Clocktime: **Type is** number
$l, m, n, p, q, p_1, p_2, q_1, q_2, p_3, q_3$: **Var** process
i, j, k: **Var** event
x, y, z, r, s, t: **Var** time
D, X, Y, Z, R, S, T: **Var** Clocktime
fun, fun1, fun2, γ, θ: **Var** function[process $\rightarrow$ Clocktime]
ppred, ppred1, ppred2: **Var** function[process $\rightarrow$ bool]

sigma_duplicate: **Lemma** $\text{sigma}((\lambda\, i\colon x), i) = i \star x$

okay_Readpred_fix_diff: **Lemma**
$\text{ppred}(p) \wedge \text{ppred}(q) \wedge \text{ppred}(p_1) \wedge \text{okay\_Readpred}(\text{fun}, X, \text{ppred})$
$\supset |\text{fix}(Y, \text{fun}(p_1), \text{fun}(p)) - \text{fun}(q)| \leq X$

okay_Readpred_fix_diff2: **Lemma**
$\text{ppred}(p) \wedge \text{ppred}(q) \wedge \text{okay\_Readpred}(\text{fun}, X, \text{ppred}) \wedge Y \geq 0$
$\supset |\text{fix}(Y, \text{fun}(p_1), \text{fun}(p)) - \text{fun}(q)| \leq X + Y$

acc_pres_sigma_pos: **Lemma**
$\text{ppred}(p) \wedge \text{ppred}(q) \wedge \text{okay\_Readpred}(\text{fun}, X, \text{ppred})$
$\supset$ sigma(($\lambda\, p_1$:
(**if** ppred(p_1)
then $|\text{fix}(Y, \text{fun}(p_1), \text{fun}(p)) - \text{fun}(q)|$
else 0
end if)),
$N) \leq \text{count}(\text{ppred}, N) \star X$

prec_enh_step4: **Lemma**
$N > 0 \wedge \text{ppred}(p)$
$\quad \wedge\, \text{ppred}(q)$
$\quad\quad \wedge\, \text{count}(\text{ppred}, N) \geq N - \text{maxfaults}$
$\quad\quad\quad \wedge\, \text{okay\_pairs}(\text{fun1}, \text{fun2}, X, \text{ppred})$
$\quad\quad\quad\quad \wedge \text{okay\_Readpred}(\text{fun1}, Z, \text{ppred}) \wedge \text{okay\_Readpred}(\text{fun2}, Z, \text{ppred})$
$\quad \supset \text{icalg\_Mu}(X, Z, \text{ppred}) \leq \text{icalg\_Pi}(X, Z)$

prec_enh_step4_pr: **Prove** prec_enh_step4 **from**
prec_enh_step,
prec_enh_step2,
prec_enh_step3 $\{Y \leftarrow \Delta\}$,
Delta_0,
icalg_Pi,
icalg_Mu,
div_ineq
$\{x \leftarrow \text{count}(\text{ppred}, N) \star (\textbf{ if } Z \leq \Delta \textbf{ then } X \textbf{ else } X + Z \textbf{ end if})$
$\quad + \text{count}((\lambda\, q_1\colon \neg\text{ppred}(q_1)), N) \star (X + Z + 2 * \Delta),$
$y \leftarrow (N - \text{maxfaults}) \star (\textbf{ if } Z \leq \Delta \textbf{ then } X \textbf{ else } X + Z \textbf{ end if})$
$\quad + \text{maxfaults} \star (X + Z + 2 * \Delta),$
$z \leftarrow N\}$

icalg_precision_enhancement_pr: **Prove** icalg_precision_enhancement **from**
prec_enh_step4, N_0, icalg_precision_enhancement_step, icalg_Mu

icalg_precision_enhancement_step_pr: **Prove** icalg_precision_enhancement_step
from prec_enh_step,
prec_enh_step2,
iconv_sigma_diff $\{Y \leftarrow \Delta\}$,
N_0,
icalg $\{\text{fun} \leftarrow \text{fun1},\ Y \leftarrow \Delta\}$,
icalg $\{p \leftarrow q,\ \text{fun} \leftarrow \text{fun2},\ Y \leftarrow \Delta\}$,
div_minus_distrib
$\{x \leftarrow \text{iconv}(p, \text{fun1}, \Delta),$
$y \leftarrow \text{iconv}(q, \text{fun2}, \Delta),$
$z \leftarrow N\},$
Delta_0,
div_ineq
$\{x \leftarrow \text{iconv}(p, \text{fun1}, \Delta) - \text{iconv}(q, \text{fun2}, \Delta),$
$y \leftarrow \text{count}(\text{ppred}, N) \star (\textbf{ if } Z \leq \Delta \textbf{ then } X \textbf{ else } X + Z \textbf{ end if})$
$\quad + \text{count}((\lambda\, q_1\colon \neg\text{ppred}(q_1)), N) \star (X + Z + 2 * \Delta),$
$z \leftarrow N\}$

prec_enh_step3_pr: **Prove** prec_enh_step3 **from**
count_complement $\{n \leftarrow N\}$,
mult_sum_ineq
$\{m \leftarrow \text{count}(\text{ppred}, N),$
$n \leftarrow \text{count}((\lambda\, q\colon \neg\text{ppred}(q)), N),$
$p \leftarrow N - \text{maxfaults},$
$q \leftarrow \text{maxfaults},$
$x \leftarrow (\textbf{ if } Z \leq Y \textbf{ then } X \textbf{ else } X + Z \textbf{ end if}),$
$y \leftarrow X + Z + 2 * Y\}$

prec_enh_step2_pr: **Prove** prec_enh_step2 **from**
okay_pairs_lr, $|\star 1|$ $\{x \leftarrow \text{fun1}(p) - \text{fun2}(p)\}$

ica3: **Module**

Using arith, countmod, clockassumptions, readbounds, ica, ica2

Exporting all with clockassumptions, ica2

Theory

process: **Type is** nat
event: **Type is** nat
time: **Type is** number
Clocktime: **Type is** number
$l, m, n, p, q, p_1, p_2, q_1, q_2, p_3, q_3$: **Var** process
i, j, k: **Var** event
x, y, z, r, s, t: **Var** time
D, X, Y, Z, R, S, T: **Var** Clocktime
fun, fun1, fun2, γ, θ: **Var** function[process → Clocktime]
ppred, ppred1, ppred2: **Var** function[process → bool]
Δ: Clocktime

Delta_0: **Axiom** $\Delta \geq 0$

mult_sum_ineq: **Lemma**
$m + n = p + q \wedge n \leq q \wedge x \leq y \supset m \star x + n \star y \leq p \star x + q \star y$

count_complement: **Lemma** count((λq: ¬ppred(q)), n) = $n-$count(ppred, n)

prec_enh_step3: **Lemma**
count(ppred, N) $\geq N -$ maxfaults $\wedge X \geq 0 \wedge Y \geq 0 \wedge Z \geq 0$
$\supset$ count(ppred, N) $\star$ (**if** $Z \leq Y$ **then** X **else** $X + Z$ **end if**)
$+$ count((λq_1: ¬ppred(q_1)), N) $\star (X + Z + 2 * Y)$
$\leq N -$ maxfaults $\star$ (**if** $Z \leq Y$ **then** X **else** $X + Z$ **end if**)
$+$ maxfaults $\star (X + Z + 2 * Y)$

icalg_Pi: function[Clocktime, Clocktime → Clocktime] =
($\lambda X, Z$: ($N -$ maxfaults $\star$ (**if** $Z \leq \Delta$ **then** X **else** $X + Z$ **end if**)
$+$ maxfaults $\star (X + Z + 2 * \Delta)$)
$/N$)

prec_enh_step: **Lemma**
ppred(p) $\wedge$ ppred(q) $\wedge$ okay_Readpred(fun1, Z, ppred) $\supset Z \geq 0$

prec_enh_step2: **Lemma** ppred(p) $\wedge$ okay_pairs(fun1, fun2, X, ppred) $\supset X \geq$
0

icalg_precision_enhancement_step: **Lemma**
ppred(p) $\wedge$ ppred(q)
$\wedge$ count(ppred, N) $\geq N -$ maxfaults
$\wedge$ okay_pairs(fun1, fun2, X, ppred)
$\wedge$okay_Readpred(fun1, Z, ppred)$\wedge$okay_Readpred(fun2, Z, ppred)
$\supset$ icalg(p, fun1, Δ) $-$ icalg(q, fun2, Δ)
$\leq$ (count(ppred, N) $\star$ (**if** $Z \leq \Delta$ **then** X **else** $X + Z$ **end if**)
$+$ count((λq_1: ¬ppred(q_1)), N) $\star (X + Z + 2 * \Delta)$)
$/N$

icalg_Mu: function[Clocktime, Clocktime, function[process → bool]
→ Clocktime] =
($\lambda X, Z$, ppred:
(count(ppred, N) $\star$ (**if** $Z \leq \Delta$ **then** X **else** $X + Z$ **end if**)
$+$ count((λq_1: ¬ppred(q_1)), N) $\star (X + Z + 2 * \Delta)$)
$/N$)

icalg_precision_enhancement: **Lemma**
ppred(p) $\wedge$ ppred(q)
$\wedge$ count(ppred, N) $\geq N -$ maxfaults
$\wedge$ okay_pairs(fun1, fun2, X, ppred)
$\wedge$okay_Readpred(fun1, Z, ppred)$\wedge$okay_Readpred(fun2, Z, ppred)
$\supset$ icalg(p, fun1, Δ) $-$ icalg(q, fun2, Δ) $\leq$ icalg_Pi(X, Z)

Proof

```
sigma_pos_pr: Prove sigma_pos from
  induction
    {prop ← ( λ i:
          okay_pairs(fun1, fun2, X, ppred)
            ⊃ sigma(( λ q:
                      ( if ppred(q)
                          then (fun1(q) − fun2(q))
                          else 0
                       end if)),
                    i) ≤ count(ppred, i) ⋆ X)},
  sigma
    {fun ← ( λ q: ( if ppred(q) then (fun1(q) − fun2(q)) else 0 end if)),
     i ← 0},
  count {i ← 0},
  mult_l0 {x ← X},
  sigma_pos_ind {i ← j@P1S}

sigma_split_ind: Lemma
  sigma(fun, i) = sigma(( λ q: ( if ppred(q) then fun(q) else 0 end if)), i)
        + sigma(( λ q: ( if ¬ppred(q) then fun(q) else 0 end if)), i)
    ⊃ sigma(fun, i + 1)
      = sigma(( λ q: ( if ppred(q) then fun(q) else 0 end if)), i + 1)
        + sigma(( λ q: ( if ¬ppred(q) then fun(q) else 0 end if)), i + 1)

sigma_split_ind_pr: Prove sigma_split_ind from
  sigma {i ← i + 1},
  sigma
    {fun ← ( λ q: ( if ppred(q) then fun(q) else 0 end if)),
     i ← i + 1},
  sigma
    {fun ← ( λ q: ( if ¬ppred(q) then fun(q) else 0 end if)),
     i ← i + 1}

sigma_split_pr: Prove sigma_split from
  induction
    {prop ← ( λ i:
          sigma(fun, i)
            = sigma(( λ q: ( if ppred(q) then fun(q) else 0 end if)), i)
              +sigma(( λq: ( if ¬ppred(q) then fun(q) else 0 end if)), i))},
  sigma {i ← 0},
  sigma
    {fun ← ( λ q: ( if ppred(q) then fun(q) else 0 end if)),
     i ← 0},
  sigma
    {fun ← ( λ q: ( if ¬ppred(q) then fun(q) else 0 end if)),
     i ← 0},
  sigma_split_ind {i ← j@P1S}

End ica2
```

sigma_neg_pr: **Prove** sigma_neg **from**
induction
$\{\text{prop} \leftarrow (\lambda\, i:$
$Y \geq 0 \wedge \text{fun1}(p) - \text{fun2}(q) \leq z$
$\supset \text{sigma}((\lambda\, q_1:$
if $\neg\text{ppred}(q_1)$
then $(\text{fix}(Y, \text{fun1}(q_1), \text{fun1}(p))$
$- \text{fix}(Y, \text{fun2}(q_1), \text{fun2}(q)))$
else 0
end if),
$i) \leq \text{count}((\lambda\, q_1: \neg\text{ppred}(q_1)), i) \star (z + 2 * Y))\}$,
sigma
$\{\text{fun} \leftarrow (\lambda\, q_1:$
(**if** $\neg\text{ppred}(q_1)$
then $\text{fix}(Y, \text{fun1}(q_1), \text{fun1}(p)) - \text{fix}(Y, \text{fun2}(q_1), \text{fun2}(q))$
else 0
end if)),
$i \leftarrow 0\}$,
count $\{i \leftarrow 0,\ \text{ppred} \leftarrow (\lambda\, q_1: \neg\text{ppred}(q_1))\}$,
mult_l0 $\{x \leftarrow z + 2 * Y\}$,
sigma_neg_ind $\{i \leftarrow j@P1S\}$

sigma_diff_ind: **Lemma**
$\text{sigma}(\text{fun1}, i) - \text{sigma}(\text{fun2}, i) = \text{sigma}((\lambda\, q: \text{fun1}(q) - \text{fun2}(q)), i)$
$\supset \text{sigma}(\text{fun1}, i+1) - \text{sigma}(\text{fun2}, i+1)$
$= \text{sigma}((\lambda\, q: \text{fun1}(q) - \text{fun2}(q)), i+1)$

sigma_diff_ind_pr: **Prove** sigma_diff_ind **from**
sigma $\{\text{fun} \leftarrow \text{fun1},\ i \leftarrow i+1\}$,
sigma $\{\text{fun} \leftarrow \text{fun2},\ i \leftarrow i+1\}$,
sigma $\{\text{fun} \leftarrow (\lambda\, q: \text{fun1}(q) - \text{fun2}(q)),\ i \leftarrow i+1\}$

sigma_diff_pr: **Prove** sigma_diff **from**
induction
$\{\text{prop} \leftarrow (\lambda\, i:$
$\text{sigma}(\text{fun1}, i) - \text{sigma}(\text{fun2}, i)$
$= \text{sigma}((\lambda\, q: \text{fun1}(q) - \text{fun2}(q)), i))\}$,
sigma $\{\text{fun} \leftarrow \text{fun1},\ i \leftarrow 0\}$,
sigma $\{\text{fun} \leftarrow \text{fun2},\ i \leftarrow 0\}$,
sigma $\{\text{fun} \leftarrow (\lambda\, q: \text{fun1}(q) - \text{fun2}(q)),\ i \leftarrow 0\}$,
sigma_diff_ind $\{i \leftarrow j@P1S\}$

sigma_pos_ind: **Lemma**
okay_pairs(fun1, fun2, X, ppred)
$\wedge$sigma((λq: (**if** ppred(q) **then** (fun1(q)$-$fun2(q)) **else** 0 **end if**)), i)
$\leq \text{count}(\text{ppred}, i) \star X$
$\supset$ sigma(($\lambda\, q$: (**if** ppred(q) **then** (fun1(q) $-$ fun2(q)) **else** 0 **end if**)),
$i+1)$
$\leq \text{count}(\text{ppred}, i+1) \star X$

sigma_pos_ind_pr: **Prove** sigma_pos_ind **from**
sigma
{fun $\leftarrow$ ($\lambda\, q$: (**if** ppred(q) **then** (fun1(q) $-$ fun2(q)) **else** 0 **end if**)),
$i \leftarrow i+1\}$,
okay_pairs $\{\gamma \leftarrow \text{fun1},\ \theta \leftarrow \text{fun2},\ p_3 \leftarrow i\}$,
count $\{i \leftarrow i+1\}$,
distrib $\{x \leftarrow 1,\ y \leftarrow \text{count}(\text{ppred}, i),\ z \leftarrow X\}$,
mult_lident $\{x \leftarrow X\}$,
abs_leq_0 $\{x \leftarrow \text{fun1}(i),\ y \leftarrow \text{fun2}(i),\ z \leftarrow X\}$

okay_pairs_fix_pr: **Prove** okay_pairs_fix **from**
  okay_pairs
    $\{\gamma \leftarrow (\lambda\, q_1 : \text{fix}(Y, \text{fun1}(q_1), \text{fun1}(p))),$
     $\theta \leftarrow (\lambda\, q_1 : \text{fix}(Y, \text{fun2}(q_1), \text{fun2}(q))),$
     $X \leftarrow (\ \textbf{if } Z \leq Y \textbf{ then } X \textbf{ else } X + Z \textbf{ end if})\},$
  fix_diff_corr $\{p_3 \leftarrow p_3@\text{P1S}\}$

sigma_neg_ind_step: **Lemma**
  $Y \geq 0 \land \text{fun1}(p) - \text{fun2}(q) \leq z$
    $\supset \text{fix}(Y, \text{fun1}(i), \text{fun1}(p)) - \text{fix}(Y, \text{fun2}(i), \text{fun2}(q)) \leq z + 2 * Y$

sigma_neg_ind_step_pr: **Prove** sigma_neg_ind_step **from**
  fix $\{X \leftarrow Y,\ Y \leftarrow \text{fun1}(i),\ Z \leftarrow \text{fun1}(p)\}$,
  fix $\{X \leftarrow Y,\ Y \leftarrow \text{fun2}(i),\ Z \leftarrow \text{fun2}(q)\}$,
  abs_leq_0 $\{x \leftarrow \text{fun1}(i),\ y \leftarrow \text{fun1}(p),\ z \leftarrow Y\}$,
  abs_com $\{x \leftarrow \text{fun2}(i),\ y \leftarrow \text{fun2}(q)\}$,
  abs_leq_0 $\{x \leftarrow \text{fun2}(q),\ y \leftarrow \text{fun2}(i),\ z \leftarrow Y\}$

sigma_neg_ind: **Lemma**
  $Y \geq 0 \land \text{fun1}(p) - \text{fun2}(q) \leq z$
      $\land\, \text{sigma}((\lambda\, q_1 :$
            $(\ \textbf{if } \neg\text{ppred}(q_1)$
                $\textbf{then } \text{fix}(Y, \text{fun1}(q_1), \text{fun1}(p))$
                  $- \text{fix}(Y, \text{fun2}(q_1), \text{fun2}(q))$
                $\textbf{else } 0$
                $\textbf{end if})),$
           $i) \leq \text{count}((\lambda\, q_1 : \neg\text{ppred}(q_1)), i) \star (z + 2 * Y)$
    $\supset \text{sigma}((\lambda\, q_1 :$
        $(\ \textbf{if } \neg\text{ppred}(q_1)$
          $\textbf{then } \text{fix}(Y, \text{fun1}(q_1), \text{fun1}(p)) - \text{fix}(Y, \text{fun2}(q_1), \text{fun2}(q))$
          $\textbf{else } 0$
          $\textbf{end if})),$
        $i + 1)$
     $\leq \text{count}((\lambda\, q_1 : \neg\text{ppred}(q_1)), i + 1) \star (z + 2 * Y)$

sigma_neg_ind_pr: **Prove** sigma_neg_ind **from**
  sigma
    $\{\text{fun} \leftarrow (\lambda\, q_1 :$
        $(\ \textbf{if } \neg\text{ppred}(q_1)$
          $\textbf{then } \text{fix}(Y, \text{fun1}(q_1), \text{fun1}(p)) - \text{fix}(Y, \text{fun2}(q_1), \text{fun2}(q))$
          $\textbf{else } 0$
          $\textbf{end if})),$
     $i \leftarrow i + 1\},$
  count $\{\text{ppred} \leftarrow (\lambda\, q_1 : \neg\text{ppred}(q_1)),\ i \leftarrow i + 1\}$,
  sigma_neg_ind_step,
  distrib
    $\{x \leftarrow 1,$
     $y \leftarrow \text{count}((\lambda\, q_1 : \neg\text{ppred}(q_1)), i),$
     $z \leftarrow z + 2 * Y\},$
  mult_lident $\{x \leftarrow z + 2 * Y\}$

fix_diff1_pr: **Prove** fix_diff1 **from**
fix $\{X \leftarrow Y,\ Y \leftarrow \text{fun1}(p_3),\ Z \leftarrow \text{fun1}(p)\}$,
abs_drift
$\{x_1 \leftarrow \text{fun1}(p)$,
$y \leftarrow \text{fun2}(p_3)$,
$x \leftarrow \text{fun1}(p_3)$,
$z \leftarrow X$,
$z_1 \leftarrow Z\}$,
abs_com $\{x \leftarrow \text{fun1}(p),\ y \leftarrow \text{fun1}(p_3)\}$

fix_diff2: **Lemma** $|\text{fun1}(p_3) - \text{fun2}(p_3)| \le X \wedge |\text{fun2}(p_3) - \text{fun2}(q)| \le Z$
$\supset |\text{fun1}(p_3) - \text{fun2}(q)| \le X + Z$

fix_diff2_pr: **Prove** fix_diff2 **from**
abs_drift
$\{x_1 \leftarrow \text{fun1}(p_3)$,
$y \leftarrow \text{fun2}(q)$,
$x \leftarrow \text{fun2}(p_3)$,
$z_1 \leftarrow X$,
$z \leftarrow Z\}$

fix_diff3: **Lemma** $|\text{fun1}(q) - \text{fun2}(q)| \le X \wedge |\text{fun1}(p) - \text{fun1}(q)| \le Z$
$\supset |\text{fun1}(p) - \text{fun2}(q)| \le X + Z$

fix_diff3_pr: **Prove** fix_diff3 **from**
abs_drift
$\{x_1 \leftarrow \text{fun1}(p)$,
$y \leftarrow \text{fun2}(q)$,
$x \leftarrow \text{fun1}(q)$,
$z_1 \leftarrow Z$,
$z \leftarrow X\}$

fix_diff: **Lemma** $Z \ge 0$
$\wedge\ |\text{fun1}(p_3) - \text{fun2}(p_3)| \le X$
$\wedge\ |\text{fun1}(q) - \text{fun2}(q)| \le X$
$\wedge\ |\text{fun1}(p_3) - \text{fun1}(p)| \le Z$
$\wedge\ |\text{fun2}(p_3) - \text{fun2}(q)| \le Z \wedge |\text{fun1}(p) - \text{fun1}(q)| \le Z$
$\supset |\text{fix}(Y, \text{fun1}(p_3), \text{fun1}(p)) - \text{fix}(Y, \text{fun2}(p_3), \text{fun2}(q))|$
$\le (\ \textbf{if}\ Z \le Y\ \textbf{then}\ X\ \textbf{else}\ X + Z\ \textbf{end if})$

fix_diff_pr: **Prove** fix_diff **from**
fix $\{X \leftarrow Y,\ Y \leftarrow \text{fun1}(p_3),\ Z \leftarrow \text{fun1}(p)\}$,
fix $\{X \leftarrow Y,\ Y \leftarrow \text{fun2}(p_3),\ Z \leftarrow \text{fun2}(q)\}$,
fix_diff1,
fix_diff2,
fix_diff3

okay_pairs_lr_pr: **Prove** okay_pairs_lr **from**
okay_pairs $\{\gamma \leftarrow \text{fun1},\ \theta \leftarrow \text{fun2},\ p_3 \leftarrow p\}$

okay_Readpred_lr_pr: **Prove** okay_Readpred_lr **from**
okay_Readpred $\{\gamma \leftarrow \text{fun1},\ Y \leftarrow Z,\ l \leftarrow p,\ m \leftarrow q\}$

fix_diff_corr: **Lemma**
$Z \ge 0 \wedge \text{ppred}(p)$
$\wedge\ \text{ppred}(q)$
$\wedge\ \text{ppred}(p_3)$
$\wedge$ okay_pairs(fun1, fun2, X, ppred)
$\wedge$okay_Readpred(fun1, Z, ppred)$\wedge$okay_Readpred(fun2, Z, ppred)
$\supset |\text{fix}(Y, \text{fun1}(p_3), \text{fun1}(p)) - \text{fix}(Y, \text{fun2}(p_3), \text{fun2}(q))|$
$\le (\ \textbf{if}\ Z \le Y\ \textbf{then}\ X\ \textbf{else}\ X + Z\ \textbf{end if})$

fix_diff_corr_pr: **Prove** fix_diff_corr **from**
fix_diff,
okay_pairs_lr $\{p \leftarrow p_3\}$,
okay_pairs_lr $\{p \leftarrow q\}$,
okay_Readpred_lr $\{p \leftarrow p_3,\ q \leftarrow p\}$,
okay_Readpred_lr $\{\text{fun1} \leftarrow \text{fun2},\ p \leftarrow p_3\}$,
okay_Readpred_lr

sigma_diff: **Lemma**
$\text{sigma}(\text{fun1}, i) - \text{sigma}(\text{fun2}, i) = \text{sigma}((\lambda\, q\colon \text{fun1}(q) - \text{fun2}(q)), i)$

sigma_neg: **Lemma** $Y \geq 0 \wedge \text{fun1}(p) - \text{fun2}(q) \leq z$
$\supset \text{sigma}((\lambda\, q_1\colon$
$(\ \textbf{if}\ \neg\text{ppred}(q_1)$
$\textbf{then}\ (\text{fix}(Y, \text{fun1}(q_1), \text{fun1}(p)) - \text{fix}(Y, \text{fun2}(q_1), \text{fun2}(q)))$
$\textbf{else}\ 0$
$\textbf{end if})),$
$i) \leq \text{count}((\lambda\, q_1\colon \neg\text{ppred}(q_1)), i) \star (z + 2 * Y)$

sigma_pos_neg: **Lemma**
$Y \geq 0 \wedge Z \geq 0 \wedge \text{ppred}(p)$
$\wedge\ \text{ppred}(q)$
$\wedge$ okay_pairs(fun1, fun2, X, ppred)
$\wedge$okay_Readpred(fun1, Z, ppred)$\wedge$okay_Readpred(fun2, Z, ppred)
$\supset \text{sigma}((\lambda\, q_1\colon \text{fix}(Y, \text{fun1}(q_1), \text{fun1}(p)) - \text{fix}(Y, \text{fun2}(q_1), \text{fun2}(q))), i)$
$\leq \text{count}(\text{ppred}, i) \star (\ \textbf{if}\ Z \leq Y\ \textbf{then}\ X\ \textbf{else}\ X + Z\ \textbf{end if})$
$+ \text{count}((\lambda\, q_1\colon \neg\text{ppred}(q_1)), i) \star (X + Z + 2 * Y)$

iconv_sigma_diff: **Lemma**
$Y \geq 0 \wedge Z \geq 0 \wedge \text{ppred}(p)$
$\wedge\ \text{ppred}(q)$
$\wedge$ okay_pairs(fun1, fun2, X, ppred)
$\wedge$okay_Readpred(fun1, Z, ppred)$\wedge$okay_Readpred(fun2, Z, ppred)
$\supset \text{iconv}(p, \text{fun1}, Y) - \text{iconv}(q, \text{fun2}, Y)$
$\leq \text{count}(\text{ppred}, N) \star (\ \textbf{if}\ Z \leq Y\ \textbf{then}\ X\ \textbf{else}\ X + Z\ \textbf{end if})$
$+ \text{count}((\lambda\, q_1\colon \neg\text{ppred}(q_1)), N) \star (X + Z + 2 * Y)$

okay_Readpred_pairs: **Lemma**
$\text{ppred}(p) \wedge \text{ppred}(q)$
$\wedge$ okay_pairs(fun1, fun2, X, ppred) $\wedge$ okay_Readpred(fun1, Z, ppred)
$\supset \text{fun1}(p) - \text{fun2}(q) \leq X + Z$

okay_Readpred_lr: **Lemma**
$\text{ppred}(p) \wedge \text{ppred}(q) \wedge$ okay_Readpred(fun1, Z, ppred)
$\supset |\text{fun1}(p) - \text{fun1}(q)| \leq Z$

okay_pairs_lr: **Lemma**
$\text{ppred}(p) \wedge$ okay_pairs(fun1, fun2, X, ppred) $\supset |\text{fun1}(p) - \text{fun2}(p)| \leq X$

Proof

okay_Readpred_pairs_pr: **Prove** okay_Readpred_pairs **from**
okay_pairs $\{\gamma \leftarrow \text{fun1},\ \theta \leftarrow \text{fun2},\ p_3 \leftarrow q\}$,
abs_leq_0 $\{x \leftarrow \text{fun1}(q),\ y \leftarrow \text{fun2}(q),\ z \leftarrow X\}$,
okay_Readpred $\{\gamma \leftarrow \text{fun1},\ Y \leftarrow Z,\ l \leftarrow p,\ m \leftarrow q\}$,
abs_leq_0 $\{x \leftarrow \text{fun1}(p),\ y \leftarrow \text{fun1}(q),\ z \leftarrow Z\}$

iconv_sigma_diff_pr: **Prove** iconv_sigma_diff **from**
sigma_pos_neg $\{i \leftarrow N\}$,
sigma_diff
$\{\text{fun1} \leftarrow (\lambda\, q_1\colon \text{fix}(Y, \text{fun1}(q_1), \text{fun1}(p))),$
$\text{fun2} \leftarrow (\lambda\, q_1\colon \text{fix}(Y, \text{fun2}(q_1), \text{fun2}(q))),$
$i \leftarrow N\}$,
iconv $\{\text{fun} \leftarrow \text{fun1}\}$,
iconv $\{p \leftarrow q,\ \text{fun} \leftarrow \text{fun2}\}$

sigma_pos_neg_pr: **Prove** sigma_pos_neg **from**
sigma_pos
$\{\text{fun1} \leftarrow (\lambda\, q_1\colon \text{fix}(Y, \text{fun1}(q_1), \text{fun1}(p))),$
$\text{fun2} \leftarrow (\lambda\, q_1\colon \text{fix}(Y, \text{fun2}(q_1), \text{fun2}(q))),$
$X \leftarrow (\ \textbf{if}\ Z \leq Y\ \textbf{then}\ X\ \textbf{else}\ X + Z\ \textbf{end if})\}$,
sigma_neg $\{z \leftarrow X + Z\}$,
okay_pairs_fix,
okay_Readpred_pairs,
sigma_split
$\{\text{fun} \leftarrow (\lambda\, q_1\colon \text{fix}(Y, \text{fun1}(q_1), \text{fun1}(p)) - \text{fix}(Y, \text{fun2}(q_1), \text{fun2}(q)))\}$

fix_diff1: **Lemma** $Z \geq 0 \wedge |\text{fun1}(p_3) - \text{fun2}(p_3)| \leq X$
$\wedge\ |\text{fun1}(p_3) - \text{fun1}(p)| \leq Z$
$\supset |\text{fix}(Y, \text{fun1}(p_3), \text{fun1}(p)) - \text{fun2}(p_3)|$
$\leq (\ \textbf{if}\ Z \leq Y\ \textbf{then}\ X\ \textbf{else}\ X + Z\ \textbf{end if})$

sigma_trans_inv_pr: **Prove** sigma_trans_inv **from**
induction
$\{\text{prop} \leftarrow (\lambda\, n\colon \text{sigma}((\lambda\, q_1\colon \text{fun}(q_1) + X), n) = \text{sigma}(\text{fun}, n) + n \star X),$
$i \leftarrow n\},$
sigma_trans_inv_base,
sigma_trans_inv_ind $\{j \leftarrow j@P1\}$,
mult_l0 $\{x \leftarrow X\}$

ica_translation_invariance1_pr: **Prove** ica_translation_invariance1 **from**
iconv,
iconv $\{\text{fun} \leftarrow (\lambda\, q\colon \text{fun}(q) + X)\}$,
fix_trans,
sigma_trans_inv $\{\text{fun} \leftarrow (\lambda\, q\colon \text{fix}(Y, \text{fun}(q), \text{fun}(p))),\ n \leftarrow N\}$

ica_translation_invariance_pr: **Prove** ica_translation_invariance **from**
ica_translation_invariance1,
icalg,
icalg $\{\text{fun} \leftarrow (\lambda\, q\colon \text{fun}(q) + X)\}$,
div_distrib $\{x \leftarrow \text{iconv}(p, \text{fun}, Y),\ y \leftarrow N \star X,\ z \leftarrow N\}$,
div_cancel $\{x \leftarrow N,\ y \leftarrow X\}$

End ica

ica2: **Module**

Using arith, countmod, clockassumptions, readbounds, ica

Exporting all with ica

Theory

process: **Type is** nat
event: **Type is** nat
time: **Type is** number
Clocktime: **Type is** number
$l, m, n, p, q, p_1, p_2, q_1, q_2, p_3, q_3$: **Var** process
i, j, k: **Var** event
x, y, z, r, s, t: **Var** time
D, X, Y, Z, R, S, T: **Var** Clocktime
fun, fun1, fun2, γ, θ: **Var** function[process $\rightarrow$ Clocktime]
ppred, ppred1, ppred2: **Var** function[process $\rightarrow$ bool]

sigma_split: **Lemma**
$\text{sigma}(\text{fun}, i) = \text{sigma}((\lambda\, q\colon (\textbf{ if } \text{ppred}(q) \textbf{ then } \text{fun}(q) \textbf{ else } 0 \textbf{ end if})), i)$
$+ \text{sigma}((\lambda\, q\colon (\textbf{ if } \neg\text{ppred}(q) \textbf{ then } \text{fun}(q) \textbf{ else } 0 \textbf{ end if})), i)$

sigma_pos: **Lemma** okay_pairs(fun1, fun2, X, ppred)
$\supset \text{sigma}((\lambda q\colon (\textbf{ if } \text{ppred}(q) \textbf{ then } (\text{fun1}(q) - \text{fun2}(q)) \textbf{ else } 0 \textbf{ end if})), i)$
$\leq \text{count}(\text{ppred}, i) \star X$

okay_pairs_fix: **Lemma**
$Z \geq 0 \wedge \text{ppred}(p)$
$\wedge\, \text{ppred}(q)$
$\wedge$ okay_pairs(fun1, fun2, X, ppred)
$\wedge$okay_Readpred(fun1, Z, ppred)$\wedge$okay_Readpred(fun2, Z, ppred)
$\supset$ okay_pairs($(\lambda\, q_1\colon \text{fix}(Y, \text{fun1}(q_1), \text{fun1}(p)))$,
$(\lambda\, q_1\colon \text{fix}(Y, \text{fun2}(q_1), \text{fun2}(q)))$,
$(\textbf{ if } Z \leq Y \textbf{ then } X \textbf{ else } X + Z \textbf{ end if})$,
ppred)

ica: **Module**

Using arith, countmod, clockassumptions, readbounds

Exporting all with clockassumptions

Theory

process: **Type is** nat
event: **Type is** nat
time: **Type is** number
Clocktime: **Type is** number
$l, m, n, p, q, p_1, p_2, q_1, q_2, p_3, q_3$: **Var** process
i, j, k: **Var** event
x, y, z, r, s, t: **Var** time
X, Y, Z, R, S, T: **Var** Clocktime
fun, γ, θ: **Var** function[process → Clocktime]
ppred, ppred1, ppred2: **Var** function[process → bool]
sigma_size: function[function[process → Clocktime], process → process] =
$(\lambda\, \text{fun}, i{:}\ i)$
sigma: function[function[process → Clocktime], process → Clocktime] =
$(\lambda\, \text{fun}, i{:}\ (\ \textbf{if } i > 0 \textbf{ then } \text{fun}(i-1) + \text{sigma}(\text{fun}, i-1) \textbf{ else } 0 \textbf{ end if}))$
by sigma_size
fix: function[Clocktime, Clocktime, Clocktime → Clocktime] =
$(\lambda\, X, Y, Z{:}\ (\ \textbf{if } |Y - Z| \le X \textbf{ then } Y \textbf{ else } Z \textbf{ end if}))$
iconv: function[process, function[process → Clocktime], Clocktime
→ Clocktime] =
$(\lambda\, p, \text{fun}, Y{:}\ \text{sigma}((\lambda\, q{:}\ \text{fix}(Y, \text{fun}(q), \text{fun}(p))), N))$
icalg: function[process, function[process → Clocktime], Clocktime
→ Clocktime] = $(\lambda\, p, \text{fun}, Y{:}\ \text{iconv}(p, \text{fun}, Y)/N)$

ica_translation_invariance1: **Lemma**
$\text{iconv}(p, (\lambda\, q{:}\ \text{fun}(q) + X), Y) = \text{iconv}(p, \text{fun}, Y) + N \star X$

ica_translation_invariance: **Lemma**
$N > 0 \supset \text{icalg}(p, (\lambda\, q{:}\ \text{fun}(q) + X), Y) = \text{icalg}(p, \text{fun}, Y) + X$

extensionality: **Axiom** $(\forall\, l{:}\ \text{ppred1}(l) = \text{ppred2}(l)) \supset \text{ppred1} = \text{ppred2}$

fun1, fun2: **Var** function[process → time]

fun_extensionality: **Axiom** $(\forall\, l{:}\ \text{fun1}(l) = \text{fun2}(l)) \supset \text{fun1} = \text{fun2}$

sigma_trans_inv: **Lemma** $\text{sigma}((\lambda\, q_1{:}\ \text{fun}(q_1) + X), n) = \text{sigma}(\text{fun}, n) + n \star X$

Proof

fix_trans: **Lemma** $(\lambda\, q{:}$
$\text{fix}(Y, ((\lambda\, q_1{:}\ \text{fun}(q_1) + X)q), ((\lambda\, q_1{:}\ \text{fun}(q_1) + X)p)))$
$= (\lambda\, q{:}\ \text{fix}(Y, \text{fun}(q), \text{fun}(p)) + X)$

fix_trans_pr: **Prove** fix_trans **from**
fun_extensionality
$\{\text{fun1} \leftarrow (\lambda\, q{:}\ \text{fix}(Y, ((\lambda\, q_1{:}\ \text{fun}(q_1) + X)q), ((\lambda\, q_1{:}\ \text{fun}(q_1) + X)p))),$
$\text{fun2} \leftarrow (\lambda\, q{:}\ \text{fix}(Y, \text{fun}(q), \text{fun}(p)) + X)\},$
fix
$\{X \leftarrow Y,$
$Y \leftarrow ((\lambda\, q_1{:}\ \text{fun}(q_1) + X)l@P1S),$
$Z \leftarrow ((\lambda\, q_1{:}\ \text{fun}(q_1) + X)p)\},$
fix $\{X \leftarrow Y,\ Y \leftarrow \text{fun}(l@P1S),\ Z \leftarrow \text{fun}(p)\}$

sigma_trans_inv_base: **Lemma** $\text{sigma}((\lambda\, q_1{:}\ \text{fun}(q_1) + X), 0) = \text{sigma}(\text{fun}, 0)$

sigma_trans_inv_base_pr: **Prove** sigma_trans_inv_base **from**
sigma $\{i \leftarrow 0\}$, sigma $\{\text{fun} \leftarrow (\lambda\, q_1{:}\ \text{fun}(q_1) + X),\ i \leftarrow 0\}$

sigma_trans_inv_ind: **Lemma**
$\text{sigma}((\lambda\, q_1{:}\ \text{fun}(q_1) + X), j) = \text{sigma}(\text{fun}, j) + j \star X$
$\supset \text{sigma}((\lambda\, q_1{:}\ \text{fun}(q_1) + X), j+1) = \text{sigma}(\text{fun}, j+1) + (j+1) \star X$

sigma_trans_inv_ind_pr: **Prove** sigma_trans_inv_ind **from**
sigma $\{\text{fun} \leftarrow (\lambda\, q_1{:}\ \text{fun}(q_1) + X),\ i \leftarrow j+1\}$,
sigma $\{i \leftarrow j+1\}$,
distrib $\{x \leftarrow j,\ y \leftarrow 1,\ z \leftarrow X\}$,
mult_ident $\{x \leftarrow X\}$

synctime_multiples_bnd_proof: **Prove** synctime_multiples_bnd **from**
ceil_plus_mult_div $\{x \leftarrow t,\ y \leftarrow r_{min}\}$,
synctime_multiples $\{i \leftarrow \lceil t/r_{min} \rceil + 1\}$,
rmin_0,
div_nonnegative $\{x \leftarrow t,\ y \leftarrow r_{min}\}$,
ceil_defn $\{x \leftarrow (t/r_{min})\}$

correct_synctime_proof: **Prove** correct_synctime **from** rts1 $\{t \leftarrow t@CS\}$

synctime_multiples_pred: function[nat, nat, posnumber → bool] ==
$(\lambda\, i, p, t\colon \text{correct}(p, t) \wedge t \geq 0 \wedge t < i * r_{min} \supset t_p^i > t)$

synctime_multiples_step: **Lemma**
$\text{correct}(p, t) \wedge t \geq t_p^i \wedge t \geq 0 \supset t_p^i \geq i \star r_{min}$

synctime_multiples_proof: **Prove** synctime_multiples **from**
synctime_multiples_step

synctime_multiples_step_pred: function[nat, nat, posnumber → bool] ==
$(\lambda\, i, p, t\colon \text{correct}(p, t) \wedge t_p^i \leq t \wedge t \geq 0 \supset t_p^i \geq i \star r_{min})$

synctime_multiples_step_proof: **Prove** synctime_multiples_step **from**
readbounds.induction
$\{\text{prop} \leftarrow (\lambda\, i\colon \text{synctime\_multiples\_step\_pred}(i, p, t))\}$,
mult_l0 $\{x \leftarrow r_{min}\}$,
synctime_0,
rts_1 $\{i \leftarrow j@P1\}$,
rmin_0,
correct_closed $\{s \leftarrow t,\ t \leftarrow t_p^{j@P1+1}\}$,
distrib $\{x \leftarrow j@P1,\ y \leftarrow 1,\ z \leftarrow r_{min}\}$,
mult_lident $\{x \leftarrow r_{min}\}$

End lemma_final

lemma_final_tcc: **Module**

Using lemma_final

Exporting all with lemma_final

Theory

p: **Var** naturalnumber
x: **Var** number
j: **Var** naturalnumber
t: **Var** posnumber

posnumber_TCC1: **Formula** $(\exists\, x\colon x \geq 0)$

synctime_multiples_bnd_TCC1: **Formula** $(\text{correct}(p, t) \wedge t \geq 0) \supset (r_{min} \neq 0)$

synctime_multiples_bnd_TCC2: **Formula**
$(\text{correct}(p, t) \wedge t \geq 0) \supset (\lceil t/r_{min} \rceil + 1 \geq 0)$

agreement_proof_TCC1: **Formula** $(r_{min} \neq 0)$

agreement_proof_TCC2: **Formula** $(\lceil t/r_{min} \rceil + 1 \geq 0)$

Proof

posnumber_TCC1_PROOF: **Prove** posnumber_TCC1

synctime_multiples_bnd_TCC1_PROOF: **Prove** synctime_multiples_bnd_TCC1

synctime_multiples_bnd_TCC2_PROOF: **Prove** synctime_multiples_bnd_TCC2

agreement_proof_TCC1_PROOF: **Prove** agreement_proof_TCC1

agreement_proof_TCC2_PROOF: **Prove** agreement_proof_TCC2

End lemma_final_tcc

drift_bnd_proof: **Prove** drift_bnd **from**
lemma_2_2a $\{i \leftarrow j\}$,
lemma_2_2a $\{q \leftarrow p\}$,
lemma_2_2b $\{i \leftarrow j\}$,
lemma_2_2b $\{q \leftarrow p\}$,
mult_ldistrib_minus $\{x \leftarrow s - t,\ y \leftarrow 1,\ z \leftarrow \rho\}$,
mult_ldistrib $\{x \leftarrow s - t,\ y \leftarrow 1,\ z \leftarrow \rho\}$,
abs_shift
$\{r \leftarrow IC_p^i(t)$,
$s \leftarrow IC_q^j(t)$,
$t_1 \leftarrow IC_p^i(s)$,
$t_2 \leftarrow IC_q^j(s)$,
$y \leftarrow (s - t) \star 1$,
$z \leftarrow (s - t) \star \rho$,
$x \leftarrow Y\}$

maxsync_max_proof: **Prove** maxsync_max **from** $(\star 1 \Uparrow \star 2)[\star 3]$

minsync_min_proof: **Prove** minsync_min **from** minsync

End lemma3

lemma_final: **Module**

Using clockassumptions, lemma3, arith, basics

Exporting all with clockassumptions, lemma3

Theory

$p, q, p_1, p_2, q_1, q_2, p_3, q_3, i, j, k$: **Var** nat
l, m, n: **Var** int
x, y, z: **Var** number
posnumber: **Type from** number **with** $(\lambda\, x: x \geq 0)$
r, s, t: **Var** posnumber

correct_synctime: **Lemma** $\text{correct}(p, t) \wedge t < t_p^i + r_{min} \supset t < t_p^{i+1}$

synctime_multiples: **Lemma** $\text{correct}(p, t) \wedge t \geq 0 \wedge t < i \star r_{min} \supset t_p^i > t$

synctime_multiples_bnd: **Lemma** $\text{correct}(p, t) \wedge t \geq 0 \supset t < t_p^{\lceil t/r_{min} \rceil + 1}$

agreement: **Lemma** $\beta \leq r_{min}$
$\wedge\, \mu \leq \delta_S \wedge \pi(2 * \Lambda + 2 * \beta \star \rho, \delta_S + 2 * ((r_{max} + \beta) \star \rho + \Lambda)) \leq \delta_S$
$\wedge\, \delta_S + 2 * r_{max} \star \rho \leq \delta$
$\wedge\, \alpha(\delta_S + 2 * (r_{max} + \beta) \star \rho + 2 * \Lambda) + \Lambda + 2 * \beta \star \rho \leq \delta$
$\wedge\, t \geq 0 \wedge \text{correct}(p, t) \wedge \text{correct}(q, t)$
$\supset |VC_p(t) - VC_q(t)| \leq \delta$

Proof

agreement_proof: **Prove** agreement **from**
lemma3_3 $\{i \leftarrow \lceil t/r_{min} \rceil + 1\}$,
okayClocks_defn_lr $\{i \leftarrow \lceil t/r_{min} \rceil + 1,\ t \leftarrow t@CS\}$,
maxsync_correct $\{s \leftarrow t,\ i \leftarrow \lceil t/r_{min} \rceil + 1\}$,
synctime_multiples_bnd $\{p \leftarrow (p \Uparrow q)[\lceil t/r_{min} \rceil + 1]\}$,
rmin_0,
div_nonnegative $\{x \leftarrow t,\ y \leftarrow r_{min}\}$,
ceil_defn $\{x \leftarrow (t/r_{min})\}$

lemma3_2_1_proof: **Prove** lemma3_2_1 **from**
lemma3_2_0,
VClock_defn $\{p \leftarrow (p \Downarrow q)[i+1],\ i \leftarrow i+1\}$,
VClock_defn $\{p \leftarrow (p \Uparrow q)[i+1]\}$,
drift_bnd
$\{s \leftarrow t,$
$t \leftarrow t^{i+1}_{(p \Downarrow q)[i+1]},$
$q \leftarrow (p \Uparrow q)[i+1],$
$p \leftarrow (p \Downarrow q)[i+1],$
$i \leftarrow i+1,$
$j \leftarrow i,$
$Y \leftarrow \alpha(X + 2 * (r_{max} + \beta) \star \rho + 2 * \Lambda) + \Lambda\}$,
rho_0,
maxsync_correct $\{s \leftarrow t,\ i \leftarrow i+1\}$,
minsync_correct $\{s \leftarrow t,\ i \leftarrow i+1\}$,
correct_closed
$\{p \leftarrow (p \Uparrow q)[i+1],$
$s \leftarrow t,$
$t \leftarrow t^{i+1}_{(p \Downarrow q)[i+1]}\}$,
correct_closed
$\{p \leftarrow (p \Downarrow q)[i+1],$
$s \leftarrow t,$
$t \leftarrow t^{i+1}_{(p \Downarrow q)[i+1]}\}$,
correct_closed $\{s \leftarrow t,\ t \leftarrow t^{i+1}_{(p \Downarrow q)[i+1]}\}$,
correct_closed $\{p \leftarrow q,\ s \leftarrow t,\ t \leftarrow t^{i+1}_{(p \Downarrow q)[i+1]}\}$,
rts1 $\{i \leftarrow i+1,\ p \leftarrow (p \Downarrow q)[i+1]\}$,
mult_leq $\{z \leftarrow \rho,\ y \leftarrow t - t^{i+1}_{(p \Downarrow q)[i+1]},\ x \leftarrow \beta\}$,
rts2 $\{i \leftarrow i+1,\ p \leftarrow (p \Uparrow q)[i+1],\ q \leftarrow (p \Downarrow q)[i+1]\}$

lemma3_2_proof: **Prove** lemma3_2 **from**
lemma3_2_1, lemma3_1, lemma3_2_step2, lemma3_2_step3

lemma3_2_step_proof: **Prove** lemma3_2_step **from**
rts2 $\{p \leftarrow (p \Uparrow q)[i],\ q \leftarrow (p \Downarrow q)[i]\}$,
rts1 $\{p \leftarrow (p \Downarrow q)[i]\}$,
minsync_correct $\{s \leftarrow t\}$,
maxsync_correct $\{s \leftarrow t\}$,
minsync_min,
correct_closed $\{p \leftarrow (p \Downarrow q)[i],\ s \leftarrow t,\ t \leftarrow t^{i}_{(p \Downarrow q)[i]}\}$

lemma3_2_step1_proof: **Prove** lemma3_2_step1 **from**
rts2 $\{p \leftarrow (p \Uparrow q)[i+1],\ q \leftarrow (p \Downarrow q)[i+1]\}$,
rts1 $\{p \leftarrow (p \Downarrow q)[i+1]\}$,
minsync_correct $\{s \leftarrow t,\ i \leftarrow i+1\}$,
maxsync_correct $\{s \leftarrow t,\ i \leftarrow i+1\}$

lemma3_2_step2_proof: **Prove** lemma3_2_step2 **from**
lemma3_2_step $\{i \leftarrow i+1\}$,
lemma3_2_step1,
VClock_defn $\{p \leftarrow (p \Downarrow q)[i+1],\ i \leftarrow i+1\}$,
VClock_defn $\{p \leftarrow (p \Uparrow q)[i+1]\}$,
minsync_correct $\{s \leftarrow t,\ i \leftarrow i+1\}$,
maxsync_correct $\{s \leftarrow t,\ i \leftarrow i+1\}$

lemma3_2_step3_proof: **Prove** lemma3_2_step3 **from**
abs_com $\{x \leftarrow VC_p(t),\ y \leftarrow VC_q(t)\}$,
minsync $\{p \leftarrow p,\ q \leftarrow q,\ i \leftarrow i+1\}$,
$(\star 1 \Uparrow \star 2)[\star 3]$ $\{p \leftarrow p,\ q \leftarrow q,\ i \leftarrow i+1\}$

maxmax_gap_proof: **Prove** maxmax_gap **from**
$(\star 1 \Uparrow \star 2)[\star 3]$ $\{i \leftarrow i+1\}$, $(\star 1 \Uparrow \star 2)[\star 3]$, rts0 $\{t \leftarrow s\}$, rts0 $\{t \leftarrow s,\ p \leftarrow q\}$

minmax_gap_proof: **Prove** minmax_gap **from**
minsync_maxsync $\{i \leftarrow i+1\}$, maxmax_gap

lemma3_3_0_proof: **Prove** lemma3_3_0 **from**
okayClocks_defn_rl $\{i \leftarrow 0\}$,
synctime_0 $\{p \leftarrow (p \Uparrow q)[0]\}$,
synctime_0,
synctime_0 $\{p \leftarrow q\}$,
VClock_defn $\{t \leftarrow t@P1S,\ i \leftarrow 0\}$,
VClock_defn $\{p \leftarrow q,\ t \leftarrow t@P1S,\ i \leftarrow 0\}$,
lemma_2_0,
rts1 $\{t \leftarrow t@P1S,\ i \leftarrow 0\}$,
rts1 $\{p \leftarrow q,\ t \leftarrow t@P1S,\ i \leftarrow 0\}$,
rmin_0

lemma3_1_1proof: **Prove** lemma3_1_1 **from**
lemma_2,
okaymaxsync_defn_lr $\{p \leftarrow p,\ q \leftarrow q\}$,
$t^{\star 3}_{\star 1, \star 2}$,
drift_bnd $\{s \leftarrow t,\ t \leftarrow t^{i}_{(p \Uparrow q)[i]},\ Y \leftarrow X,\ j \leftarrow i\}$,
rho_0,
correct_closed $\{s \leftarrow t,\ t \leftarrow t^{i}_{(p \Uparrow q)[i]}\}$,
correct_closed $\{s \leftarrow t,\ t \leftarrow t^{i}_{(p \Uparrow q)[i]},\ p \leftarrow q\}$,
mult_leq $\{z \leftarrow \rho,\ y \leftarrow t - t^{i}_{(p \Uparrow q)[i]},\ x \leftarrow r_{max}\}$,
maxsync_max,
minsync_min $\{i \leftarrow i + 1\}$,
minmax_gap $\{s \leftarrow t,\ t \leftarrow t^{i}_{p,q}\}$

lemma3_1_proof: **Prove** lemma3_1 **from**
lemma3_1_1,
VClock_defn,
VClock_defn $\{p \leftarrow q\}$,
rts0,
mult_leq $\{z \leftarrow \rho,\ y \leftarrow t - t^{i}_{(p \Uparrow q)[i]},\ x \leftarrow r_{max}\}$,
maxsync_max,
minsync_min $\{i \leftarrow i + 1\}$,
rho_0

lemma3_2_0_proof: **Prove** lemma3_2_0 **from**
lemma3_1_1 $\{p \leftarrow l@P2S,\ q \leftarrow m@P2S,\ t \leftarrow t^{i+1}_{(p \Downarrow q)[i+1]}\}$,
accuracy_preservation
$\{p \leftarrow (p \Downarrow q)[i + 1]$,
$q \leftarrow (p \Uparrow q)[i + 1]$,
$X \leftarrow X + 2 * (r_{max} + \beta) \star \rho\}$,
lemma_1_2 $\{p \leftarrow (p \Downarrow q)[i + 1],\ q \leftarrow (l@P2S \Uparrow m@P2S)[i]\}$,
mult_leq
$\{x \leftarrow r_{max} + \beta$,
$y \leftarrow t^{i+1}_{(p \Downarrow q)[i+1]} - t^{i}_{(l@P2S \Uparrow m@P2S)[i]}$,
$z \leftarrow \rho\}$,
lemma_1_1 $\{q \leftarrow (p \Downarrow q)[i + 1],\ p \leftarrow (l@P2S \Uparrow m@P2S)[i]\}$,
rho_0,
minsync_correct $\{i \leftarrow i + 1,\ s \leftarrow t^{i+1}_{(p \Downarrow q)[i+1]}\}$,
maxsync_correct $\{i \leftarrow i + 1,\ s \leftarrow t^{i+1}_{(p \Downarrow q)[i+1]}\}$,
maxsync_correct
$\{p \leftarrow l@P2S$,
$q \leftarrow m@P2S$,
$s \leftarrow t^{i+1}_{(p \Downarrow q)[i+1]}\}$,
correct_closed
$\{s \leftarrow t^{i+1}_{(p \Downarrow q)[i+1]}$,
$t \leftarrow t^{i}_{(l@P2S \Uparrow m@P2S)[i]}$,
$p \leftarrow (l@P2S \Uparrow m@P2S)[i]\}$

accuracy_preservation_pr: **Prove**
accuracy_preservation $\{l \leftarrow l@P2S,\ m \leftarrow m@P2S\}$ **from**
accuracy_preservation_ax
$\{$ppred $\leftarrow (\lambda\, q\colon \mathrm{correct}(q, t_p^{i+1}))$,
$\gamma \leftarrow \Theta_p^{i+1}$,
$X \leftarrow X + 2 * \Lambda\}$,
okay_Readpred
$\{Y \leftarrow X + 2 * \Lambda$,
ppred $\leftarrow (\lambda\, q\colon \mathrm{correct}(q, t_p^{i+1}))$,
$\gamma \leftarrow \Theta_p^{i+1}\}$,
accuracy_pres_step1 $\{l \leftarrow l@P2S,\ m \leftarrow m@P2S\}$,
accuracy_pres_step2
$\{z \leftarrow \Lambda$,
$y_1 \leftarrow \Theta_p^{i+1} q)$,
$y \leftarrow IC_q^i(t_p^{i+1})$,
$x \leftarrow IC_p^{i+1}(t_p^{i+1})\}$,
ReadClock_bnd1,
ReadClock_bnd2,
correct_count $\{t \leftarrow t_p^{i+1}\}$,
IClock_defn $\{i \leftarrow i+1,\ t \leftarrow t_p^{i+1}\}$,
Adj $\{i \leftarrow i+1\}$

abs_diff_2: **Lemma** $|x-y| \le z \supset x - y \le z \wedge y - x \le z$

abs_diff_2_pr: **Prove** abs_diff_2 **from** $|\star 1|$ $\{x \leftarrow x - y\}$

accuracy_pres_step0_pr: **Prove** accuracy_pres_step0 **from**
okay_Readvars_shift_step2,
okay_Readvars_shift_step2
$\{t_1 \leftarrow t_2$,
$t_2 \leftarrow t_1$,
$s \leftarrow t$,
$t \leftarrow s\}$,
abs_diff_2 $\{x \leftarrow t_1,\ y \leftarrow t_2,\ z \leftarrow x\}$,
abs_com $\{x \leftarrow s,\ y \leftarrow t\}$

accuracy_pres_step1_pr: **Prove** accuracy_pres_step1 **from**
accuracy_pres_step0
$\{y \leftarrow \Lambda$,
$x \leftarrow |IC_l^i(t_p^{i+1}) - IC_m^i(t_p^{i+1})|$,
$s \leftarrow \Theta_p^{i+1} l)$,
$t_1 \leftarrow IC_l^i(t_p^{i+1})$,
$t \leftarrow \Theta_p^{i+1} m)$,
$t_2 \leftarrow IC_m^i(t_p^{i+1})\}$,
Readerror $\{q \leftarrow l\}$,
Readerror $\{q \leftarrow m\}$,
abs_com $\{x \leftarrow IC_l^i(t_p^{i+1}),\ y \leftarrow \Theta_p^{i+1} l)\}$,
abs_com $\{x \leftarrow IC_m^i(t_p^{i+1}),\ y \leftarrow \Theta_p^{i+1} m)\}$

lemma3_3_proof: **Prove** lemma3_3 **from**
lemma3_3_ind $\{i \leftarrow j@P2S\}$,
readbounds.induction
$\{$prop $\leftarrow (\lambda\, i \rightarrow \mathrm{bool}\colon$
$\beta \le r_{min} \wedge \mu \le \delta_S$
$\wedge\, \pi(2 * \Lambda + 2 * \beta \star \rho, \delta_S + 2 * ((r_{max} + \beta) \star \rho + \Lambda)) \le \delta_S$
$\wedge\, \delta_S + 2 * r_{max} \star \rho \le \delta$
$\wedge\, \alpha(\delta_S + 2 * (r_{max} + \beta) \star \rho + 2 * \Lambda) + \Lambda + 2 * \beta \star \rho \le \delta$
$\supset \mathrm{okayClocks}(p, q, i))\}$,
lemma3_3_0,
pos_product $\{x \leftarrow r_{max},\ y \leftarrow \rho\}$,
rmax_0,
rho_0

lemma3_3_ind_proof: **Prove** lemma3_3_ind **from**
lemma3_2 $\{t \leftarrow t@P3S,\ X \leftarrow \delta_S\}$,
okayClocks_defn_lr $\{t \leftarrow t@P3S\}$,
okayClocks_defn_rl $\{i \leftarrow i+1\}$

lemma3_2: **Lemma** $\text{correct}(p, t)$
$\quad \wedge \text{correct}(q, t)$
$\quad\quad \wedge \beta \leq r_{min}$
$\quad\quad\quad \wedge \mu \leq X$
$\quad\quad\quad\quad \wedge \pi(2 * \Lambda + 2 * \beta \star \rho, X + 2 * ((r_{max} + \beta) \star \rho + \Lambda)) \leq X$
$\quad\quad\quad\quad\quad \wedge \alpha(X + 2 * (r_{max} + \beta) \star \rho + 2 * \Lambda) + \Lambda + 2 * \beta \star \rho \leq \delta$
$\quad\quad\quad\quad\quad\quad \wedge X + 2 * r_{max} \star \rho \leq \delta$
$\quad\quad\quad\quad\quad\quad\quad \wedge t \geq t^{i}_{(p \Uparrow q)[i]} \wedge t < t^{i+1}_{(p \Uparrow q)[i+1]}$
$\quad \supset |VC_p(t) - VC_q(t)| \leq \delta$

okayClocks: $\text{function}[\text{process}, \text{process}, \text{nat} \rightarrow \text{bool}] =$
$\quad (\lambda\, p, q, i: (\forall\, t:$
$\quad\quad t \geq 0 \wedge t < t^{i}_{(p \Uparrow q)[i]} \wedge \text{correct}(p, t) \wedge \text{correct}(q, t)$
$\quad\quad\quad \supset |VC_p(t) - VC_q(t)| \leq \delta))$

okayClocks_defn_lr: **Lemma**
$\quad \text{okayClocks}(p, q, i)$
$\quad\quad \supset (\forall\, t: t \geq 0 \wedge t < t^{i}_{(p \Uparrow q)[i]} \wedge \text{correct}(p, t) \wedge \text{correct}(q, t)$
$\quad\quad\quad \supset |VC_p(t) - VC_q(t)| \leq \delta)$

okayClocks_defn_rl: **Lemma**
$\quad (\forall\, t: t \geq 0 \wedge t < t^{i}_{(p \Uparrow q)[i]} \wedge \text{correct}(p, t) \wedge \text{correct}(q, t)$
$\quad\quad\quad \supset |VC_p(t) - VC_q(t)| \leq \delta)$
$\quad\quad \supset \text{okayClocks}(p, q, i)$

lemma3_3_0: **Lemma** $\mu \leq \delta \supset \text{okayClocks}(p, q, 0)$

lemma3_3_ind: **Lemma**
$\quad \beta \leq r_{min} \wedge \mu \leq \delta_S$
$\quad\quad \wedge \pi(2 * \Lambda + 2 * \beta \star \rho, \delta_S + 2 * ((r_{max} + \beta) \star \rho + \Lambda)) \leq \delta_S$
$\quad\quad\quad \wedge \delta_S + 2 * r_{max} \star \rho \leq \delta$
$\quad\quad\quad\quad \wedge \alpha(\delta_S + 2 * (r_{max} + \beta) \star \rho + 2 * \Lambda) + \Lambda + 2 * \beta \star \rho \leq \delta$
$\quad\quad\quad\quad\quad \wedge \text{okayClocks}(p, q, i)$
$\quad \supset \text{okayClocks}(p, q, i + 1)$

lemma3_3: **Lemma** $\beta \leq r_{min}$
$\quad \wedge \mu \leq \delta_S \wedge \pi(2 * \Lambda + 2 * \beta \star \rho, \delta_S + 2 * ((r_{max} + \beta) \star \rho + \Lambda)) \leq \delta_S$
$\quad\quad \wedge \delta_S + 2 * r_{max} \star \rho \leq \delta$
$\quad\quad\quad \wedge \alpha(\delta_S + 2 * (r_{max} + \beta) \star \rho + 2 * \Lambda) + \Lambda + 2 * \beta \star \rho \leq \delta$
$\quad \supset \text{okayClocks}(p, q, i)$

Proof

okayClocks_defn_lr_pr: **Prove** okayClocks_defn_lr **from** okayClocks $\{t \leftarrow t@CS\}$

okayClocks_defn_rl_pr: **Prove** okayClocks_defn_rl $\{t \leftarrow t@P1S\}$ **from** okayClocks

accuracy_pres_step2: **Lemma**
$\quad z \geq 0 \wedge y_1 - z \leq y \wedge y_1 + z \geq y \supset |x - y| \leq |x - y_1| + z$

accuracy_pres_step2_pr: **Prove** accuracy_pres_step2 **from**
$\quad |\star 1|\ \{x \leftarrow x - y\},\ |\star 1|\ \{x \leftarrow x - y_1\}$

accuracy_preservation: **Lemma**

$$\begin{aligned}
&\text{correct}(p, t_p^{i+1})\\
&\quad \wedge \text{correct}(q, t_p^{i+1})\\
&\quad\quad \wedge (\forall\, l, m:\\
&\quad\quad\quad \text{correct}(l, t_p^{i+1}) \wedge \text{correct}(m, t_p^{i+1})\\
&\quad\quad\quad\quad \supset |IC_l^i(t_p^{i+1}) - IC_m^i(t_p^{i+1})| \le X)\\
&\quad \supset |IC_p^{i+1}(t_p^{i+1}) - IC_q^i(t_p^{i+1})| \le \alpha(X + 2 * \Lambda) + \Lambda
\end{aligned}$$

accuracy_pres_step0: **Lemma**

$$|s - t_1| \le y \wedge |t - t_2| \le y \wedge |t_1 - t_2| \le x \supset |s - t| \le 2 * y + x$$

accuracy_pres_step1: **Lemma**

$$\begin{aligned}
&\text{correct}(p, t_p^{i+1}) \wedge \text{correct}(l, t_p^{i+1}) \wedge \text{correct}(m, t_p^{i+1})\\
&\quad \supset |\Theta_p^{i+1} l) - \Theta_p^{i+1} m)|\\
&\quad\quad \le |IC_l^i(t_p^{i+1}) - IC_m^i(t_p^{i+1})| + 2 * \Lambda
\end{aligned}$$

lemma3_1_1: **Lemma**

$$\begin{aligned}
&\text{correct}(p, t) \wedge \text{correct}(q, t)\\
&\quad\quad \wedge \beta \le r_{min}\\
&\quad\quad\quad \wedge \mu \le X\\
&\quad\quad\quad\quad \wedge \pi(2 * \Lambda + 2 * \beta \star \rho, X + 2 * ((r_{max} + \beta) \star \rho + \Lambda)) \le X\\
&\quad\quad\quad\quad\quad \wedge t \ge t^i_{(p \Uparrow q)[i]}\\
&\quad \supset |IC_p^i(t) - IC_q^i(t)| \le X + 2 * (t - t^i_{(p \Uparrow q)[i]}) \star \rho
\end{aligned}$$

lemma3_1: **Lemma** correct(p, t)

$$\begin{aligned}
&\quad \wedge \text{correct}(q, t)\\
&\quad\quad \wedge \beta \le r_{min}\\
&\quad\quad\quad \wedge \mu \le X\\
&\quad\quad\quad\quad \wedge \pi(2 * \Lambda + 2 * \beta \star \rho, X + 2 * ((r_{max} + \beta) \star \rho + \Lambda)) \le X\\
&\quad\quad\quad\quad\quad \wedge t \ge t^i_{(p \Uparrow q)[i]} \wedge t < t^{i+1}_{(p \Downarrow q)[i+1]}\\
&\quad \supset |VC_p(t) - VC_q(t)| \le X + 2 * r_{max} \star \rho
\end{aligned}$$

lemma3_2_0: **Lemma**

$$\begin{aligned}
&\text{correct}(p, t^{i+1}_{(p \Downarrow q)[i+1]})\\
&\quad \wedge \text{correct}(q, t^{i+1}_{(p \Downarrow q)[i+1]})\\
&\quad\quad \wedge \beta \le r_{min}\\
&\quad\quad\quad \wedge \mu \le X \wedge \pi(2 * \Lambda + 2 * \beta \star \rho, X + 2 * ((r_{max} + \beta) \star \rho + \Lambda)) \le X\\
&\quad \supset |IC^{i+1}_{(p \Downarrow q)[i+1]}(t^{i+1}_{(p \Downarrow q)[i+1]})\\
&\quad\quad\quad - IC^i_{(p \Uparrow q)[i+1]}(t^{i+1}_{(p \Downarrow q)[i+1]})|\\
&\quad\quad \le \alpha(X + 2 * (r_{max} + \beta) \star \rho + 2 * \Lambda) + \Lambda
\end{aligned}$$

lemma3_2_1: **Lemma**

$$\begin{aligned}
&\text{correct}(p, t) \wedge \text{correct}(q, t)\\
&\quad\quad \wedge \beta \le r_{min}\\
&\quad\quad\quad \wedge \mu \le X\\
&\quad\quad\quad\quad \wedge \pi(2 * \Lambda + 2 * \beta \star \rho, X + 2 * ((r_{max} + \beta) \star \rho + \Lambda)) \le X\\
&\quad\quad\quad\quad\quad \wedge \alpha(X + 2 * (r_{max} + \beta) \star \rho + 2 * \Lambda) + \Lambda + 2 * \beta \star \rho \le \delta\\
&\quad\quad\quad\quad\quad\quad \wedge t \ge t^{i+1}_{(p \Downarrow q)[i+1]} \wedge t < t^{i+1}_{(p \Uparrow q)[i+1]}\\
&\quad \supset |IC^{i+1}_{(p \Downarrow q)[i+1]}(t) - IC^i_{(p \Uparrow q)[i+1]}(t)| \le \delta
\end{aligned}$$

lemma3_2_step: **Lemma**

$$\begin{aligned}
&\text{correct}(p, t) \wedge \text{correct}(q, t) \wedge \beta \le r_{min} \wedge t \ge t^i_{(p \Downarrow q)[i]} \wedge t < t^i_{(p \Uparrow q)[i]}\\
&\quad \supset t < t^{i+1}_{(p \Downarrow q)[i]}
\end{aligned}$$

lemma3_2_step1: **Lemma**

$$\begin{aligned}
&\text{correct}(p, t) \wedge \text{correct}(q, t) \wedge \beta \le r_{min} \wedge t \ge t^{i+1}_{(p \Downarrow q)[i+1]}\\
&\quad \supset t \ge t^i_{(p \Uparrow q)[i+1]}
\end{aligned}$$

lemma3_2_step2: **Lemma**

$$\begin{aligned}
&\text{correct}(p, t) \wedge \text{correct}(q, t)\\
&\quad\quad \wedge \beta \le r_{min} \wedge t \ge t^{i+1}_{(p \Downarrow q)[i+1]} \wedge t < t^{i+1}_{(p \Uparrow q)[i+1]}\\
&\quad \supset |IC^{i+1}_{(p \Downarrow q)[i+1]}(t) - IC^i_{(p \Uparrow q)[i+1]}(t)|\\
&\quad\quad = |VC_{(p \Downarrow q)[i+1]}(t) - VC_{(p \Uparrow q)[i+1]}(t)|
\end{aligned}$$

lemma3_2_step3: **Lemma**

$$|VC_{(p \Downarrow q)[i+1]}(t) - VC_{(p \Uparrow q)[i+1]}(t)| = |VC_p(t) - VC_q(t)|$$

lemma_2_ind_proof: **Prove** lemma_2_ind **from**
lemma_2_ind3 $\{p \leftarrow (p@P2S \Uparrow q@P2S)[i{+}1],\ q \leftarrow (p@P2S \Downarrow q@P2S)[i{+}1]\}$,
okaymaxsync_defn_rl $\{i \leftarrow i+1\}$,
lemma_2_ind_step
$\{i \leftarrow i+1,$
$p \leftarrow p@P2S,$
$q \leftarrow q@P2S,$
$t \leftarrow t^{i+1}_{p@P2S,q@P2S}\}$,
$t^{\star 3}_{\star 1,\star 2}$ $\{i \leftarrow i+1,\ p \leftarrow p@P2S,\ q \leftarrow q@P2S\}$,
minsync_maxsync $\{i \leftarrow i+1,\ p \leftarrow p@P2S,\ q \leftarrow q@P2S\}$,
maxsync_correct
$\{s \leftarrow t^{i+1}_{p,q},$
$i \leftarrow i+1,$
$p \leftarrow p@P2S,$
$q \leftarrow q@P2S\}$,
minsync_correct
$\{s \leftarrow t^{i+1}_{p,q},$
$i \leftarrow i+1,$
$p \leftarrow p@P2S,$
$q \leftarrow q@P2S\}$

lemma_2_proof: **Prove** lemma_2 **from**
readbounds.induction
$\{\text{prop} \leftarrow (\lambda\, i \rightarrow \text{bool}:$
$\beta \leq r_{min} \wedge \mu \leq X$
$\wedge\, \pi(2 * \Lambda + 2 * \beta \star \rho, X + 2 * ((r_{max} + \beta) \star \rho + \Lambda)) \leq X$
$\supset \text{okaymaxsync}(i, X))\}$,
lemma_2_ind $\{i \leftarrow j@P1S\}$,
lemma_2_base,
mu_0

End readbounds

lemma3: **Module**

Using readbounds, basics, clockassumptions, arith

Exporting all with readbounds

Theory

prop: **Var** function[nat → bool]
$l, m, n, p_0, q_0, p, q, p_1, p_2, q_1, q_2$: **Var** process
i, j, k: **Var** event
$x, y, z, r, s, t, t_1, t_2, x_1, x_2, y_1, y_2$: **Var** time
$X, Y, Z, R, S, T, T_1, T_2, X_1, X_2, Y_1, Y_2$: **Var** Clocktime
γ, θ: **Var** function[process → Clocktime]
abs_IClock_diff: function[nat, Clocktime → bool]
IClock_Reading: function[nat, time → function[process → Clocktime]]
δ_S: time

maxmax_gap: **Lemma**
$\text{correct}(p, s) \wedge \text{correct}(q, s)$
$\wedge\, s \geq t \wedge s \leq t^{i+1}_{(p \Uparrow q)[i+1]} \wedge t \geq t^{i}_{(p \Uparrow q)[i]}$
$\supset s - t \leq r_{max}$

minmax_gap: **Lemma**
$\text{correct}(p, s) \wedge \text{correct}(q, s)$
$\wedge\, s \geq t \wedge s \leq t^{i+1}_{(p \Downarrow q)[i+1]} \wedge t \geq t^{i}_{(p \Uparrow q)[i]}$
$\supset s - t \leq r_{max}$

drift_bnd: **Lemma** $t \leq s$
$\wedge\, \text{correct}(p, s) \wedge \text{correct}(q, s) \wedge |IC^i_p(t) - IC^j_q(t)| \leq Y$
$\supset |IC^i_p(s) - IC^j_q(s)| \leq Y + 2 * (s - t) \star \rho$

maxsync_max: **Lemma** $t^i_{(p \Uparrow q)[i]} \geq t^i_p \wedge t^i_{(p \Uparrow q)[i]} \geq t^i_q$

minsync_min: **Lemma** $t^i_{(p \Downarrow q)[i]} \leq t^i_p \wedge t^i_{(p \Downarrow q)[i]} \leq t^i_q$

lemma_2: **Lemma** $\beta \le r_{min}$
$\wedge\, \mu \le X \wedge \pi(2 * \Lambda + 2 * \beta \star \rho, X + 2 * ((r_{max} + \beta) \star \rho + \Lambda)) \le X$
$\supset$ okaymaxsync(i, X)

induction: **Axiom** prop$(0) \wedge (\forall\, j\colon$ prop$(j) \supset$ prop$(j+1)) \supset$ prop(i)

Proof

okaymaxsync_defn_lr_pr: **Prove** okaymaxsync_defn_lr **from**
okaymaxsync $\{p \leftarrow p@CS,\ q \leftarrow q@CS\}$

okaymaxsync_defn_rl_pr: **Prove**
okaymaxsync_defn_rl $\{p \leftarrow p@P1S,\ q \leftarrow q@P1S\}$ **from** okaymaxsync

lemma_2_base_proof: **Prove** lemma_2_base **from**
$t^{\star 3}_{\star 1, \star 2}$ $\{i \leftarrow 0,\ p \leftarrow p@P4S,\ q \leftarrow q@P4S\}$,
synctime_0 $\{p \leftarrow (p@P4S \Uparrow q@P4S)[0]\}$,
lemma_2_0 $\{p \leftarrow p@P4S,\ q \leftarrow q@P4S\}$,
okaymaxsync_defn_rl $\{i \leftarrow 0\}$

okay_Reading_shift2_proof: **Prove** okay_Reading_shift2 **from**
okay_Reading_shift1, okaymaxsync_defn_lr $\{p \leftarrow p@P1S,\ q \leftarrow q@P1S\}$

Cfn_IClock1_proof: **Prove** Cfn_IClock1 **from**
IClock_defn $\{p \leftarrow q,\ t \leftarrow t_p^{i+1},\ i \leftarrow i+1\}$,
Adj $\{p \leftarrow q,\ i \leftarrow i+1\}$,
translation_invariance
$\{p \leftarrow q,$
$\gamma \leftarrow \Theta_q^{i+1},$
$X \leftarrow PC_q(t_p^{i+1}) - PC_q(t_q^{i+1})\}$,
rate_2 $\{p \leftarrow q,\ s \leftarrow t_p^{i+1},\ t \leftarrow t_q^{i+1}\}$,
rho_1,
pos_product $\{x \leftarrow t_p^{i+1} - t_q^{i+1},\ y \leftarrow 1 - \rho\}$

okay_Reading_plus_proof: **Prove** okay_Reading_plus **from**
okay_Reading_defn_lr $\{p_1 \leftarrow p_1$@P2S, $q_1 \leftarrow q_1$@P2S$\}$,
okay_Reading_defn_rl $\{\gamma \leftarrow (\lambda\, p_1 \to \text{time}\colon \gamma(p_1) + X)\}$

lemma_2_ind1_proof: **Prove** lemma_2_ind1 **from**
precision_enhancement
$\{\theta \leftarrow (\lambda\, p_1 \to \text{time}\colon \Theta_q^{i+1} p_1) + PC_q(t_p^{i+1}) - PC_q(t_q^{i+1})),$
$\gamma \leftarrow \Theta_p^{i+1},$
$X \leftarrow 2 * \Lambda + 2 * \beta \star \rho,$
$Y \leftarrow X + 2 * ((r_{max} + \beta) \star \rho + \Lambda)\}$,
okay_Readvars_shift $\{t \leftarrow t_p^{i+1}\}$,
okay_Reading_shift2 $\{p_1 \leftarrow p,\ s \leftarrow t_p^{i+1}\}$,
okay_Reading_shift2 $\{p_1 \leftarrow q,\ s \leftarrow t_p^{i+1}\}$,
okay_Reading_plus
$\{\gamma \leftarrow \Theta_q^{i+1},$
$t \leftarrow t_p^{i+1},$
$X \leftarrow PC_q(t_p^{i+1}) - PC_q(t_q^{i+1}),$
$Y \leftarrow X + 2 * ((r_{max} + \beta) \star \rho + \Lambda)\}$,
correct_closed $\{p \leftarrow q,\ s \leftarrow t_p^{i+1},\ t \leftarrow t_q^{i+1}\}$

lemma2_abs_fact_proof: **Prove** lemma2_abs_fact **from**
$|\star 1|$ $\{x \leftarrow s - t_1\}$, $|\star 1|$ $\{x \leftarrow s - t_2\}$, $|\star 1|$ $\{x \leftarrow s - t\}$

lemma_2_ind3_proof: **Prove** lemma_2_ind3 **from**
lemma_2_ind1,
lemma2_abs_fact
$\{s \leftarrow IC_p^{i+1}(t_p^{i+1}),$
$t \leftarrow IC_q^{i+1}(t_p^{i+1}),$
$t_1 \leftarrow cfn(q, \Theta_q^{i+1}),$
$t_2 \leftarrow cfn(q, (\lambda\, p_1 \to \text{time}\colon \Theta_q^{i+1} p_1) + \beta \star (1 + \rho))),$
$X \leftarrow X\}$,
lemma_2_1 $\{q \leftarrow p\}$,
Cfn_IClock1

lemma_2_ind_step_proof: **Prove** lemma_2_ind_step **from**
$(\star 1 \Uparrow \star 2)[\star 3]$, minsync, abs_com $\{x \leftarrow IC_p^i(t),\ y \leftarrow IC_q^i(t)\}$

readbounds: **Module**

Using basics, clockassumptions, arith

Exporting all with basics

Theory

$p, q, p_1, p_2, q_1, q_2, l, m, n$: **Var** process
i, j, k: **Var** event
$X, Y, Z, R, S, T, T_1, T_2$: **Var** Clocktime
$x, y, z, r, s, t, t_1, t_2$: **Var** number
γ, θ: **Var** function[process $\rightarrow$ Clocktime]
prop: **Var** function[nat $\rightarrow$ bool]
okaymaxsync: function[nat, Clocktime $\rightarrow$ bool] =

$$(\lambda\, i, X : (\forall\, p, q : \mathrm{correct}(p, t^i_{p,q}) \wedge \mathrm{correct}(q, t^i_{p,q}) \supset |IC^i_p(t^i_{p,q}) - IC^i_q(t^i_{p,q})| \leq X))$$

okaymaxsync_defn_lr: **Lemma**

$$\mathrm{okaymaxsync}(i, X) \supset (\forall\, p, q : \mathrm{correct}(p, t^i_{p,q}) \wedge \mathrm{correct}(q, t^i_{p,q}) \supset |IC^i_p(t^i_{p,q}) - IC^i_q(t^i_{p,q})| \leq X)$$

okaymaxsync_defn_rl: **Lemma**

$$(\forall\, p, q : \mathrm{correct}(p, t^i_{p,q}) \wedge \mathrm{correct}(q, t^i_{p,q}) \supset |IC^i_p(t^i_{p,q}) - IC^i_q(t^i_{p,q})| \leq X) \supset \mathrm{okaymaxsync}(i, X)$$

lemma_2_base: **Lemma** $\mu \leq X \supset \mathrm{okaymaxsync}(0, X)$

okay_Reading_shift2: **Lemma**

$$\mathrm{correct}(p_1, s) \wedge s \geq t^{i+1}_{p_1} \wedge \beta \leq r_{min} \wedge \mathrm{okaymaxsync}(i, X) \supset \mathrm{okay\_Reading}(\Theta^{i+1}_{p_1}, X + 2 * ((r_{max} + \beta) \star \rho + \Lambda), s)$$

Cfn_IClock1: **Lemma**

$$\mathrm{correct}(q, t^{i+1}_p) \wedge \mathrm{correct}(p, t^{i+1}_p) \wedge t^{i+1}_p \geq t^{i+1}_q \supset IC^{i+1}_q(t^{i+1}_p) = \mathit{cfn}(q, (\lambda\, p_1 \rightarrow \mathrm{time} : \Theta^{i+1}_q p_1) + PC_q(t^{i+1}_p) - PC_q(t^{i+1}_q)))$$

okay_Reading_plus: **Lemma**

$$\mathrm{okay\_Reading}(\gamma, Y, t) \supset \mathrm{okay\_Reading}((\lambda\, p_1 \rightarrow \mathrm{time} : \gamma(p_1) + X), Y, t)$$

lemma_2_ind1: **Lemma**

$$\beta \leq r_{min} \wedge \pi(2 * \Lambda + 2 * \beta \star \rho, X + 2 * ((r_{max} + \beta) \star \rho + \Lambda)) \leq X \wedge \mathrm{okaymaxsync}(i, X) \wedge t^{i+1}_p \geq t^{i+1}_q \wedge \mathrm{correct}(p, t^{i+1}_p) \wedge \mathrm{correct}(q, t^{i+1}_p) \supset |\mathit{cfn}(p, \Theta^{i+1}_p) - \mathit{cfn}(q, (\lambda\, p_1 \rightarrow \mathrm{time} : \Theta^{i+1}_q p_1) + PC_q(t^{i+1}_p) - PC_q(t^{i+1}_q)))| \leq X$$

lemma2_abs_fact: **Lemma**

$$t_1 \leq t \wedge t \leq t_2 \wedge |s - t_1| \leq X \wedge |s - t_2| \leq X \supset |s - t| \leq X$$

lemma_2_ind3: **Lemma**

$$\beta \leq r_{min} \wedge \pi(2 * \Lambda + 2 * \beta \star \rho, X + 2 * ((r_{max} + \beta) \star \rho + \Lambda)) \leq X \wedge \mathrm{okaymaxsync}(i, X) \wedge t^{i+1}_p \geq t^{i+1}_q \wedge \mathrm{correct}(p, t^{i+1}_p) \wedge \mathrm{correct}(q, t^{i+1}_p) \supset |IC^{i+1}_p(t^{i+1}_p) - IC^{i+1}_q(t^{i+1}_p)| \leq X$$

lemma_2_ind_step: **Lemma**

$$|IC^i_{(p \Uparrow q)[i]}(t) - IC^i_{(p \Downarrow q)[i]}(t)| \leq X \supset |IC^i_p(t) - IC^i_q(t)| \leq X$$

lemma_2_ind: **Lemma**

$$\beta \leq r_{min} \wedge \pi(2 * \Lambda + 2 * \beta \star \rho, X + 2 * ((r_{max} + \beta) \star \rho + \Lambda)) \leq X \wedge \mathrm{okaymaxsync}(i, X) \supset \mathrm{okaymaxsync}(i + 1, X)$$

lemma_1_proof: **Prove** lemma_1 **from**
rts_1 $\{p \leftarrow q\}$,
rts_2,
rmin_0,
correct_closed $\{p \leftarrow q,\ s \leftarrow t_q^{i+1},\ t \leftarrow t_q^i\}$

lemma_1_2_proof: **Prove** lemma_1_2 **from**
rts_0,
rts_1,
rts_2,
rmin_0,
correct_closed $\{s \leftarrow t_p^{i+1},\ t \leftarrow t_p^i\}$

lemma_2_0_proof: **Prove** lemma_2_0 **from**
synctime_0,
synctime_0 $\{p \leftarrow q\}$,
IClock_defn $\{p \leftarrow q,\ i \leftarrow 0,\ t \leftarrow 0\}$,
IClock_defn $\{i \leftarrow 0,\ t \leftarrow 0\}$,
Adj $\{i \leftarrow 0,\ p \leftarrow q\}$,
Adj $\{i \leftarrow 0\}$,
init $\{p \leftarrow q\}$,
init,
rts_1 $\{p \leftarrow q,\ i \leftarrow 0\}$,
rts_1 $\{i \leftarrow 0\}$,
rmin_0,
mu_0,
abs_bnd $\{x \leftarrow IC_p^0(t_p^0),\ y \leftarrow IC_q^0(t_p^0),\ z \leftarrow \mu\}$

lemma_2_1_proof: **Prove** lemma_2_1 **from**
IClock_defn $\{p \leftarrow q,\ i \leftarrow i+1,\ t \leftarrow t_q^{i+1}\}$,
Adj $\{i \leftarrow i+1,\ p \leftarrow q\}$

lemma_2_2a_proof: **Prove** lemma_2_2a **from**
IClock_defn $\{p \leftarrow q,\ t \leftarrow s\}$,
IClock_defn $\{p \leftarrow q\}$,
rate_1 $\{p \leftarrow q\}$,
correct_closed $\{p \leftarrow q\}$

lemma_2_2b_proof: **Prove** lemma_2_2b **from**
IClock_defn $\{p \leftarrow q,\ t \leftarrow s\}$,
IClock_defn $\{p \leftarrow q\}$,
rate_2 $\{p \leftarrow q\}$,
correct_closed $\{p \leftarrow q\}$

abs_shift_proof: **Prove** abs_shift **from** $|\star 1|$ $\{x \leftarrow r - s\}$, $|\star 1|$ $\{x \leftarrow t_1 - t_2\}$

lemma_1_1_proof: **Prove** lemma_1_1 **from**
rts_1 $\{p \leftarrow q\}$,
rts2 $\{t \leftarrow t_q^{i+1}\}$,
beta_0,
rmin_0,
correct_closed $\{p \leftarrow q,\ s \leftarrow t_q^{i+1},\ t \leftarrow t_q^i\}$

End basics

ReadClock_bnd12_proof: **Prove** ReadClock_bnd12 **from**
ReadClock_bnd2,
lemma_2_2b $\{s \leftarrow t_p^{i+1},\ t \leftarrow t_{p_1}^i\}$,
lemma_1_2 $\{q \leftarrow p_1\}$,
lemma_1 $\{q \leftarrow p,\ p \leftarrow p_1\}$,
mult_ldistrib_minus $\{x \leftarrow t_p^{i+1} - t_{p_1}^i,\ y \leftarrow 1,\ z \leftarrow \rho\}$,
mult_leq $\{x \leftarrow r_{max} + \beta,\ y \leftarrow t_p^{i+1} - t_{p_1}^i,\ z \leftarrow \rho\}$,
mult_rident $\{x \leftarrow t_p^{i+1} - t_{p_1}^i\}$,
rho_0

ReadClock_bnd1_proof: **Prove** ReadClock_bnd1 **from**
Readerror, $|\star 1|$ $\{x \leftarrow \Theta_p^{i+1} q) - IC_q^i(t_p^{i+1})\}$

ReadClock_bnd2_proof: **Prove** ReadClock_bnd2 **from**
Readerror, $|\star 1|$ $\{x \leftarrow \Theta_p^{i+1} q) - IC_q^i(t_p^{i+1})\}$

okay_Readvars_shift_step1_proof: **Prove** okay_Readvars_shift_step1 **from**
okay_Readvars_shift_step, $|\star 1|$ $\{x \leftarrow s - t_1\}$, $|\star 1|$ $\{x \leftarrow t - t_2\}$

okay_Readvars_shift_step2_proof: **Prove** okay_Readvars_shift_step2 **from**
okay_Readvars_shift_stepb, $|\star 1|$ $\{x \leftarrow s - t_1\}$, $|\star 1|$ $\{x \leftarrow t - t_2\}$

okay_Readvars_shift11_proof: **Prove** okay_Readvars_shift11 **from**
ReadClock_bnd2 $\{q \leftarrow p_1\}$,
ReadClock_bnd1 $\{p \leftarrow q,\ q \leftarrow p_1\}$,
correct_closed $\{s \leftarrow t_p^{i+1},\ t \leftarrow t_q^{i+1},\ p \leftarrow p_1\}$,
correct_closed $\{s \leftarrow t_p^{i+1},\ t \leftarrow t_q^{i+1},\ p \leftarrow q\}$,
lemma_2_2b $\{q \leftarrow p_1,\ s \leftarrow t_p^{i+1},\ t \leftarrow t_q^{i+1}\}$,
rate_1 $\{s \leftarrow t_p^{i+1},\ t \leftarrow t_q^{i+1},\ p \leftarrow q\}$,
mult_ldistrib_minus $\{x \leftarrow t_p^{i+1} - t_q^{i+1},\ y \leftarrow 1,\ z \leftarrow \rho\}$,
mult_ldistrib $\{x \leftarrow t_p^{i+1} - t_q^{i+1},\ y \leftarrow 1,\ z \leftarrow \rho\}$,
mult_leq $\{x \leftarrow \beta,\ y \leftarrow t_p^{i+1} - t_q^{i+1},\ z \leftarrow \rho\}$,
rts_2 $\{i \leftarrow i+1\}$,
rho_0

okay_Readvars_shift12_proof: **Prove** okay_Readvars_shift12 **from**
ReadClock_bnd1 $\{q \leftarrow p_1\}$,
ReadClock_bnd2 $\{p \leftarrow q,\ q \leftarrow p_1\}$,
correct_closed $\{s \leftarrow t_p^{i+1},\ t \leftarrow t_q^{i+1},\ p \leftarrow p_1\}$,
correct_closed $\{s \leftarrow t_p^{i+1},\ t \leftarrow t_q^{i+1},\ p \leftarrow q\}$,
lemma_2_2a $\{q \leftarrow p_1,\ s \leftarrow t_p^{i+1},\ t \leftarrow t_q^{i+1}\}$,
rate_2 $\{s \leftarrow t_p^{i+1},\ t \leftarrow t_q^{i+1},\ p \leftarrow q\}$,
mult_ldistrib_minus $\{x \leftarrow t_p^{i+1} - t_q^{i+1},\ y \leftarrow 1,\ z \leftarrow \rho\}$,
mult_ldistrib $\{x \leftarrow t_p^{i+1} - t_q^{i+1},\ y \leftarrow 1,\ z \leftarrow \rho\}$,
mult_leq $\{x \leftarrow \beta,\ y \leftarrow t_p^{i+1} - t_q^{i+1},\ z \leftarrow \rho\}$,
rts_2 $\{i \leftarrow i+1\}$,
rho_0

okay_Readvars_shift1_proof: **Prove** okay_Readvars_shift1 **from**
okay_Readvars_shift11,
okay_Readvars_shift12,
abs_diff_3
$\{y \leftarrow \Theta_q^{i+1} p_1) + (PC_q(t_p^{i+1}) - PC_q(t_q^{i+1})),$
$x \leftarrow \Theta_p^{i+1} p_1),$
$z \leftarrow 2 * \Lambda + 2 * \beta \star \rho\}$

okay_Readvars_shift_step_proof: **Prove** okay_Readvars_shift_step **from**
$|\star 1|$ $\{x \leftarrow s + x - t\}$

okay_Readvars_shift_stepb_proof: **Prove** okay_Readvars_shift_stepb **from**
$|\star 1|$ $\{x \leftarrow s - t\}$, $|\star 1|$ $\{x \leftarrow t_2 - t_1\}$

okay_Readvars_shift_proof: **Prove** okay_Readvars_shift **from**
okay_Readvars_shift1 $\{p_1 \leftarrow p_3@\mathrm{P2S}\}$,
okay_Readvars_defn_rl
$\{\theta \leftarrow (\lambda\, p_1 \rightarrow \mathrm{time}\colon \Theta_q^{i+1} p_1) + PC_q(t_p^{i+1}) - PC_q(t_q^{i+1})),$
$\gamma \leftarrow \Theta_p^{i+1},$
$X \leftarrow 2 * \Lambda + 2 * \beta \star \rho\},$
correct_closed $\{s \leftarrow t,\ t \leftarrow t_p^{i+1}\}$,
correct_closed $\{p \leftarrow q,\ s \leftarrow t,\ t \leftarrow t_p^{i+1}\}$,
correct_closed $\{p \leftarrow p_3@\mathrm{P2S},\ s \leftarrow t,\ t \leftarrow t_p^{i+1}\}$

okay_Reading_shift1_proof: **Prove**
okay_Reading_shift1 $\{p \leftarrow p_1@\mathrm{P1S},\ q \leftarrow q_1@\mathrm{P1S}\}$ **from**
okay_Reading_defn_rl
$\{\gamma \leftarrow \Theta_{p_1}^{i+1},$
$Y \leftarrow X + 2 * ((r_{max} + \beta) \star \rho + \Lambda),$
$t \leftarrow s\},$
ReadClock_bnd $\{p \leftarrow p_1,\ q \leftarrow p_1@\mathrm{P1S},\ q_1 \leftarrow q_1@\mathrm{P1S}\},$
$t^{\star 3}_{\star 1, \star 2}$ $\{p \leftarrow p_1@\mathrm{P1S},\ q \leftarrow q_1@\mathrm{P1S}\},$
maxsync_correct $\{p \leftarrow p_1@\mathrm{P1S},\ q \leftarrow q_1@\mathrm{P1S},\ s \leftarrow t_{p_1}^{i+1}\},$
correct_closed $\{p \leftarrow p_1@\mathrm{P1S},\ t \leftarrow t_{p_1}^{i+1}\},$
correct_closed $\{p \leftarrow q_1@\mathrm{P1S},\ t \leftarrow t_{p_1}^{i+1}\},$
correct_closed $\{p \leftarrow p_1@\mathrm{P1S},\ t \leftarrow t_{p,q}^{i},\ s \leftarrow t_{p_1}^{i+1}\},$
correct_closed $\{p \leftarrow q_1@\mathrm{P1S},\ t \leftarrow t_{p,q}^{i},\ s \leftarrow t_{p_1}^{i+1}\},$
correct_closed $\{p \leftarrow p_1,\ t \leftarrow t_{p_1}^{i+1}\},$
lemma_1_1 $\{q \leftarrow p_1,\ p \leftarrow (p \Uparrow q)[i]\}$

ReadClock_bnd_proof: **Prove** ReadClock_bnd **from**
ReadClock_bnd11 $\{p_1 \leftarrow (q \Uparrow q_1)[i]\},$
ReadClock_bnd12 $\{p_1 \leftarrow (q \Uparrow q_1)[i]\},$
ReadClock_bnd11 $\{q \leftarrow q_1,\ p_1 \leftarrow (q \Uparrow q_1)[i]\},$
ReadClock_bnd12 $\{q \leftarrow q_1,\ p_1 \leftarrow (q \Uparrow q_1)[i]\},$
lemma_1_1 $\{p \leftarrow (q \Uparrow q_1)[i],\ q \leftarrow p\},$
correct_closed
$\{p \leftarrow (q \Uparrow q_1)[i],$
$s \leftarrow t_p^{i+1},$
$t \leftarrow t^i_{(q \Uparrow q_1)[i]}\},$
abs_shift
$\{r \leftarrow IC_q^i(t^i_{q,q_1}),$
$s \leftarrow IC_{q_1}^i(t^i_{q,q_1}),$
$t_1 \leftarrow \Theta_p^{i+1} q),$
$t_2 \leftarrow \Theta_p^{i+1} q_1),$
$y \leftarrow (t_p^{i+1} - t^i_{q,q_1}),$
$z \leftarrow (r_{max} + \beta) \star \rho + \Lambda,$
$x \leftarrow X\},$
$t^{\star 3}_{\star 1, \star 2}$ $\{p \leftarrow q,\ q \leftarrow q_1\},$
maxsync_correct $\{p \leftarrow q,\ q \leftarrow q_1,\ s \leftarrow t_p^{i+1}\}$

ReadClock_bnd11_proof: **Prove** ReadClock_bnd11 **from**
ReadClock_bnd1,
lemma_2_2a $\{s \leftarrow t_p^{i+1},\ t \leftarrow t^i_{p_1}\},$
lemma_1_2 $\{q \leftarrow p_1\},$
lemma_1 $\{q \leftarrow p,\ p \leftarrow p_1\},$
mult_ldistrib $\{x \leftarrow t_p^{i+1} - t^i_{p_1},\ y \leftarrow 1,\ z \leftarrow \rho\},$
mult_leq $\{x \leftarrow r_{max} + \beta,\ y \leftarrow t_p^{i+1} - t^i_{p_1},\ z \leftarrow \rho\},$
mult_rident $\{x \leftarrow t_p^{i+1} - t^i_{p_1}\},$
rho_0

okay_Reading_shift1: **Lemma**

$$\begin{array}{l} \text{correct}(p_1, s) \wedge s \geq t_{p_1}^{i+1} \\ \quad \wedge\, \beta \leq r_{min} \\ \quad\quad \wedge\, (\forall\, p, q: \\ \quad\quad\quad \text{correct}(p, t_{p,q}^{i}) \wedge \text{correct}(q, t_{p,q}^{i}) \\ \quad\quad\quad\quad \supset |IC_p^i(t_{p,q}^i) - IC_q^i(t_{p,q}^i)| \leq X) \\ \quad \supset \text{okay\_Reading}(\Theta_{p_1}^{i+1}, X + 2 * ((r_{max} + \beta) \star \rho + \Lambda), s) \end{array}$$

okay_Readvars_shift_step: **Lemma**

$$\begin{array}{l} s \geq t_1 - y \wedge s \leq t_1 + y \\ \quad \wedge\, t \geq t_2 - y \wedge t \leq t_2 + y \wedge 0 \leq t_2 - t_1 \wedge t_2 - t_1 \leq x \\ \quad \supset |s + x - t| \leq 2 * y + x \end{array}$$

okay_Readvars_shift_stepb: **Lemma**

$$\begin{array}{l} s \geq t_1 - y \wedge s \leq t_1 + y \\ \quad \wedge\, t \geq t_2 - y \wedge t \leq t_2 + y \wedge 0 \leq t_2 - t_1 \wedge t_2 - t_1 \leq x \\ \quad \supset |s - t| \leq 2 * y + x \end{array}$$

okay_Readvars_shift_step1: **Lemma**

$$\begin{array}{l} |s - t_1| \leq y \wedge |t - t_2| \leq y \wedge 0 \leq t_2 - t_1 \wedge t_2 - t_1 \leq x \\ \quad \supset |s + x - t| \leq 2 * y + x \end{array}$$

okay_Readvars_shift_step2: **Lemma**

$$\begin{array}{l} |s - t_1| \leq y \wedge |t - t_2| \leq y \wedge 0 \leq t_2 - t_1 \wedge t_2 - t_1 \leq x \\ \quad \supset |s - t| \leq 2 * y + x \end{array}$$

okay_Readvars_shift11: **Lemma**

$$\begin{array}{l} \text{correct}(p, t_p^{i+1}) \\ \quad \wedge\, \text{correct}(q, t_p^{i+1}) \wedge \text{correct}(p_1, t_p^{i+1}) \wedge t_p^{i+1} \geq t_q^{i+1} \\ \quad \supset \Theta_q^{i+1} p_1) + (PC_q(t_p^{i+1}) - PC_q(t_q^{i+1})) - \Theta_p^{i+1} p_1) \\ \quad\quad \leq 2 * \Lambda + 2 * \beta \star \rho \end{array}$$

okay_Readvars_shift12: **Lemma**

$$\begin{array}{l} \text{correct}(p, t_p^{i+1}) \\ \quad \wedge\, \text{correct}(q, t_p^{i+1}) \wedge \text{correct}(p_1, t_p^{i+1}) \wedge t_p^{i+1} \geq t_q^{i+1} \\ \quad \supset \Theta_p^{i+1} p_1) - (\Theta_q^{i+1} p_1) + (PC_q(t_p^{i+1}) - PC_q(t_q^{i+1}))) \\ \quad\quad \leq 2 * \Lambda + 2 * \beta \star \rho \end{array}$$

okay_Readvars_shift1: **Lemma**

$$\begin{array}{l} \text{correct}(p, t_p^{i+1}) \\ \quad \wedge\, \text{correct}(q, t_p^{i+1}) \wedge \text{correct}(p_1, t_p^{i+1}) \wedge t_p^{i+1} \geq t_q^{i+1} \\ \quad \supset |\Theta_p^{i+1} p_1) - (\Theta_q^{i+1} p_1) + (PC_q(t_p^{i+1}) - PC_q(t_q^{i+1})))| \\ \quad\quad \leq 2 * \Lambda + 2 * \beta \star \rho \end{array}$$

okay_Readvars_shift2: **Lemma**

$$\begin{array}{l} \text{correct}(p, t_p^{i+1}) \\ \quad \wedge\, \text{correct}(q, t_p^{i+1}) \wedge t \geq t_p^{i+1} \wedge t_p^{i+1} \geq t_q^{i+1} \\ \quad \supset \text{okay\_Readvars}(\Theta_p^{i+1}, \Theta_q^{i+1}, 2 * \Lambda + 2 * \beta \star \rho, t) \end{array}$$

okay_Readvars_shift: **Lemma**

$$\begin{array}{l} t \geq t_p^{i+1} \wedge \text{correct}(p, t) \wedge \text{correct}(q, t) \wedge t_p^{i+1} \geq t_q^{i+1} \\ \quad \supset \text{okay\_Readvars}(\Theta_p^{i+1}, \\ \quad\quad\quad (\lambda\, p_1 \rightarrow \text{time}: \\ \quad\quad\quad\quad \Theta_q^{i+1} p_1) + (PC_q(t_p^{i+1}) - PC_q(t_q^{i+1}))), \\ \quad\quad\quad 2 * \Lambda + 2 * \beta \star \rho, \\ \quad\quad\quad t) \end{array}$$

Proof

maxsync_correct_pr: **Prove** maxsync_correct **from** $(\star 1 \Uparrow \star 2)[\star 3]$

minsync_correct_pr: **Prove** minsync_correct **from** minsync

minsync_maxsync_pr: **Prove** minsync_maxsync **from** minsync, $(\star 1 \Uparrow \star 2)[\star 3]$

basics: **Module**

Using clockassumptions, arith

Exporting all with clockassumptions

Theory

$p, q, p_1, p_2, q_1, q_2, l, m, n$: **Var** process
i, j, k: **Var** event
x, y, z: **Var** number
r, s, t, t_1, t_2: **Var** time
$X, Y, Z, R, S, T, T_1, T_2$: **Var** Clocktime
γ, θ: **Var** function[process $\rightarrow$ time]
$(\star 1 \Uparrow \star 2)[\star 3]$: **Definition** function[process, process, event $\rightarrow$ process] =
$(\lambda\, p, q, i: (\textbf{ if } t_p^i \geq t_q^i \textbf{ then } p \textbf{ else } q \textbf{ end if}))$

maxsync_correct: **Lemma** $\mathrm{correct}(p, s) \wedge \mathrm{correct}(q, s) \supset \mathrm{correct}((p \Uparrow q)[i], s)$

minsync: **Definition** function[process, process, event $\rightarrow$ process] =
$(\lambda\, p, q, i: (\textbf{ if } t_p^i \geq t_q^i \textbf{ then } q \textbf{ else } p \textbf{ end if}))$

minsync_correct: **Lemma** $\mathrm{correct}(p, s) \wedge \mathrm{correct}(q, s) \supset \mathrm{correct}((p \Downarrow q)[i], s)$

minsync_maxsync: **Lemma** $t^i_{(p \Downarrow q)[i]} \leq t^i_{(p \Uparrow q)[i]}$

$t^{\star 3}_{\star 1, \star 2}$: **Definition** function[process, process, event $\rightarrow$ time] =
$(\lambda\, p, q, i: t^i_{(p \Uparrow q)[i]})$

lemma_1: **Lemma** $\mathrm{correct}(p, t_p^i) \wedge \mathrm{correct}(q, t_q^{i+1}) \wedge \beta \leq r_{min}$
$\supset t_p^i \leq t_q^{i+1}$

lemma_1_1: **Lemma** $\mathrm{correct}(p, t_q^{i+1}) \wedge \mathrm{correct}(q, t_q^{i+1}) \wedge \beta \leq r_{min}$
$\supset t_p^i \leq t_q^{i+1}$

lemma_1_2: **Lemma** $\mathrm{correct}(p, t_p^{i+1}) \wedge \mathrm{correct}(q, t_q^i)$
$\supset t_p^{i+1} \leq t_q^i + r_{max} + \beta$

lemma_2_0: **Lemma** $\mathrm{correct}(p, 0) \wedge \mathrm{correct}(q, 0) \supset |IC_p^0(0) - IC_q^0(0)| \leq \mu$

lemma_2_1: **Lemma** $\mathrm{correct}(q, t_q^{i+1})$
$\supset IC_q^{i+1}(t_q^{i+1}) = cfn(q, \Theta_q^{i+1})$

lemma_2_2a: **Lemma**
$\mathrm{correct}(q, s) \wedge s \geq t \supset IC_q^i(s) \leq IC_q^i(t) + (s - t) \star (1 + \rho)$

lemma_2_2b: **Lemma**
$\mathrm{correct}(q, s) \wedge s \geq t \supset IC_q^i(s) \geq IC_q^i(t) + (s - t) \star (1 - \rho)$

abs_shift: **Lemma** $|r - s| \leq x$
$\wedge\, t_1 \leq r + y + z \wedge t_1 \geq r + y - z \wedge t_2 \leq s + y + z \wedge t_2 \geq s + y - z$
$\supset |t_1 - t_2| \leq x + 2 * z$

ReadClock_bnd1: **Lemma**
$\mathrm{correct}(p, t_p^{i+1}) \wedge \mathrm{correct}(q, t_p^{i+1})$
$\supset \Theta_p^{i+1} q) \leq IC_q^i(t_p^{i+1}) + \Lambda$

ReadClock_bnd2: **Lemma**
$\mathrm{correct}(p, t_p^{i+1}) \wedge \mathrm{correct}(q, t_p^{i+1})$
$\supset \Theta_p^{i+1} q) \geq IC_q^i(t_p^{i+1}) - \Lambda$

ReadClock_bnd11: **Lemma**
$\mathrm{correct}(p, t_p^{i+1}) \wedge \mathrm{correct}(q, t_p^{i+1}) \wedge \mathrm{correct}(p_1, t_{p_1}^i) \wedge \beta \leq r_{min}$
$\supset \Theta_p^{i+1} q) \leq IC_q^i(t_{p_1}^i) + (t_p^{i+1} - t_{p_1}^i) + (r_{max} + \beta) \star \rho + \Lambda$

ReadClock_bnd12: **Lemma**
$\mathrm{correct}(p, t_p^{i+1}) \wedge \mathrm{correct}(q, t_p^{i+1}) \wedge \mathrm{correct}(p_1, t_{p_1}^i) \wedge \beta \leq r_{min}$
$\supset \Theta_p^{i+1} q) \geq IC_q^i(t_{p_1}^i) + (t_p^{i+1} - t_{p_1}^i) - (r_{max} + \beta) \star \rho - \Lambda$

ReadClock_bnd: **Lemma**
$\mathrm{correct}(p, t_p^{i+1})$
$\wedge\, \mathrm{correct}(q, t_p^{i+1})$
$\wedge\, \mathrm{correct}(q_1, t_p^{i+1})$
$\wedge\, |IC_q^i(t_{q,q_1}^i) - IC_{q_1}^i(t_{q,q_1}^i)| \leq X \wedge \beta \leq r_{min}$
$\supset |\Theta_p^{i+1} q) - \Theta_p^{i+1} q_1)| \leq X + 2 * ((r_{max} + \beta) \star \rho + \Lambda)$

precision_enhancement: **Lemma**

$\text{okay\_Reading}(\gamma, Y, t_p^{i+1})$
$\quad \wedge \text{okay\_Reading}(\theta, Y, t_p^{i+1})$
$\quad\quad \wedge \text{okay\_Readvars}(\gamma, \theta, X, t_p^{i+1})$
$\quad\quad\quad \wedge \text{correct}(p, t_p^{i+1}) \wedge \text{correct}(q, t_p^{i+1})$
$\quad \supset |\mathit{cfn}(p,\gamma) - \mathit{cfn}(q,\theta)| \le \pi(X, Y)$

okay_Reading_defn_lr: **Lemma**

$\text{okay\_Reading}(\gamma, Y, t)$
$\quad \supset (\forall\, p_1, q_1 : \text{correct}(p_1, t) \wedge \text{correct}(q_1, t) \supset |\gamma(p_1) - \gamma(q_1)| \le Y)$

okay_Reading_defn_rl: **Lemma**

$(\forall\, p_1, q_1 : \text{correct}(p_1, t) \wedge \text{correct}(q_1, t) \supset |\gamma(p_1) - \gamma(q_1)| \le Y)$
$\quad \supset \text{okay\_Reading}(\gamma, Y, t)$

okay_Readvars_defn_lr: **Lemma**

$\text{okay\_Readvars}(\gamma, \theta, X, t) \supset (\forall\, p_3 : \text{correct}(p_3, t) \supset |\gamma(p_3) - \theta(p_3)| \le X)$

okay_Readvars_defn_rl: **Lemma**

$(\forall\, p_3 : \text{correct}(p_3, t) \supset |\gamma(p_3) - \theta(p_3)| \le X) \supset \text{okay\_Readvars}(\gamma, \theta, X, t)$

accuracy_preservation_ax: **Axiom**

$\text{okay\_Readpred}(\gamma, X, \text{ppred})$
$\quad \wedge \text{count}(\text{ppred}, N) \ge N - \text{maxfaults} \wedge \text{ppred}(p) \wedge \text{ppred}(q)$
$\quad \supset |\mathit{cfn}(p,\gamma) - \gamma(q)| \le \alpha(X)$

Proof

okay_Reading_defn_rl_pr: **Prove**
okay_Reading_defn_rl $\{p_1 \leftarrow p_1@\text{P1S},\ q_1 \leftarrow q_1@\text{P1S}\}$ **from** okay_Reading

okay_Reading_defn_lr_pr: **Prove** okay_Reading_defn_lr **from**
okay_Reading $\{p_1 \leftarrow p_1@\text{CS},\ q_1 \leftarrow q_1@\text{CS}\}$

okay_Readvars_defn_rl_pr: **Prove** okay_Readvars_defn_rl $\{p_3 \leftarrow p_3@\text{P1S}\}$ **from**
okay_Readvars

okay_Readvars_defn_lr_pr: **Prove** okay_Readvars_defn_lr **from**
okay_Readvars $\{p_3 \leftarrow p_3@\text{CS}\}$

precision_enhancement_pr: **Prove** precision_enhancement **from**
precision_enhancement_ax $\{\text{ppred} \leftarrow (\lambda\, q : \text{correct}(q, t_p^{i+1}))\}$,
okay_Readpred_Reading $\{t \leftarrow t_p^{i+1}\}$,
okay_Readpred_Reading $\{t \leftarrow t_p^{i+1},\ \gamma \leftarrow \theta\}$,
okay_pairs_Readvars $\{t \leftarrow t_p^{i+1}\}$,
correct_count $\{t \leftarrow t_p^{i+1}\}$

okay_Readpred_Reading_pr: **Prove** okay_Readpred_Reading **from**
okay_Readpred $\{\text{ppred} \leftarrow (\lambda\, p : \text{correct}(p, t))\}$,
okay_Reading $\{p_1 \leftarrow l@P1S,\ q_1 \leftarrow m@P1S\}$

okay_pairs_Readvars_pr: **Prove** okay_pairs_Readvars **from**
okay_pairs $\{\text{ppred} \leftarrow (\lambda\, p : \text{correct}(p, t))\}$, okay_Readvars $\{p_3 \leftarrow p_3@\text{P1S}\}$

rts_0_proof: **Prove** rts_0 **from** rts0 $\{t \leftarrow t_p^{i+1}\}$

rts_1_proof: **Prove** rts_1 **from** rts1 $\{t \leftarrow t_p^{i+1}\}$

End clockassumptions

beta_0: **Axiom** $\beta \geq 0$

lamb_0: **Axiom** $\Lambda \geq 0$

init: **Axiom** $\text{correct}(p, 0) \supset PC_p(0) \geq 0 \wedge PC_p(0) \leq \mu$

correct_closed: **Axiom** $s \geq t \wedge \text{correct}(p, s) \supset \text{correct}(p, t)$

rate_1: **Axiom** $\text{correct}(p, s) \wedge s \geq t \supset PC_p(s) - PC_p(t) \leq (s - t) \star (1 + \rho)$

rate_2: **Axiom** $\text{correct}(p, s) \wedge s \geq t \supset PC_p(s) - PC_p(t) \geq (s - t) \star (1 - \rho)$

rts0: **Axiom** $\text{correct}(p, t) \wedge t \leq t_p^{i+1} \supset t - t_p^i \leq r_{max}$

rts1: **Axiom** $\text{correct}(p, t) \wedge t \geq t_p^{i+1} \supset t - t_p^i \geq r_{min}$

rts_0: **Lemma** $\text{correct}(p, t_p^{i+1}) \supset t_p^{i+1} - t_p^i \leq r_{max}$

rts_1: **Lemma** $\text{correct}(p, t_p^{i+1}) \supset t_p^{i+1} - t_p^i \geq r_{min}$

rts2: **Axiom** $\text{correct}(p, t) \wedge t \geq t_q^i + \beta \wedge \text{correct}(q, t) \supset t \geq t_p^i$

rts_2: **Axiom** $\text{correct}(p, t_p^i) \wedge \text{correct}(q, t_q^i) \supset t_p^i - t_q^i \leq \beta$

synctime_0: **Axiom** $t_p^0 = 0$

VClock_defn: **Axiom**
$\text{correct}(p, t) \wedge t \geq t_p^i \wedge t < t_p^{i+1} \supset VC_p(t) = IC_p^i(t)$

Adj: function[process, event → Clocktime] =
$(\lambda\, p, i\colon (\textbf{ if } i > 0 \textbf{ then } \mathit{cfn}(p, \Theta_p^i) - PC_p(t_p^i) \textbf{ else } 0 \textbf{ end if}))$

IClock_defn: **Axiom** $\text{correct}(p, t) \supset IC_p^i(t) = PC_p(t) + \text{Adj}(p, i)$

Readerror: **Axiom** $\text{correct}(p, t_p^{i+1}) \wedge \text{correct}(q, t_p^{i+1})$
$\supset |\Theta_p^{i+1} q) - IC_q^i(t_p^{i+1})| \leq \Lambda$

translation_invariance: **Axiom**
$X \geq 0 \supset \mathit{cfn}(p, (\lambda\, p_1 \to \text{Clocktime}\colon \gamma(p_1) + X)) = \mathit{cfn}(p, \gamma) + X$

ppred: **Var** function[process → bool]
maxfaults: process
okay_Readpred: function[function[process → Clocktime], Clocktime,
function[process → bool] → bool] =
$(\lambda\, \gamma, Y, \text{ppred}\colon (\forall\, l, m\colon \text{ppred}(l) \wedge \text{ppred}(m) \supset |\gamma(l) - \gamma(m)| \leq Y))$
okay_pairs: function[function[process → Clocktime],
function[process → Clocktime], Clocktime,
function[process → bool] → bool] =
$(\lambda\, \gamma, \theta, X, \text{ppred}\colon (\forall\, p_3\colon \text{ppred}(p_3) \supset |\gamma(p_3) - \theta(p_3)| \leq X))$
N: process

N_0: **Axiom** $N > 0$

N_maxfaults: **Axiom** $\text{maxfaults} \leq N$

precision_enhancement_ax: **Axiom**
$\text{count}(\text{ppred}, N) \geq N - \text{maxfaults}$
$\wedge\, \text{okay\_Readpred}(\gamma, Y, \text{ppred})$
$\wedge\, \text{okay\_Readpred}(\theta, Y, \text{ppred})$
$\wedge\, \text{okay\_pairs}(\gamma, \theta, X, \text{ppred}) \wedge \text{ppred}(p) \wedge \text{ppred}(q)$
$\supset |\mathit{cfn}(p, \gamma) - \mathit{cfn}(q, \theta)| \leq \pi(X, Y)$

correct_count: **Axiom** $\text{count}((\lambda\, p\colon \text{correct}(p, t)), N) \geq N - \text{maxfaults}$

okay_Reading: function[function[process → Clocktime], Clocktime, time
→ bool] =
$(\lambda\, \gamma, Y, t\colon (\forall\, p_1, q_1\colon$
$\text{correct}(p_1, t) \wedge \text{correct}(q_1, t) \supset |\gamma(p_1) - \gamma(q_1)| \leq Y))$
okay_Readvars: function[function[process → Clocktime],
function[process → Clocktime], Clocktime, Clocktime
→ bool] =
$(\lambda\, \gamma, \theta, X, t\colon (\forall\, p_3\colon \text{correct}(p_3, t) \supset |\gamma(p_3) - \theta(p_3)| \leq X))$

okay_Readpred_Reading: **Lemma**
$\text{okay\_Reading}(\gamma, Y, t) \supset \text{okay\_Readpred}(\gamma, Y, (\lambda\, p\colon \text{correct}(p, t)))$

okay_pairs_Readvars: **Lemma**
$\text{okay\_Readvars}(\gamma, \theta, X, t) \supset \text{okay\_pairs}(\gamma, \theta, X, (\lambda\, p\colon \text{correct}(p, t)))$

countmod_tcc: **Module**

Using countmod

Exporting all with countmod

Theory

i: **Var** naturalnumber
ppred: **Var** function[naturalnumber $\rightarrow$ boolean]

count_TCC1: **Formula** $(i > 0) \supset (i - 1 \geq 0)$

count_TCC2: **Formula** $(\text{ppred}(i-1)) \wedge (i > 0) \supset (i - 1 \geq 0)$

count_TCC3: **Formula** $(\neg(\text{ppred}(i-1))) \wedge (i > 0) \supset (i - 1 \geq 0)$

count_TCC4: **Formula**
$(\text{ppred}(i-1)) \wedge (i > 0) \supset \text{countsize}(\text{ppred}, i) > \text{countsize}(\text{ppred}, i-1)$

count_TCC5: **Formula**
$(\neg(\text{ppred}(i-1))) \wedge (i > 0) \supset \text{countsize}(\text{ppred}, i) > \text{countsize}(\text{ppred}, i-1)$

Proof

count_TCC1_PROOF: **Prove** count_TCC1

count_TCC2_PROOF: **Prove** count_TCC2

count_TCC3_PROOF: **Prove** count_TCC3

count_TCC4_PROOF: **Prove** count_TCC4

count_TCC5_PROOF: **Prove** count_TCC5

End countmod_tcc

clockassumptions: **Module**

Using arith, countmod

Exporting all with countmod, arith

Theory

process: **Type is** nat
event: **Type is** nat
time: **Type is** number
Clocktime: **Type is** number
$l, m, n, p, q, p_1, p_2, q_1, q_2, p_3, q_3$: **Var** process
i, j, k: **Var** event
x, y, z, r, s, t: **Var** time
X, Y, Z, R, S, T: **Var** Clocktime
γ, θ: **Var** function[process $\rightarrow$ Clocktime]
$\delta, \mu, \rho, r_{min}, r_{max}, \beta, \Lambda$: number
$PC_{\star 1}(\star 2), VC_{\star 1}(\star 2)$: function[process, time $\rightarrow$ Clocktime]
$t_{\star 1}^{\star 2}$: function[process, event $\rightarrow$ time]
$\Theta_{\star 1}^{\star 2}$: function[process, event $\rightarrow$ function[process $\rightarrow$ Clocktime]]
$IC_{\star 1}^{\star 2}(\star 3)$: function[process, event, time $\rightarrow$ Clocktime]
correct: function[process, time $\rightarrow$ bool]
cfn: function[process, function[process $\rightarrow$ Clocktime] $\rightarrow$ Clocktime]
π: function[Clocktime, Clocktime $\rightarrow$ Clocktime]
α: function[Clocktime $\rightarrow$ Clocktime]

delta_0: **Axiom** $\delta \geq 0$

mu_0: **Axiom** $\mu \geq 0$

rho_0: **Axiom** $\rho \geq 0$

rho_1: **Axiom** $\rho < 1$

rmin_0: **Axiom** $r_{min} > 0$

rmax_0: **Axiom** $r_{max} > 0$

arith: **Module**

Using multiplication, division, absmod

Exporting all with multiplication, division, absmod

End arith

countmod: **Module**

Exporting all

Theory

$l, m, n, p, q, p_1, p_2, q_1, q_2, p_3, q_3$: **Var** nat
i, j, k: **Var** nat
x, y, z, r, s, t: **Var** number
X, Y, Z: **Var** number
ppred, ppred1, ppred2: **Var** function[nat → bool]
fun, fun1, fun2: **Var** function[nat → number]
countsize: function[function[nat → bool], nat → nat] = (λ ppred, i: i)
count: **Recursive** function[function[nat → bool], nat → nat] =
(λ ppred, i: (**if** $i > 0$
then (**if** ppred($i - 1$)
then $1 +$ (count(ppred, $i - 1$))
else count(ppred, $i - 1$)
end if)
else 0
end if)) **by** countsize

End countmod

division_tcc: **Module**

End division_tcc

Using division

Exporting all with division

Theory

x: **Var** number
y: **Var** number
z: **Var** number

mult_div_1_TCC1: **Formula** $(z \neq 0) \supset (z \neq 0)$

mult_div_TCC1: **Formula** $(y \neq 0) \supset (y \neq 0)$

div_cancel_TCC1: **Formula** $(x \neq 0) \supset (x \neq 0)$

ceil_mult_div_TCC1: **Formula** $(y > 0) \supset (y \neq 0)$

div_nonnegative_TCC1: **Formula** $(x \geq 0 \wedge y > 0) \supset (y \neq 0)$

div_ineq_TCC1: **Formula** $(z > 0 \wedge x \leq y) \supset (z \neq 0)$

div_minus_1_TCC1: **Formula** $(y > 0 \wedge x < 0) \supset (y \neq 0)$

Proof

mult_div_1_TCC1_PROOF: **Prove** mult_div_1_TCC1

mult_div_TCC1_PROOF: **Prove** mult_div_TCC1

div_cancel_TCC1_PROOF: **Prove** div_cancel_TCC1

ceil_mult_div_TCC1_PROOF: **Prove** ceil_mult_div_TCC1

div_nonnegative_TCC1_PROOF: **Prove** div_nonnegative_TCC1

div_ineq_TCC1_PROOF: **Prove** div_ineq_TCC1

div_minus_1_TCC1_PROOF: **Prove** div_minus_1_TCC1

Proof

div_nonnegative_pr: **Prove** div_nonnegative **from**
mult_non_neg $\{x \leftarrow (\textbf{ if } y \neq 0 \textbf{ then } (x/y) \textbf{ else } 0 \textbf{ end if})\}$, mult_div

div_distrib_pr: **Prove** div_distrib **from**
mult_div_1 $\{x \leftarrow x + y,\ y \leftarrow 1,\ z \leftarrow z\}$,
mult_rident $\{x \leftarrow x + y\}$,
mult_div_1 $\{x \leftarrow x,\ y \leftarrow 1,\ z \leftarrow z\}$,
mult_rident,
mult_div_1 $\{x \leftarrow y,\ y \leftarrow 1,\ z \leftarrow z\}$,
mult_rident $\{x \leftarrow y\}$,
distrib $\{z \leftarrow (\textbf{ if } z \neq 0 \textbf{ then } (1/z) \textbf{ else } 0 \textbf{ end if})\}$

div_cancel_pr: **Prove** div_cancel **from**
mult_div_2 $\{z \leftarrow x\}$, mult_div_3 $\{z \leftarrow x\}$, mult_lident $\{x \leftarrow y\}$

mult_div_pr: **Prove** mult_div **from**
mult_div_2 $\{z \leftarrow y\}$, mult_div_1 $\{z \leftarrow y\}$, mult_div_3 $\{z \leftarrow y\}$, mult_rident

abs_div_pr: **Prove** abs_div **from**
$|\star 1|$ $\{x \leftarrow (\textbf{ if } y \neq 0 \textbf{ then } (x/y) \textbf{ else } 0 \textbf{ end if})\}$,
$|\star 1|$,
div_nonnegative,
div_minus_1,
mult_minus

mult_minus_pr: **Prove** mult_minus **from**
mult_div_1 $\{x \leftarrow -1,\ y \leftarrow x,\ z \leftarrow y\}$,
$\star 1 \star\star 2$ $\{x \leftarrow -1,\ y \leftarrow x\}$,
$\star 1 \star\star 2$ $\{x \leftarrow -1,\ y \leftarrow (\textbf{ if } y \neq 0 \textbf{ then } (x/y) \textbf{ else } 1 \textbf{ end if})\}$

div_minus_1_pr: **Prove** div_minus_1 **from**
mult_div,
pos_product $\{x \leftarrow (\textbf{ if } y \neq 0 \textbf{ then } (x/y) \textbf{ else } 0 \textbf{ end if}),\ y \leftarrow y\}$

div_minus_distrib_pr: **Prove** div_minus_distrib **from**
div_distrib $\{y \leftarrow -y\}$, mult_minus $\{x \leftarrow y,\ y \leftarrow z\}$

div_ineq_pr: **Prove** div_ineq **from**
mult_div $\{y \leftarrow z\}$,
mult_div $\{x \leftarrow y,\ y \leftarrow z\}$,
mult_gt
$\{x \leftarrow (\textbf{ if } z \neq 0 \textbf{ then } (x/z) \textbf{ else } 0 \textbf{ end if}),$
$y \leftarrow (\textbf{ if } z \neq 0 \textbf{ then } (y/z) \textbf{ else } 0 \textbf{ end if})\}$

ceil_plus_mult_div_proof: **Prove** ceil_plus_mult_div **from**
ceil_mult_div,
distrib
$\{x \leftarrow \lceil(\textbf{ if } y \neq 0 \textbf{ then } (x/y) \textbf{ else } 0 \textbf{ end if})\rceil,$
$y \leftarrow 1,$
$z \leftarrow y\}$,
mult_lident $\{x \leftarrow y\}$

ceil_mult_div_proof: **Prove** ceil_mult_div **from**
mult_div,
mult_leq
$\{x \leftarrow \lceil(\textbf{ if } y \neq 0 \textbf{ then } (x/y) \textbf{ else } 0 \textbf{ end if})\rceil,$
$y \leftarrow (\textbf{ if } y \neq 0 \textbf{ then } (x/y) \textbf{ else } 0 \textbf{ end if}),$
$z \leftarrow y\}$,
ceil_defn $\{x \leftarrow (\textbf{ if } y \neq 0 \textbf{ then } (x/y) \textbf{ else } 0 \textbf{ end if})\}$

End division

abs_2_bnd_proof: **Prove** abs_2_bnd **from** $|\star 1|$ $\{x \leftarrow (x - y)\}$

End absmod

division: **Module**

Using multiplication, absmod

Exporting all

Theory

$x, y, z, x_1, y_1, z_1, x_2, y_2, z_2$: **Var** number
$\lceil \star 1 \rceil$: function[number → int]

ceil_defn: **Axiom** $\lceil x \rceil \geq x \wedge \lceil x \rceil - 1 < x$

mult_div_1: **Axiom** $z \neq 0 \supset x \star y / z = x \star (y/z)$

mult_div_2: **Axiom** $z \neq 0 \supset x \star y / z = (x/z) \star y$

mult_div_3: **Axiom** $z \neq 0 \supset (z/z) = 1$

mult_div: **Lemma** $y \neq 0 \supset (x/y) \star y = x$

div_cancel: **Lemma** $x \neq 0 \supset x \star y / x = y$

div_distrib: **Lemma** $z \neq 0 \supset ((x + y)/z) = (x/z) + (y/z)$

ceil_mult_div: **Lemma** $y > 0 \supset \lceil x/y \rceil \star y \geq x$

ceil_plus_mult_div: **Lemma** $y > 0 \supset \lceil x/y \rceil + 1 \star y > x$

div_nonnegative: **Lemma** $x \geq 0 \wedge y > 0 \supset (x/y) \geq 0$

div_minus_distrib: **Lemma** $z \neq 0 \supset (x - y)/z = (x/z) - (y/z)$

div_ineq: **Lemma** $z > 0 \wedge x \leq y \supset (x/z) \leq (y/z)$

abs_div: **Lemma** $y > 0 \supset |x/y| = |x|/y$

mult_minus: **Lemma** $y \neq 0 \supset -(x/y) = (-x/y)$

div_minus_1: **Lemma** $y > 0 \wedge x < 0 \supset (x/y) < 0$

absmod: **Module**

Using multiplication

Exporting all

Theory

$x, y, z, x_1, y_1, z_1, x_2, y_2, z_2$: **Var** number
$|\star 1|$: **Definition** function[number → number] =
$(\lambda\, x: (\textbf{ if } x < 0 \textbf{ then } -x \textbf{ else } x \textbf{ end if}))$

abs_main: **Lemma** $|x| < z \supset (x < z \vee -x < z)$

abs_leq_0: **Lemma** $|x - y| \leq z \supset (x - y) \leq z$

abs_diff: **Lemma** $|x - y| < z \supset ((x - y) < z \vee (y - x) < z)$

abs_leq: **Lemma** $|x| \leq z \supset (x \leq z \vee -x \leq z)$

abs_bnd: **Lemma** $0 \leq z \wedge 0 \leq x \wedge x \leq z \wedge 0 \leq y \wedge y \leq z \supset |x - y| \leq z$

abs_1_bnd: **Lemma** $|x - y| \leq z \supset x \leq y + z$

abs_2_bnd: **Lemma** $|x - y| \leq z \supset x \geq y - z$

abs_3_bnd: **Lemma** $x \leq y + z \wedge x \geq y - z \supset |x - y| \leq z$

abs_drift: **Lemma** $|x - y| \leq z \wedge |x_1 - x| \leq z_1 \supset |x_1 - y| \leq z + z_1$

abs_com: **Lemma** $|x - y| = |y - x|$

abs_drift_2: **Lemma**
$|x - y| \leq z \wedge |x_1 - x| \leq z_1 \wedge |y_1 - y| \leq z_2 \supset |x_1 - y_1| \leq z + z_1 + z_2$

abs_geq: **Lemma** $x \geq y \wedge y \geq 0 \supset |x| \geq |y|$

abs_ge0: **Lemma** $x \geq 0 \supset |x| = x$

abs_plus: **Lemma** $|x + y| \leq |x| + |y|$

abs_diff_3: **Lemma** $x - y \leq z \wedge y - x \leq z \supset |x - y| \leq z$

Proof

abs_plus_pr: **Prove** abs_plus **from** $|\star 1|$ $\{x \leftarrow x + y\}$, $|\star 1|$, $|\star 1|$ $\{x \leftarrow y\}$

abs_diff_3_pr: **Prove** abs_diff_3 **from** $|\star 1|$ $\{x \leftarrow x - y\}$

abs_ge0_proof: **Prove** abs_ge0 **from** $|\star 1|$

abs_geq_proof: **Prove** abs_geq **from** $|\star 1|$, $|\star 1|$ $\{x \leftarrow y\}$

abs_drift_2_proof: **Prove** abs_drift_2 **from**
abs_drift,
abs_drift $\{x \leftarrow y,\ y \leftarrow y_1,\ z \leftarrow z_2,\ z_1 \leftarrow z + z_1\}$,
abs_com $\{x \leftarrow y_1\}$

abs_com_proof: **Prove** abs_com **from** $|\star 1|$ $\{x \leftarrow (x - y)\}$, $|\star 1|$ $\{x \leftarrow (y - x)\}$

abs_drift_proof: **Prove** abs_drift **from**
abs_1_bnd,
abs_1_bnd $\{x \leftarrow x_1,\ y \leftarrow x,\ z \leftarrow z_1\}$,
abs_2_bnd,
abs_2_bnd $\{x \leftarrow x_1,\ y \leftarrow x,\ z \leftarrow z_1\}$,
abs_3_bnd $\{x \leftarrow x_1,\ z \leftarrow z + z_1\}$

abs_3_bnd_proof: **Prove** abs_3_bnd **from** $|\star 1|$ $\{x \leftarrow (x - y)\}$

abs_main_proof: **Prove** abs_main **from** $|\star 1|$

abs_leq_0_proof: **Prove** abs_leq_0 **from** $|\star 1|$ $\{x \leftarrow x - y\}$

abs_diff_proof: **Prove** abs_diff **from** $|\star 1|$ $\{x \leftarrow (x - y)\}$

abs_leq_proof: **Prove** abs_leq **from** $|\star 1|$

abs_bnd_proof: **Prove** abs_bnd **from** $|\star 1|$ $\{x \leftarrow (x - y)\}$

abs_1_bnd_proof: **Prove** abs_1_bnd **from** $|\star 1|$ $\{x \leftarrow (x - y)\}$

multiplication: **Module**

Exporting all

Theory

$x, y, z, x_1, y_1, z_1, x_2, y_2, z_2$: **Var** number
$\star 1 \star \star 2$: function[number, number $\rightarrow$ number] = $(\lambda\, x, y: (x * y))$

mult_ldistrib: **Lemma** $x \star (y + z) = x \star y + x \star z$

mult_ldistrib_minus: **Lemma** $x \star (y - z) = x \star y - x \star z$

mult_rident: **Lemma** $x \star 1 = x$

mult_lident: **Lemma** $1 \star x = x$

distrib: **Lemma** $(x + y) \star z = x \star z + y \star z$

distrib_minus: **Lemma** $(x - y) \star z = x \star z - y \star z$

mult_non_neg: **Axiom** $((x \geq 0 \wedge y \geq 0) \vee (x \leq 0 \wedge y \leq 0)) \Leftrightarrow x \star y \geq 0$

mult_pos: **Axiom** $((x > 0 \wedge y > 0) \vee (x < 0 \wedge y < 0)) \Leftrightarrow x \star y > 0$

mult_com: **Lemma** $x \star y = y \star x$

pos_product: **Lemma** $x \geq 0 \wedge y \geq 0 \supset x \star y \geq 0$

mult_leq: **Lemma** $z \geq 0 \wedge x \geq y \supset x \star z \geq y \star z$

mult_leq_2: **Lemma** $z \geq 0 \wedge x \geq y \supset z \star x \geq z \star y$

mult_l0: **Axiom** $0 \star x = 0$

mult_gt: **Lemma** $z > 0 \wedge x > y \supset x \star z > y \star z$

Proof

mult_gt_pr: **Prove** mult_gt **from**
mult_pos $\{x \leftarrow x - y,\ y \leftarrow z\}$, distrib_minus

distrib_minus_pr: **Prove** distrib_minus **from**
mult_ldistrib_minus $\{x \leftarrow z,\ y \leftarrow x,\ z \leftarrow y\}$,
mult_com $\{x \leftarrow x - y,\ y \leftarrow z\}$,
mult_com $\{y \leftarrow z\}$,
mult_com $\{x \leftarrow y,\ y \leftarrow z\}$

mult_leq_2_pr: **Prove** mult_leq_2 **from**
mult_ldistrib_minus $\{x \leftarrow z,\ y \leftarrow x,\ z \leftarrow y\}$,
mult_non_neg $\{x \leftarrow z,\ y \leftarrow x - y\}$

mult_leq_pr: **Prove** mult_leq **from**
distrib_minus, mult_non_neg $\{x \leftarrow x - y,\ y \leftarrow z\}$

mult_com_pr: **Prove** mult_com **from** $\star 1 \star \star 2$, $\star 1 \star \star 2$ $\{x \leftarrow y,\ y \leftarrow x\}$

pos_product_pr: **Prove** pos_product **from** mult_non_neg

mult_rident_proof: **Prove** mult_rident **from** $\star 1 \star \star 2$ $\{y \leftarrow 1\}$

mult_lident_proof: **Prove** mult_lident **from** $\star 1 \star \star 2$ $\{x \leftarrow 1,\ y \leftarrow x\}$

distrib_proof: **Prove** distrib **from**
$\star 1 \star \star 2$ $\{x \leftarrow x + y,\ y \leftarrow z\}$,
$\star 1 \star \star 2$ $\{y \leftarrow z\}$,
$\star 1 \star \star 2$ $\{x \leftarrow y,\ y \leftarrow z\}$

mult_ldistrib_proof: **Prove** mult_ldistrib **from**
$\star 1 \star \star 2$ $\{y \leftarrow y + z,\ x \leftarrow x\}$, $\star 1 \star \star 2$, $\star 1 \star \star 2$ $\{y \leftarrow z\}$

mult_ldistrib_minus_proof: **Prove** mult_ldistrib_minus **from**
$\star 1 \star \star 2$ $\{y \leftarrow y - z,\ x \leftarrow x\}$, $\star 1 \star \star 2$, $\star 1 \star \star 2$ $\{y \leftarrow z\}$

End multiplication

Appendix B

The Complete EHDM Proof

Note that the modules ending with `_tcc` are automatically generated during type checking. The proofs declared in these modules may not succeed, but all the automatically generated theorems have been proved as illustrated by the completeness of the proof chain analyses in Appendix C.

```
Module multiplication:   11 successful proofs,  0 failures,  0 errors
Module arith:            no proofs
Module top:               1 successful proof,   0 failures,  0 errors

Totals:                  182 successful proofs,  15 failures,   0 errors


Total time: 3227 seconds.
```

Appendix A

Proof Summary

The proof summary is the result of executing a command to attempt to prove all the proof declarations in the context. The only failures are in the automatically generated proof declarations for the type correctness conditions (tcc). The time given below is the running time on a SUN 3/470.

```
Proof summaries for modules on using chain of module top

Proof summaries for modules on using chain of module top

Module division_tcc:      7 successful proofs,  0 failures,  0 errors
Module tcc_proofs_tcc:    2 successful proofs,  1 failure,   0 errors
Module ica3_tcc:          0 successful proofs,  3 failures,  0 errors
Module ica4_tcc:          0 successful proofs,  2 failures,  0 errors
Module ica_tcc:           1 successful proof,   2 failures,  0 errors
Module lemma_final_tcc:   0 successful proofs,  5 failures,  0 errors
Module countmod_tcc:      3 successful proofs,  2 failures,  0 errors
Module tcc_proofs:       14 successful proofs,  0 failures,  0 errors
Module ica3:              8 successful proofs,  0 failures,  0 errors
Module ica2:             20 successful proofs,  0 failures,  0 errors
Module ica:               6 successful proofs,  0 failures,  0 errors
Module ica4:              8 successful proofs,  0 failures,  0 errors
Module basics:           25 successful proofs,  0 failures,  0 errors
Module readbounds:       12 successful proofs,  0 failures,  0 errors
Module lemma3:           24 successful proofs,  0 failures,  0 errors
Module countmod:         no proofs
Module clockassumptions:  9 successful proofs,  0 failures,  0 errors
Module lemma_final:       5 successful proofs,  0 failures,  0 errors
Module absmod:           15 successful proofs,  0 failures,  0 errors
Module division:         11 successful proofs,  0 failures,  0 errors
```

Bibliography

[1] Schneider, Fred B.: *Understanding Protocols for Byzantine Clock Synchronization.* Department of Computer Science, Cornell University, Technical Report 87-859, Ithaca, NY, August 1987.

[2] Rushby, John; von Henke, Friedrich; and Owre, Sam: *An Introduction to Formal Specification and Verification Using* EHDM. Computer Science Laboratory, SRI International, Technical Report SRI-CSL-91-2, Menlo Park, CA, Feb. 1991.

[3] Lamport, L.; and Melliar-Smith, P.M.: Synchronizing Clocks in the Presence of Faults. *Journal of the ACM*, vol. 32, no. 1, January 1985, pp. 52–78.

[4] Garman, J. R.: The "Bug" Heard 'Round The World. *ACM SIGSOFT Software Engineering Notes*, vol. 6, no. 5, October 1981, pp. 3–10.

[5] Anonymous: Reprogramming Capability Proves Key to Extending Voyager 2's Journey. *Aviation Week and Space Technology*, August 7, 1989, pp. 72.

[6] Butler, Ricky W.: *A Survey of Provably Correct Fault-Tolerant Clock Synchronization Techniques.* NASA TM-100553, February 1988.

[7] Lamport, Leslie; Shostak, Robert; and Pease, Marshall: The Byzantine Generals Problem. *ACM TOPLAS*, vol. 4, no. 3, July 1982, pp. 382–401.

[8] Rushby, John; and von Henke, Friedrich: *Formal Verification of a Fault Tolerant Clock Synchronization Algorithm.* NASA CR-4239, June 1989.

that p fails exactly at t. Then clearly the end of the interval is earlier than t, but at any point earlier than t, processor p is nonfaulty and has yet to read T. This "natural" definition of the end of an interval thus yields a contradiction. Many similar problem arose frequently in attempting to set down the clock axioms. The most natural statement of these axioms often turned out to be either wrong or too restrictive. It is also important to observe that these problems would never have been noticed in most informal presentations since these details, though important, would have been largely ignored.

The most useful features of EHDM for this verification were the decision procedures for linear integer and real inequalities and equalities. The informal proof is of course replete with long chains of inequality reasoning, and the decision procedures handled those steps in a fairly mechanical manner. The higher-order features of the language were also used to formalize the conditions of *translation invariance*, *precision enhancement*, and *accuracy preservation*, but these were not essential. These could have also been formalized in terms of lists or finite arrays. The language of EHDM underwent a number of improvements during this project, and not all of these improvements have been exploited in this proof. The use of predicate subtypes would have permitted the introduction of types corresponding to the non-negative and the positive numbers.

Fault-tolerant distributed protocols are sufficiently delicate to warrant careful, formal, mechanized analysis. Schneider's presentation of Byzantine fault-tolerant clock synchronization protocols provides a valuable mathematical framework for such an analysis. The machine-checked proof of Schneider's protocol led to a more precise formulation of the protocol and a more closely reasoned proof. It is inconceivable that the same degree of logical rigor and accuracy could be achieved without computational assistance.

argument. The derivation we present is extremely tight, given the structure of the proof. Schneider's monotonicity condition is avoided in the proofs here. This condition is used heavily by Schneider in his arguments, but it actually turns out to be false for many protocols. The statement of accuracy preservation is also slightly different here from that of Schneider. Schneider also presents the proof for the case of continuous resynchronization which is not handled here.

The initial proof using EHDM took about a month. The proof has been considerably revised and improved since that first effort. Verifying that the egocentric mean function of ICA satisfied the conditions of *translation invariance, accuracy preservation*, and *precision enhancement*, took about two weeks. The EHDM modules are listed in Appendix B. The proof involves 182 theorems or lemmas. A rerun of the entire proof on a SUN 3/470 takes 3227 CPU seconds (see Appendix A).

An early difficulty in the verification attempt was in arriving at a satisfactory formalization that suitably revised the one from Schneider. The proper treatment of failure proved to be a pervasive and important difficulty. Unlike other similar informal and machine-verified proofs, our formalization was careful to permit processors to fail at any time. Rushby and von Henke [8], for example, regard processors as nonfaulty in an interval between synchronizations only if they have been nonfaulty for the entire interval. This is an adequate model for most practical purposes but it is less general because it does not distinguish between processors that may have failed at the beginning of the interval and those that failed at the very end of an interval. An even coarser model, and the one unwittingly used in most informal presentations of clock synchronization, is one where the only correct processors are those that never fail. In some sense, this is acceptable since often the only significant requirement is that a sufficient number of processors be nonfaulty at any given time. However, such a formalization allows no conclusion to be drawn regarding a processor which has yet to fail but does eventually fail, since it is regarded as always having been faulty.

To illustrate the circularity lurking in the formalization of time and failure, consider the following seemingly natural formalization of nonfaultiness in an interval. Suppose that a processor is described as nonfaulty for an interval if it functions normally through the end of the interval. Let the end of the interval be the time at which the nonfaulty clocks indicate a certain reading or have performed a certain operation such as resetting their readings. Suppose, for example, that the end of the interval is given by the time t when the slowest of the "nonfaulty" clocks p reads T. Now suppose

Chapter 4

Conclusions

Rigorously proving the correctness of distributed protocols is an extremely difficult task, with or without mechanical assistance. Fault-tolerant clock synchronization is an excellent example of a problem where the algorithms, though often simple, are not at all easily verified. In such cases, it is extremely important to have certain organizing principles which capture the common features of the various protocols with convincing generality. Schneider's schema for Byzantine clock synchronization provides such principles to unify the presentation and proofs of a number of different protocols. Schneider starts with certain axioms constraining the behaviors of clocks, the selection of synchronization times, and the convergence functions. He uses these constraints to derive a bound on the skew between any two nonfaulty clocks. It is worth noting for the discussion below that Schneider's work is described in an unpublished technical report that has not had the benefit of widespread examination.

The formalization here revises a few details from Schneider's presentation. Schneider's notion of a global signal to trigger resynchronization has been dropped because such a notion is difficult to instantiate for many protocols. Though the quantities r_{max} and r_{min} have a different meaning from Schneider's, these differences ought not to matter in any of the bounds derived. For instance, r_{max} here bounds $t_p^{i+1} - t_p^i$, but Schneider's bound on this quantity would be $r_{max} + \beta$. However, the significant quantity in the proof is the difference $t_p^{i+1} - t_q^i$ and the bound on this quantity is $r_{max} + \beta$ in either formalization. In other words, Schneider's bounds on δ and δ_S ought to have been the same as those derived in Section 2.3.2, but there were certain minor errors of algebra in his proofs and some latitude in his ica_translation_invariance: **Lemma**

$N > 0 \supset \text{icalg}(p, (\lambda\, q\colon \text{fun}(q) + X), Y) = \text{icalg}(p, \text{fun}, Y) + X$

icalg_precision_enhancement: **Lemma**

$$\begin{array}{l} \text{ppred}(p) \wedge \text{ppred}(q) \\ \qquad\qquad \wedge\ \text{count}(\text{ppred}, N) \geq N - \text{maxfaults} \\ \qquad\qquad\quad \wedge\ \text{okay\_pairs}(\text{fun1}, \text{fun2}, X, \text{ppred}) \\ \qquad\qquad\qquad \wedge\ \text{okay\_Readpred}(\text{fun1}, Z, \text{ppred}) \wedge \text{okay\_Readpred}(\text{fun2}, Z, \text{ppred}) \\ \qquad \supset \text{icalg}(p, \text{fun1}, \Delta) - \text{icalg}(q, \text{fun2}, \Delta) \leq \text{icalg\_Pi}(X, Z) \end{array}$$

icalg_accuracy_preservation: **Lemma**

$$\begin{array}{l} \text{ppred}(p) \wedge \text{ppred}(q) \\ \qquad\qquad \wedge\ \text{count}(\text{ppred}, N) \geq N - \text{maxfaults} \wedge \text{okay\_Readpred}(\text{fun}, X, \text{ppred}) \\ \qquad \supset |\text{icalg}(p, \text{fun}, \Delta) - \text{fun}(q)| \\ \qquad\quad \leq ((N - \text{maxfaults}) \star X + \text{maxfaults} \star (X + \Delta))/N \end{array}$$

Figure 3.10: Properties of egocentric mean from modules ica, ica3, and ica4

process: **Type is** nat
event: **Type is** nat
time: **Type is** number
Clocktime: **Type is** number
$l, m, n, p, q, p_1, p_2, q_1, q_2, p_3, q_3$: **Var** process
i, j, k: **Var** event
x, y, z, r, s, t: **Var** time
X, Y, Z, R, S, T: **Var** Clocktime
fun, γ, θ: **Var** function[process $\rightarrow$ Clocktime]
ppred, ppred1, ppred2: **Var** function[process $\rightarrow$ bool]
sigma_size: function[function[process $\rightarrow$ Clocktime], process $\rightarrow$ process] =
    $(\lambda\, \mathrm{fun}, i\colon i)$
sigma: function[function[process $\rightarrow$ Clocktime], process $\rightarrow$ Clocktime] =
    $(\lambda\, \mathrm{fun}, i\colon (\ \mathbf{if}\ i > 0\ \mathbf{then}\ \mathrm{fun}(i-1) + \mathrm{sigma}(\mathrm{fun}, i-1)\ \mathbf{else}\ 0\ \mathbf{end\ if}))$
    by sigma_size
fix: function[Clocktime, Clocktime, Clocktime $\rightarrow$ Clocktime] =
    $(\lambda\, X, Y, Z\colon (\ \mathbf{if}\ |Y - Z| \leq X\ \mathbf{then}\ Y\ \mathbf{else}\ Z\ \mathbf{end\ if}))$
iconv: function[process, function[process $\rightarrow$ Clocktime], Clocktime
    $\rightarrow$ Clocktime] =
    $(\lambda\, p, \mathrm{fun}, Y\colon \mathrm{sigma}((\lambda\, q\colon \mathrm{fix}(Y, \mathrm{fun}(q), \mathrm{fun}(p))), N))$
icalg: function[process, function[process $\rightarrow$ Clocktime], Clocktime
    $\rightarrow$ Clocktime] = $(\lambda\, p, \mathrm{fun}, Y\colon \mathrm{iconv}(p, \mathrm{fun}, Y)/N)$

Figure 3.9: Egocentric mean from module ica

okayClocks: function[process, process, nat $\rightarrow$ bool] =
$(\lambda\, p, q, i\colon (\forall\, t\colon$
$t \geq 0 \wedge t < t^i_{(p \Uparrow q)[i]} \wedge \mathrm{correct}(p,t) \wedge \mathrm{correct}(q,t)$
$\supset |VC_p(t) - VC_q(t)| \leq \delta))$

lemma3_3: **Lemma** $\beta \leq r_{min}$
$\wedge\, \mu \leq \delta_S \wedge \pi(2 * \Lambda + 2 * \beta \star \rho, \delta_S + 2 * ((r_{max} + \beta) \star \rho + \Lambda)) \leq \delta_S$
$\wedge\, \delta_S + 2 * r_{max} \star \rho \leq \delta$
$\wedge\, \alpha(\delta_S + 2 * (r_{max} + \beta) \star \rho + 2 * \Lambda) + \Lambda + 2 * \beta \star \rho \leq \delta$
$\supset$ okayClocks(p, q, i)

Figure 3.8: Skew up to ith resynchronization from module lemma3

and q otherwise.

The EHDM definition of the egocentric mean function is given by `icalg` in Figure 3.9.

The verification of the *translation invariance*, *precision enhancement*, and *accuracy preservation* properties of the egocentric mean function is presented in Figure 3.10. The proof chain analyses for these theorems appear in Appendices C.2, C.3, and C.4.

$$
\begin{aligned}
&\text{agreement: } \textbf{Lemma } \beta \le r_{min}\\
&\quad \wedge \mu \le \delta_S \wedge \pi(2*\Lambda + 2*\beta\star\rho, \delta_S + 2*((r_{max}+\beta)\star\rho + \Lambda)) \le \delta_S\\
&\quad\quad \wedge \delta_S + 2*r_{max}\star\rho \le \delta\\
&\quad\quad\quad \wedge \alpha(\delta_S + 2*(r_{max}+\beta)\star\rho + 2*\Lambda) + \Lambda + 2*\beta\star\rho \le \delta\\
&\quad\quad\quad\quad \wedge t \ge 0 \wedge \mathrm{correct}(p,t) \wedge \mathrm{correct}(q,t)\\
&\quad \supset |VC_p(t) - VC_q(t)| \le \delta
\end{aligned}
$$

Figure 3.6: Main Theorem in module lemma_final

$$
\begin{aligned}
&\text{okaymaxsync: function[nat, Clocktime} \to \text{bool]} =\\
&\quad (\lambda\, i, X: (\forall\, p, q:\\
&\quad\quad \mathrm{correct}(p, t^i_{p,q}) \wedge \mathrm{correct}(q, t^i_{p,q})\\
&\quad\quad\quad \supset |IC^i_p(t^i_{p,q}) - IC^i_q(t^i_{p,q})| \le X))
\end{aligned}
$$

$$
\begin{aligned}
&\text{lemma\_2: } \textbf{Lemma } \beta \le r_{min}\\
&\quad \wedge \mu \le X \wedge \pi(2*\Lambda + 2*\beta\star\rho, X + 2*((r_{max}+\beta)\star\rho+\Lambda)) \le X\\
&\quad \supset \mathrm{okaymaxsync}(i, X)
\end{aligned}
$$

Figure 3.7: Skew immediately following resynchronization from module readbounds

3.2 The Proof Highlights

The conclusion corresponding to Theorem 2.1.1 is the theorem `agreement` that appears in the module `lemma_final` listed at page 63 of Appendix B. This theorem is displayed in Figure 3.6. It should be compared to the statement of Theorem 2.1.1 (page 8) and to the conditions at the end of Section 2.3.2 (page 21). The axioms, definitions, and lemmas used, whether in a direct or indirect manner, in the proof of `agreement` are analyzed in Appendix C.1 to ensure that all proof obligations have been discharged. Both the process and the result of checking these dependencies are part of what is termed the *proof chain analysis*.

The verified version of Theorem 2.3.1 is given in Figure 3.7 extracted from the module `readbounds` listed at page 55 of Appendix B.

The verified version of Theorem 2.3.2 appears in Figure 3.8 which is taken from the module `lemma3` listed at page B of Appendix B. The expression $t^i_{(p \Uparrow q)[i]}$ is an alternative notation for $t^i_{p,q}$ since $(p \Uparrow q)[i]$ represents p if $t^i_p \ge t^i_q$,

Readerror: **Axiom** $\text{correct}(p, t_p^{i+1}) \wedge \text{correct}(q, t_p^{i+1})$
$\supset |\Theta_p^{i+1}(q) - IC_q^i(t_p^{i+1})| \leq \Lambda$

translation_invariance: **Axiom**
$X \geq 0 \supset cfn(p, (\lambda\, p_1 \rightarrow \text{Clocktime}: \gamma(p_1) + X)) = cfn(p, \gamma) + X$

ppred: **Var** function[process → bool]
maxfaults: process
okay_Readpred: function[function[process → Clocktime], Clocktime,
function[process → bool] → bool] =
$(\lambda\, \gamma, Y, \text{ppred}: (\forall\, l, m: \text{ppred}(l) \wedge \text{ppred}(m) \supset |\gamma(l) - \gamma(m)| \leq Y))$
okay_pairs: function[function[process → Clocktime],
function[process → Clocktime], Clocktime,
function[process → bool] → bool] =
$(\lambda\, \gamma, \theta, X, \text{ppred}: (\forall\, p_3: \text{ppred}(p_3) \supset |\gamma(p_3) - \theta(p_3)| \leq X))$
N: process

N_0: **Axiom** $N > 0$

N_maxfaults: **Axiom** $\text{maxfaults} \leq N$

precision_enhancement_ax: **Axiom**
$\text{count}(\text{ppred}, N) \geq N - \text{maxfaults}$
$\wedge\, \text{okay\_Readpred}(\gamma, Y, \text{ppred})$
$\wedge\, \text{okay\_Readpred}(\theta, Y, \text{ppred})$
$\wedge\, \text{okay\_pairs}(\gamma, \theta, X, \text{ppred}) \wedge \text{ppred}(p) \wedge \text{ppred}(q)$
$\supset |cfn(p, \gamma) - cfn(q, \theta)| \leq \pi(X, Y)$

correct_count: **Axiom** $\text{count}((\lambda\, p: \text{correct}(p, t)), N) \geq N - \text{maxfaults}$

accuracy_preservation_ax: **Axiom**
$\text{okay\_Readpred}(\gamma, X, \text{ppred})$
$\wedge\, \text{count}(\text{ppred}, N) \geq N - \text{maxfaults} \wedge \text{ppred}(p) \wedge \text{ppred}(q)$
$\supset |cfn(p, \gamma) - \gamma(q)| \leq \alpha(X)$

Figure 3.5: Conditions on Logical Clocks in module clockassumptions

init: **Axiom** correct$(p, 0) \supset PC_p(0) \geq 0 \wedge PC_p(0) \leq \mu$

correct_closed: **Axiom** $s \geq t \wedge$ correct$(p, s) \supset$ correct(p, t)

rate_1: **Axiom** correct$(p, s) \wedge s \geq t \supset PC_p(s) - PC_p(t) \leq (s - t) \star (1 + \rho)$

rate_2: **Axiom** correct$(p, s) \wedge s \geq t \supset PC_p(s) - PC_p(t) \geq (s - t) \star (1 - \rho)$

rts0: **Axiom** correct$(p, t) \wedge t \leq t_p^{i+1} \supset t - t_p^i \leq r_{max}$

rts1: **Axiom** correct$(p, t) \wedge t \geq t_p^{i+1} \supset t - t_p^i \geq r_{min}$

rts_0: **Lemma** correct$(p, t_p^{i+1}) \supset t_p^{i+1} - t_p^i \leq r_{max}$

rts_1: **Lemma** correct$(p, t_p^{i+1}) \supset t_p^{i+1} - t_p^i \geq r_{min}$

rts2: **Axiom** correct$(p, t) \wedge t \geq t_q^i + \beta \wedge$ correct$(q, t) \supset t \geq t_p^i$

rts_2: **Axiom** correct$(p, t_p^i) \wedge$ correct$(q, t_q^i) \supset t_p^i - t_q^i \leq \beta$

synctime_0: **Axiom** $t_p^0 = 0$

Figure 3.3: Physical clock axioms in module clockassumptions

VClock_defn: **Axiom**
correct$(p, t) \wedge t \geq t_p^i \wedge t < t_p^{i+1} \supset VC_p(t) = IC_p^i(t)$

Adj: function[process, event $\rightarrow$ Clocktime] =
($\lambda\, p, i$: (**if** $i > 0$ **then** $cfn(p, \Theta_p^i) - PC_p(t_p^i)$ **else** 0 **end if**))

IClock_defn: **Axiom** correct$(p, t) \supset IC_p^i(t) = PC_p(t) + \mathrm{Adj}(p, i)$

Figure 3.4: Clock definitions in module clockassumptions

delta_0: **Axiom** $\delta \geq 0$

mu_0: **Axiom** $\mu \geq 0$

rho_0: **Axiom** $\rho \geq 0$

rho_1: **Axiom** $\rho < 1$

rmin_0: **Axiom** $r_{min} > 0$

rmax_0: **Axiom** $r_{max} > 0$

beta_0: **Axiom** $\beta \geq 0$

lamb_0: **Axiom** $\Lambda \geq 0$

Figure 3.2: Constants in module clockassumptions

axiom `init` essentially corresponds to *initial skew*. Axiom `correct_closed` asserts that a failed processor never recovers. Axioms `rate_1` and `rate_2` together express the *bounded drift* condition. The axioms `rts0` and `rts1` capture the *bounded interval* condition. These axioms look strange because the variable t, needed to properly capture the correctness condition, appears in them but not in *bounded interval*. Most of the obvious ways of stating these axioms are either too restrictive or wrong. The axiom `rts2` captures *bounded delay*, and `synctime_0` is just *initial synchronization*. The condition of *nonoverlap* appears as an antecedent to the concluding theorem rather than as an axiom. In the LaTeX format below, multiplication is represented by $*$ as well as $\star$. These are synonymous, but the latter represents the uninterpreted form of multiplication whereas the former is interpreted by the linear arithmetic decision procedures of EHDM.

The definitions of the virtual clock and the interval clock in terms of the physical clock appear in Figure 3.4. These correspond to (2.1.1), (2.1.4), and (2.1.3), respectively.

The conditions on the convergence function appear in Figure 3.5. The axiom `Readerror` corresponds to the condition *reading error*. The axiom `correct_count` corresponds to *bounded faults*. The remaining correspondences should be self-evident.

Some of the definitions and lemmas from the module `clockassumptions` have been omitted from this discussion.

clockassumptions: **Module**

Using arith, countmod

Exporting all with countmod, arith

Theory

process: **Type is** nat
event: **Type is** nat
time: **Type is** number
Clocktime: **Type is** number
$l, m, n, p, q, p_1, p_2, q_1, q_2, p_3, q_3$: **Var** process
i, j, k: **Var** event
x, y, z, r, s, t: **Var** time
X, Y, Z, R, S, T: **Var** Clocktime
γ, θ: **Var** function[process $\rightarrow$ Clocktime]
$\delta, \mu, \rho, r_{min}, r_{max}, \beta, \Lambda$: number
$PC_{\star 1}(\star 2), VC_{\star 1}(\star 2)$: function[process, time $\rightarrow$ Clocktime]
$t_{\star 1}^{\star 2}$: function[process, event $\rightarrow$ time]
$\Theta_{\star 1}^{\star 2}$: function[process, event $\rightarrow$ function[process $\rightarrow$ Clocktime]]
$IC_{\star 1}^{\star 2}(\star 3)$: function[process, event, time $\rightarrow$ Clocktime]
correct: function[process, time $\rightarrow$ bool]
cfn: function[process, function[process $\rightarrow$ Clocktime] $\rightarrow$ Clocktime]
π: function[Clocktime, Clocktime $\rightarrow$ Clocktime]
α: function[Clocktime $\rightarrow$ Clocktime]

Figure 3.1: Declarations from module clockassumptions

Chapter 3

The Verification of Schneider's Protocol using EHDM

The outline in Chapter 2 was adapted from Schneider's description but differs from his presentation in many of the details. The mechanized formalization using EHDM follows the informal description in Chapter 2 fairly closely. We illustrate the highlights of the machine proof below and indicate the correspondence to the informal description. Details regarding the language and capabilities of EHDM are contained in the EHDM tutorial document [2].

3.1 The Clock Assumptions

This section contains the EHDM formalization of the conditions axiomatizing the behavior of clocks. These axioms are contained in a module labeled `clockassumptions` that is listed in Appendix B starting from page 47. Figure 3.1 contains the type declarations for some of the variables and constants used in `clockassumptions`. The `clockassumptions` module makes use of the module `arith`, which contains the basic arithmetic facts, and `countmod`, which introduces a counting function. Nonfaultiness is expressed by the predicate `correct`.

The first few axioms express various minor constraints on the constants as shown in Figure 3.2.

The axioms constraining the physical behavior of the clock appear in Figure 3.3. Since we require the initial skew bound μ to not exceed δ_S, mean to be up to $f \times (x + \Delta)/N$ away from a good clock. The total thus yields

$$\alpha(x) = x + \frac{f\Delta}{N}.$$

The final step is to demonstrate the failure of the monotonicity condition for ICA. The monotonicity condition mentioned at the end of Section 2.2 asserts that if for each processor l, $\theta(l) \geq \gamma(l)$, then $cfn(p, \theta) \geq cfn(p, \gamma)$. The key reason for the failure of the monotonicity condition is that if some readings in γ were ignored because they were more than Δ below $\gamma(p)$ but were increased in θ so that they were no longer ignored, then $cfn(p, \theta)$ could effectively be smaller than $cfn(p, \gamma)$ even though for every l, $\theta(l) \geq \gamma(l)$. More specifically, let $\theta(p) = \gamma(p)$. Observe now that if there is some l such that $\theta(l) + \Delta < \theta(p)$, but with $\gamma(p) > \gamma(l) \geq \gamma(p) - \Delta$, then $fix_p(\theta(l)) > fix_p(\gamma(l))$ holds. So, it is possible to have $fix_p(\theta(l)) > fix_p(\gamma(l))$, even though we have $\theta(l) < \gamma(l)$.

For the mechanical verification of ICA as an instance of Schneider's protocol, we have verified the constraints, i.e., *translation invariance*, *precision enhancement*, and *accuracy preservation*, hold for the egocentric mean taken as a convergence function. We have not yet instantiated the quantities r_{min}, r_{max} and β, nor verified the conditions of *bounded interval*, *bounded delay* and *nonoverlap*, since these depend on specific implementation choices. It would also be useful to mechanically verify various other Byzantine fault tolerant clock synchronization algorithms to be instances of Schneider's scheme.

which is no greater than

$$\frac{\sum_{l=0}^{N-1} |fix_p(\gamma(l)) - fix_q(\theta(l))|}{N}.$$

This in turn can be rewritten as

$$\frac{\sum_{l\in C} |fix_p(\gamma(l)) - fix_q(\theta(l))|}{N} + \frac{\sum_{l\notin C} |fix_p(\gamma(l)) - fix_q(\theta(l))|}{N}.$$

Assuming $y \leq \Delta$ and $l \in C$, we get $fix_p(\gamma(l))$ to be $\gamma(l)$ and $fix_q(\theta(l))$ to be $\theta(l)$, so that

$$|fix_p(\gamma(l)) - fix_q(\theta(l))| \leq x$$

and hence,

$$\frac{\sum_{l\in C} |fix_p(\gamma(l)) - fix_q(\theta(l))|}{N} \leq \frac{(N-f)x}{N}.$$

For $l \notin C$, the difference

$$|fix_p(\gamma(l)) - fix_q(\theta(l))| \leq 2\Delta + |\gamma(p) - \theta(q)| \leq 2\Delta + x + y$$

and hence

$$\frac{\sum_{l\notin C} |fix_p(\gamma(l)) - fix_q(\theta(l))|}{N} \leq \frac{2f\Delta + fx + fy}{N}.$$

We thus get, when $y \leq \Delta$, that

$$\pi(x, y) = \frac{(N-f)x}{N} + \frac{2f\Delta + fx + fy}{N}. \tag{2.4.63}$$

In the typical situation when the egocentric mean is computed, the quantity x representing the reading error is negligible, and y representing the clock skew is bounded by Δ. Since the skew following synchronization should be smaller than Δ, we can see that in Equation (2.4.63), the number of failed processors f should be below $N/3$. Though the derivation of $\pi(x, y)$ for the case when $y > \Delta$ is carried out in the machine proof, it is not essential since in practice, y will not exceed Δ

To show that cfn_I satisfies accuracy preservation, it is sufficient to observe that if all the nonfaulty clocks are within x of each other, then the nonfaulty clocks can cause the egocentric mean to be at most $(N-f)x/N$ away from any nonfaulty clock. The faulty clocks can cause the egocentric

2.4 ICA as an instance of Schneider's scheme

The egocentric mean function which is used as a convergence function in the Interactive Convergence Algorithm of Lamport and Melliar-Smith [3] can be shown to satisfy Schneider's conditions of translation invariance, precision enhancement, and accuracy preservation.

With the interactive convergence algorithm, the convergence function cfn_I takes the *egocentric mean* of p's estimate of the readings of the N clocks numbered from 0 to $N-1$, i.e., any readings that are more than Δ away from p's own reading are replaced by p's own reading. This yields the definition

$$cfn_I(p,\theta) = \frac{\sum_{l=0}^{N-1} fix_p(\theta(l))}{N} \tag{2.4.56}$$

where

$$fix_p(x) = \begin{cases} x & \text{if } |x-\theta(p)| \leq \Delta \\ \theta(p) & \text{otherwise.} \end{cases}$$

Translation invariance follows from the observation that

$$fix_p((\lambda l\colon \theta(l)+t)(q)) = fix_p(\theta(q)) + t \tag{2.4.57}$$

and

$$\frac{\sum_{l=0}^{N-1}(\theta(l)+t)}{N} = \frac{\sum_{l=0}^{N-1}(\theta(l))}{N} + t \tag{2.4.58}$$

To demonstrate *precision enhancement*, we start with a set of processors C of cardinality $|C|$ greater than $N-F$. Let f be $N-|C|$. The hypotheses for precision enhancement are that for any l and m in C,

$$|\gamma(l)-\theta(l)| \leq x \tag{2.4.59}$$

$$|\gamma(l)-\gamma(m)| \leq y \tag{2.4.60}$$

$$|\theta(l)-\theta(m)| \leq y. \tag{2.4.61}$$

We need to determine $\pi(x,y)$ so that for any p and q in C, we get

$$|cfn_I(p,\gamma) - cfn_I(q,\theta)| \leq \pi(x,y). \tag{2.4.62}$$

This difference can be rewritten as

$$\left| \frac{\sum_{l=0}^{N-1} fix_p(\gamma(l))}{N} - \frac{\sum_{l=0}^{N-1} fix_q(\theta(l))}{N} \right|$$

Therefore δ has to be chosen to satisfy

$$\alpha(\delta_S + 2\rho(r_{max} + \beta) + 2\Lambda) + \Lambda + 2\rho\beta \leq \delta. \tag{2.3.54}$$

This completes both cases of the proof of Theorem 2.3.2. ∎

Theorem 2.3.2 forms the induction step in the proof of the following theorem.

Theorem 2.3.3 *For any two nonfaulty clocks p, q, and* $t < t^i_{p,q}$

$$|VC_p(t) - VC_q(t)| \leq \delta \tag{2.3.55}$$

Proof. The proof is by straightforward induction over i. When $i = 0$, the antecedent fails since $t^i_{p,q} = 0$. The induction hypothesis asserts that for $t < t^i_{p,q}$, the quantity $|VC_p(t) - VC_q(t)|$ does not exceed δ. The induction conclusion requires showing that δ bounds $|VC_p(t) - VC_q(t)|$ even when $t < t^{i+1}_{p,q}$. We observe that either $t < t^i_{p,q}$, in which case the conclusion follows from the induction hypothesis, or, $t^i_{p,q} \leq t < t^{i+1}_{p,q}$, and the conclusion easily follows from Theorem 2.3.2. ∎

One small step remains in the proof of *bounded skew* from Theorem 2.3.3.

Theorem 2.3.4 *For any* $t > 0$ *and nonfaulty processors p and q, there is an i such that*

$$t < t^i_{p,q}.$$

Proof. By *bounded interval*, $0 < r_{min} \leq t^{j+1}_p - t^j_p$. Thus, $t^{j+1}_p > jr_{min}$. If we let i be $\lceil t/r_{min} \rceil + 1$, then $t^i_p > t$. ∎

The main result, Theorem 2.1.1, easily follows from the Theorems 2.3.3 and 2.3.4.

We take note of the various conditions on δ and δ_S[6]:

1. $\pi(2\Lambda + 2\beta\rho, \delta_S + 2\rho(r_{max} + \beta) + 2\Lambda) \leq \delta_S$, by 2.3.40.
2. $\delta_S + 2\rho r_{max} \leq \delta$, by 2.3.45
3. $\alpha(\delta_S + 2\rho(r_{max} + \beta) + 2\Lambda) + \Lambda + 2\rho\beta \leq \delta$, by 2.3.54

This concludes the informal presentation of the proof.

[6]Note that these conditions are significantly different from those derived by Schneider [1] due to various inaccuracies that have been corrected in the mechanical proof.

Case 2: Assuming $t_q^{i+1} < t < t_{p,q}^{i+1}$. In this interval, $VC_q(t) = IC_q^{i+1}(t)$, whereas $VC_p(t) = IC_p^i(t)$. The strategy here is to bound the skew at t_q^{i+1} and then compute the additional quantity by which the clocks can drift apart in the given interval. By Equations (2.1.5) and (2.1.4), we have

$$|VC_p(t_q^{i+1}) - VC_q(t_q^{i+1})| = |IC_p^i(t_q^{i+1}) - cfn(q, \Theta_q^{i+1})|. \tag{2.3.46}$$

We now need to use the condition of *accuracy preservation* with C as the subset of nonfaulty processors as allowed by *bounded faults*. To satisfy the hypothesis of *accuracy preservation,* we need a bound x such that, for any pair of nonfaulty clocks l and m,

$$|\Theta_q^{i+1}(l) - \Theta_q^{i+1}(m)| \leq x. \tag{2.3.47}$$

The next few steps are similar to those required to establish Hypotheses 2 and 3 of *precision enhancement*. By Equation (2.2.16), we have

$$\begin{aligned} |\Theta_q^{i+1}(l) - IC_l^i(t_q^{i+1})| &\leq \Lambda & (2.3.48)\\ |\Theta_q^{i+1}(m) - IC_m^i(t_q^{i+1})| &\leq \Lambda. & (2.3.49) \end{aligned}$$

By Equation (2.2.15), $t_q^{i+1} - t_{l,m}^i \leq r_{max} + \beta$ holds. Theorem 2.3.1 and (2.2.10) can now be applied to get

$$|IC_l^i(t_q^{i+1}) - IC_m^i(t_q^{i+1})| \leq \delta_S + 2\rho(r_{max} + \beta). \tag{2.3.50}$$

Letting x be $\delta_S + 2\rho(r_{max} + \beta) + 2\Lambda$, and substituting p for q and q for p in *accuracy preservation*, we can combine Equations (2.3.48), (2.3.49), and (2.3.50), to get

$$|cfn(q, \Theta_q^{i+1}) - \Theta_q^{i+1}(p)| \leq \alpha(\delta_S + 2\rho(r_{max} + \beta) + 2\Lambda). \tag{2.3.51}$$

Since Equation (2.2.16) yields $|\Theta_q^{i+1}(p) - IC_p^i(t_q^{i+1})| \leq \Lambda$, it follows from Equations (2.3.51) and (2.3.46), that

$$\begin{aligned} &|VC_p(t_q^{i+1}) - VC_q(t_q^{i+1})|\\ &= \ |IC_p^i(t_q^{i+1}) - cfn(q, \Theta_q^{i+1})|\\ &\leq \ \alpha(\delta_S + 2\rho(r_{max} + \beta) + 2\Lambda) + \Lambda. & (2.3.52) \end{aligned}$$

Having bounded the skew at t_q^{i+1}, we can bound the skew over the interval $t_q^{i+1} \leq t < t_p^{i+1}$, by observing that $t_p^{i+1} - t_q^{i+1} \leq \beta$ by (2.2.12), and applying Equation (2.2.10) to derive the inequality,

$$|VC_p(t) - VC_q(t)| \leq \alpha(\delta_S + 2\rho(r_{max} + \beta) + 2\Lambda) + \Lambda + 2\rho\beta. \tag{2.3.53}$$

Combining Equations (2.3.34), (2.3.35), (2.3.36), and (2.3.38), we get

$$|\Theta_k^{i+1}(l) - \Theta_k^{i+1}(m)| \leq \delta_S + 2\rho(r_{max} + \beta) + 2\Lambda. \tag{2.3.39}$$

So the expression $\delta_S + 2\rho(r_{max} + \beta) + 2\Lambda$ is the required bound y satisfying both Hypotheses 2 and 3 of *precision enhancement*.

If we now choose δ_S so that

$$\pi(2\Lambda + 2\beta\rho, \delta_S + 2\rho(r_{max} + \beta) + 2\Lambda) \leq \delta_S, \tag{2.3.40}$$

then the conclusion of *precision enhancement* along with Equation (2.1.6) ensures that

$$|IC_p^{i+1}(t_p^{i+1}) - IC_q^{i+1}(t_p^{i+1})| \leq \delta_S$$

to complete the proof of Theorem 2.3.1. ∎

We have now shown that for any pair of nonfaulty processors p and q, the skew between their clock readings at $t_{p,q}^i$, given by $|IC_p^i(t_{p,q}^i) - IC_q^i(t_{p,q}^i)|$, does not exceed δ_S. The next step is to show that for any i, the clock skew between $t_{p,q}^i$ and $t_{p,q}^{i+1}$, is bounded.

Theorem 2.3.2 *For any two nonfaulty clocks p, q, and* $t_{p,q}^i \leq t < t_{p,q}^{i+1}$,

$$|VC_p(t) - VC_q(t)| \leq \delta. \tag{2.3.41}$$

Proof. Assume without loss of generality that $t_q^{i+1} \leq t_p^{i+1}$. The proof has two cases according to whether $t_{p,q}^i \leq t < t_q^{i+1}$ or $t_q^{i+1} \leq t < t_p^{i+1}$.

Case 1: Assuming $t_{p,q}^i \leq t < t_q^{i+1}$, from *bounded interval* we get $t - t_{p,q}^i \leq r_{max}$. By Equation (2.1.4), it is clear that for t in this interval $VC_p(t) = IC_p^i(t)$ and $VC_q(t) = IC_q^i(t)$. Then by (2.2.10) and (2.1.3), it follows that

$$|VC_p(t) - VC_q(t)| \leq |VC_p(t_{p,q}^i) - VC_q(t_{p,q}^i)| + 2\rho r_{max}. \tag{2.3.42}$$

Recall that Theorem 2.3.1 yields

$$|VC_p(t_{p,q}^i) - VC_q(t_{p,q}^i)| \leq \delta_S. \tag{2.3.43}$$

Combining Equations (2.3.42) and (2.3.43), we have

$$|VC_p(t) - VC_q(t)| \leq \delta_S + 2\rho r_{max}. \tag{2.3.44}$$

The bound δ should therefore be chosen so that

$$\delta_S + 2\rho r_{max} \leq \delta. \tag{2.3.45}$$

Putting together Equations (2.3.27), (2.3.28), and (2.3.29), we get the required inequality

$$|\Theta_q^{i+1}(l) + PC_q(t_p^{i+1}) - PC_q(t_q^{i+1}) - \Theta_p^{i+1}(l)| \le 2\rho\beta + 2\Lambda. \tag{2.3.30}$$

The substitution $2\rho\beta + 2\Lambda$ for x thus satisfies Hypothesis 1 of *precision enhancement*.

The next step is to satisfy Hypotheses 2 and 3 of *precision enhancement* for the specified substitutions. For these, we need a y such that for any nonfaulty processors l and m, the following inequalities hold.

$$\begin{aligned} |(\Theta_q^{i+1}(l) + PC_q(t_p^{i+1}) - PC_q(t_q^{i+1})) - \\ (\Theta_q^{i+1}(m) + PC_q(t_p^{i+1}) - PC_q(t_q^{i+1}))| &\le y \quad (2.3.31) \\ |\Theta_p^{i+1}(l) - \Theta_p^{i+1}(m)| &\le y \quad (2.3.32) \end{aligned}$$

Since (2.3.31) can be simplified by cancellation, both (2.3.31) and (2.3.32) can derived by deriving a bound y such that for all nonfaulty clocks k, l, and m, we get

$$|\Theta_k^{i+1}(l) - \Theta_k^{i+1}(m)| \le y \tag{2.3.33}$$

First note that

$$\begin{aligned} &|\Theta_k^{i+1}(l) - \Theta_k^{i+1}(m)| \\ &\le |\Theta_k^{i+1}(l) - IC_l^i(t_k^{i+1})| + |IC_l^i(t_k^{i+1}) - IC_m^i(t_k^{i+1})| + \\ &\quad |\Theta_k^{i+1}(m) - IC_m^i(t_k^{i+1})| \end{aligned} \tag{2.3.34}$$

In (2.3.34), we know by Equation (2.2.16) that

$$\begin{aligned} |\Theta_k^{i+1}(l) - IC_l^i(t_k^{i+1})| &\le \Lambda \text{ and} \quad (2.3.35) \\ |\Theta_k^{i+1}(m) - IC_m^i(t_k^{i+1})| &\le \Lambda \quad (2.3.36) \end{aligned}$$

By the induction hypothesis (2.3.21), we get

$$|IC_l^i(t_{l,m}^i) - IC_m^i(t_{l,m}^i)| \le \delta_S. \tag{2.3.37}$$

We know by (2.2.15) that, $t_k^{i+1} - t_{l,m}^i \le r_{max} + \beta$. Then by (2.1.3), (2.2.10), and (2.3.37), we get

$$|IC_l^i(t_k^{i+1}) - IC_m^i(t_k^{i+1})| \le \delta_S + 2\rho(r_{max} + \beta). \tag{2.3.38}$$

The condition of *translation invariance* provides an estimate of $IC_q^{i+1}(t_p^{i+1})$ in terms of the convergence function *cfn*. With Θ_q^{i+1} for θ in Equation (2.2.17), we get

$$\begin{aligned}
&cfn(q, \Theta_q^{i+1}) + PC_q(t_p^{i+1}) - PC_q(t_q^{i+1}) \\
&\quad = \quad cfn(q, (\lambda n\colon \Theta_q^{i+1}(n) + PC_q(t_p^{i+1}) - PC_q(t_q^{i+1}))). \qquad (2.3.25)
\end{aligned}$$

By (2.3.24) and (2.3.25), the bound on the initial skews can be rewritten as follows:

$$\begin{aligned}
&|IC_q^{i+1}(t_p^{i+1}) - IC_p^{i+1}(t_p^{i+1})| \\
&\quad = \quad |cfn(q, (\lambda n\colon \Theta_q^{i+1}(n) + PC_q(t_p^{i+1}) - PC_q(t_q^{i+1}))) \\
&\qquad\quad -cfn(p, \Theta_p^{i+1})|. \qquad (2.3.26)
\end{aligned}$$

The right-hand side of (2.3.26) can be bounded by $\pi(x, y)$ for some x and y using *precision enhancement* with $(\lambda n\colon \Theta_q^{i+1}(n) + PC_q(t_p^{i+1}) - PC_q(t_q^{i+1}))$ for γ and Θ_p^{i+1} for θ. The set C in *precision enhancement* is taken to be the subset of nonfaulty clocks as permitted by *bounded faults*. The next few steps demonstrate that the remaining hypotheses of *precision enhancement* can be satisfied with these substitutions. To satisfy Hypothesis 1, we need to find an x such that for any nonfaulty l we can derive

$$|(\Theta_q^{i+1}(l) + PC_q(t_p^{i+1}) - PC_q(t_q^{i+1})) - \Theta_p^{i+1}(l)| \leq x.$$

As shown below, the value $2\rho\beta + 2\Lambda$ can be substituted for x. By Equation (2.2.16), we easily get

$$\begin{aligned}
|IC_l^i(t_q^{i+1}) - \Theta_q^{i+1}(l)| &\leq \Lambda, \text{ and} \qquad (2.3.27)\\
|IC_l^i(t_p^{i+1}) - \Theta_p^{i+1}(l)| &\leq \Lambda. \qquad (2.3.28)
\end{aligned}$$

Note that $t_p^{i+1} - t_q^{i+1} \leq \beta$ by (2.2.12). So from Equation (2.1.3) and *bounded drift*, we have

$$\begin{aligned}
&|(IC_l^i(t_q^{i+1}) + PC_q(t_p^{i+1}) - PC_q(t_q^{i+1})) - IC_l^i(t_p^{i+1})| \\
&\quad = \quad |(PC_q(t_p^{i+1}) - PC_q(t_q^{i+1})) - (IC_l^i(t_p^{i+1}) - IC_l^i(t_q^{i+1}))| \\
&\quad = \quad |(PC_q(t_p^{i+1}) - PC_q(t_q^{i+1})) - (PC_l(t_p^{i+1}) - PC_l(t_q^{i+1}))| \\
&\quad \leq \quad |(1+\rho)(t_p^{i+1} - t_q^{i+1}) - (1-\rho)(t_p^{i+1} - t_q^{i+1})| \\
&\quad = \quad |2\rho(t_p^{i+1} - t_q^{i+1})| \\
&\quad \leq \quad 2\rho\beta. \qquad (2.3.29)
\end{aligned}$$

and *accuracy preservation,* are satisfied by many of the known Byzantine fault tolerant convergence functions [1].

2.3.2 The Proof

The details of the proof of *bounded skew* are presented below. Let $t_{p,q}^{i+1}$ denote $max(t_p^i, t_q^i)$. The first major step in Schneider's proof is to prove:

Theorem 2.3.1 *There is a bound δ_S such that for synchronization round i and any two nonfaulty processors p and q*

$$|IC_p^i(t_{p,q}^i) - IC_q^i(t_{p,q}^i)| \leq \delta_S. \tag{2.3.20}$$

Proof. The proof of Theorem 2.3.1 is by induction on the round number i.

Base case: When $i = 0$, by (2.2.13) we have $t_p^0 = t_q^0 = 0$. Then by Definitions (2.1.3) and (2.1.1), $IC_p^0(t_p^0) = PC_p(0)$ and $IC_q^0(t_p^0) = PC_q(0)$. The condition of *initial skew* asserts $|PC_p(0) - PC_q(0)| \leq \delta_S$. Hence, $|IC_p^0(0) - IC_q^0(0)|$ is also bounded by δ_S.

Induction case: The induction hypothesis asserts that for every pair of nonfaulty processors, l and m

$$|IC_l^i(t_{l,m}^i) - IC_m^i(t_{l,m}^i)| \leq \delta_S. \tag{2.3.21}$$

The goal is to establish for any pair of nonfaulty processors p and q, that

$$|IC_p^{i+1}(t_{p,q}^{i+1}) - IC_q^{i+1}(t_{p,q}^{i+1})| \leq \delta_S. \tag{2.3.22}$$

Without loss of generality, assume that t_q^{i+1} precedes t_p^{i+1} so that $t_{p,q}^{i+1} = t_p^{i+1}$. Then Equation (2.1.6) yields

$$IC_q(t_p^{i+1}) = cfn(q, \Theta_q^{i+1}) + PC_q(t_p^{i+1}) - PC_q(t_q^{i+1}). \tag{2.3.23}$$

By Equation (2.1.5), we have

$$IC_p^{i+1}(t_p^{i+1}) = cfn(p, \Theta_p^{i+1}). \tag{2.3.24}$$

The proof described in this section is itself a somewhat simplified rendering of the mechanically verified proof. The main difference is that in the mechanical proof, the faultiness of a processor is itself a time-varying property, i.e., processors can fail at any time. A brief overview is given below to provide an outline of the detailed proof. The words *processor* and *clock* are used interchangeably.

2.3.1 Overview

To establish the main result, Theorem 2.1.1, we must show that the skew, or absolute difference, between the readings of any two nonfaulty clocks p and q at time t, given by $|VC_p(t) - VC_q(t)|$, is bounded by a quantity δ. By the definition of VC in (2.1.4), this reduces to the following two cases:

1. When both clocks have been resynchronized for the i'th time but not for the $(i+1)$'th time, i.e., if $max(t_p^i, t_q^i) \leq t < min(t_p^{i+1}, t_q^{i+1})$, then the skew between $IC_p^i(t)$ and $IC_q^i(t)$ is bounded by δ, and

2. When only one clock, say q, has been resynchronized for the $(i+1)$'th time, i.e., if $t_q^{i+1} \leq t < t_p^{i+1}$, then the skew between $IC_p^i(t)$ and $IC_q^{i+1}(t)$ is bounded by δ.

For two nonfaulty clocks p and q, the time immediately following their i'th round of synchronization is $max(t_p^i, t_q^i)$. The main step in the argument is to show that the skew between the readings $IC_p^i(t)$ and $IC_q^i(t)$ at time $t = max(t_p^i, t_q^i)$, is bounded by a quantity δ_S. This is shown by induction on i, and employs the conditions of *initial skew, translation invariance,* and *precision enhancement*.

We now know that the clocks IC_p^i and IC_q^i start off no more than δ_S apart at $max(t_p^i, t_q^i)$. By *bounded interval* and *bounded drift*, the skew between $IC_p^i(t)$ and $IC_q^i(t)$ does not increase by more than $2\rho r_{max}$ in the interval $max(t_p^i, t_q^i) \leq t < min(t_p^i, t_q^i)$. Assuming that $t_q^{i+1} \leq t_p^{i+1}$, then the restriction of *accuracy preservation* on the convergence function is used to bound the skew between $IC_p^i(t_q^{i+1})$ and $IC_q^{i+1}(t_q^{i+1})$. By *bounded delay* and *bounded drift*, the additional skew between the readings $IC_p^i(t)$ and $IC_q^{i+1}(t)$ over the interval $t_q^{i+1} \leq t < t_p^{i+1}$ is no more than $2\rho\beta$. To obtain the final result, we need to constrain the quantities ρ, δ_S, r_{min}, r_{max}, and β so that the skew bounds derived over the various intervals are within δ. Schneider also shows that the restrictions of *translation invariance, precision enhancement,*

Condition 11 (accuracy preservation) *Given any subset C of the N clocks with $|C| \geq N - F$, and clock readings θ such that for any l and m in C, the bound $|\theta(l) - \theta(m)| \leq x$ holds, there is a bound $\alpha(x)$ such that for any q in C*

$$|cfn(p, \theta) - \theta(q)| \leq \alpha(x) \tag{2.2.19}$$

In addition to the conditions enumerated above, Schneider presents a condition called *monotonicity* that is actually not satisfied by several clock synchronization protocols. Fortunately, this condition turns out to be unnecessary in the derivation. The monotonicity condition asserts that if for each processor l, $\theta(l) \geq \gamma(l)$, then $cfn(p, \theta) \geq cfn(p, \gamma)$. The failure of the monotonicity condition for ICA is demonstrated in Section 2.4.

2.3 The Correctness Proof

The proof described below closely follows Schneider's outline. A few of the details are different, mainly reflecting corrections or perceived improvements. These seemingly small revisions do, however, lead to drastic changes in the statements of many of the theorems. The details of the correctness proof are both conceptually and notationally complicated. The formal arguments are extremely delicate to carry out carefully and correctly due to the additional consideration of processor failure. The true difficulty of constructing watertight proofs may not be apparent in the descriptions below since they only capture the end result of a mechanical verification and not the tenuous intermediate steps. It would be extremely difficult for even the most diligent mathematician to correctly capture all the details of such proofs without machine assistance. One difficulty is the care that is needed to ensure that no assumptions are made regarding failed clocks. Schneider [1], for instance, asserts, "We make no assumptions about the behavior of clocks at faulty processors — not even that they can be modeled by functions." The present formulation does not go as far as to avoid the use of functions to model the behavior of failed clocks but no constraints are placed on the values of these functions when a processor has failed. The use of functions does not seem to contradict any intuitive understanding of the physical behavior of failed clocks. The possibility of processor failure adds significantly to the complexity of the formalization as well as the proof.

more than x, then $cfn(p, \gamma)$ and $cfn(q, \theta)$ are within $\pi(x, y)$ of each other.[4] The parameter y will roughly correspond to the amount by which the clocks have drifted relative to each other and x roughly indicates the message delay in communicating clock values. Typically, the parameter y dominates x. The quantity $\pi(x, y)$ provides the bound on the skew δ_S immediately following resynchronization. For the precision to be truly enhanced, it is crucial for $\pi(x, y)$ to be smaller than y.

Condition 10 (precision enhancement) *Given any subset C of the N clocks with $|C| \geq N - F$, and clocks p and q in C, then for any readings γ and θ satisfying the conditions*

1. *for any l in C, $|\gamma(l) - \theta(l)| \leq x$*
2. *for any l, m in C, $|\gamma(l) - \gamma(m)| \leq y$*
3. *for any l, m in C, $|\theta(l) - \theta(m)| \leq y$*

there is a bound $\pi(x, y)$, such that

$$|cfn(p, \gamma) - cfn(q, \theta)| \leq \pi(x, y) \tag{2.2.18}$$

The final condition of *accuracy preservation* bounds the distance between the value of $cfn(p, \theta)$ and the nonfaulty entries in θ. If $t_q^i \leq t_p^i$, then *accuracy preservation*[5] can be used to bound the difference between $IC_q^{i+1}(t_q^{i+1})$ and $IC_p^i(t_q^{i+1})$.

[4]Note that the order of arguments to π are reversed from their order in Schneider's description [1].

[5]Footnote 7 in Schneider [1] explains the choice of the terms *precision enhancement* and *accuracy preservation*. 'Precision' is defined as the closeness with which a measurement can be reproduced, whereas 'accuracy' is the proximity of the measurement to the actual value being measured. The virtual clocks represent various measurements of real time. The condition of *precision enhancement* characterizes the closeness of these measurements to each other. The condition of *accuracy preservation* can be seen as bounding the drift rate of the virtual clock with respect to real time.

of the convergence function should be the same as adding x to each clock reading instead. Recall that the array of clock readings is represented by a function from clocks to readings so that *cfn* is a higher-order function.

Condition 9 (translation invariance) *For any function θ mapping clocks to clock values,*

$$cfn(p, (\lambda n: \theta(n) + x)) = cfn(p, \theta) + x \tag{2.2.17}$$

As a consequence of *translation invariance*, we know that at t_p^i, both p and q have been resynchronized and $VC_q(t_p^i) = cfn(q, (\lambda n: \Theta_q(n) + x))$ for some x, and $VC_p(t_p^{i+1}) = cfn(p, \Theta_p)$. We clearly need some condition to bound the difference between these two values of the convergence function to within δ_S. The condition of *precision enhancement* allows exactly such a comparison between values of the convergence function based on the range of values of some subset of the clock readings that intuitively correspond to the readings of nonfaulty clocks.

In the statement of *precision enhancement*, γ and θ are any two arrays (or functions) of clock readings, and C is to be intuitively interpreted as the subset of nonfaulty processors. This interpretation of C is permissible by the *bounded faults* condition. The reason it is not directly taken to be the set of nonfaulty clocks is because the protocol cannot assume that any individual clock can distinguish the faulty from the nonfaulty clocks. The convergence functions for some protocols can neglect readings of nonfaulty clocks while considering readings of faulty clocks.

Precision enhancement is used to bound the skew between two clocks immediately after both have been resynchronized whereas *accuracy preservation* is used to bound the skew between a clock that has been resynchronized and one that has yet to be resynchronized in the ith round. The condition of *precision enhancement* bounds the skew between two clocks as computed by the convergence function, based on the skews between the clock readings that are inputs to the convergence function. We will refer to the clocks in C as C-clocks. Precision enhancement then asserts that if the readings of different C-clocks in γ fall within a range y as do the C-clock readings in θ, and the corresponding readings in γ and in θ of any C-clock differ by no in this reading is assumed to be bounded by Λ.

Condition 7 (reading error) *For nonfaulty clocks p and q,*

$$|IC_q^i(t_p^{i+1}) - \Theta_p^{i+1}(q)| \leq \Lambda \tag{2.2.16}$$

The above conditions turn out to be sufficient to bound the skew in the period between successive rounds of synchronization in terms of the skew bound δ_S immediately following synchronization. The conditions below of *bounded faults*, *translation invariance*, and *precision enhancement*, are needed to derive the skew bound δ_S. The condition of *accuracy preservation* below is needed to bound the skew between virtual clocks when, for instance, q has synchronized for the i'th time but p has not.

The parameter N is the total number of processors, and F is the maximum number of faulty clocks that the algorithm is expected to tolerate. This property of the system is captured by the following condition.

Condition 8 (bounded faults) *At any time t, the number processors faulty at time t is at most F.*

The conditions below are mathematical constraints placed on the convergence function, e.g., clocks, drifts, and failures, do not play any role in the statements. The isolation of the constraints makes it possible to demonstrate that the egocentric mean function of ICA satisfies the conditions of *translation invariance*, *precision enhancement*, and *accuracy preservation*, in purely mathematical terms. Note that these conditions do not make any distinction between the faulty and the nonfaulty clocks but are instead given in terms of a subset C of clocks satisfying certain mathematical constraints.

Suppose that $t_p^i \geq t_q^i$ for nonfaulty p and q, then in order to compute δ_S, we are interested in comparing the clock times for p and q at t_p^i, the time when clocks p and q have both just been synchronized for the i'th time. Processor q starts its i'th interval clock at t_q^i with value $cfn(q, \Theta_q)$, so that its reading at t_p^i is $cfn(q, \Theta_q) + x$, where $x = PC_q(t_p^i) - PC_q(t_q^i)$. The condition of *translation invariance* indicates that adding x to the value for the various instances of Schneider's protocol. The different choice of parameters do not affect the proof of correctness in any significant way. For individual synchronization protocols, it should be possible to derive one set of parameters from the other.

Condition 3 (bounded interval) *For nonfaulty clock p*

$$0 < r_{min} \leq t_p^{i+1} - t_p^i \leq r_{max} \tag{2.2.11}$$

Condition 4 (bounded delay) *For nonfaulty clocks p and q*[3]

$$|t_q^i - t_p^i| \leq \beta \tag{2.2.12}$$

Condition 5 (initial synchronization) *For nonfaulty clock p*

$$t_p^0 = 0 \tag{2.2.13}$$

From the conditions of *bounded interval* and *bounded delay* above, it follows that if $\beta \leq r_{min}$, then $t_p^i \leq t_q^{i+1}$ for nonfaulty clocks p and q; i.e., there is no overlap between the i'th and the $(i+1)$'th rounds of synchronization. Since we do want the synchronization rounds not to overlap, we state the following as a condition. If the periods were allowed to overlap, then the protocol would be difficult to implement since p could have started its $(i+1)$'th clock before another processor q had started its i'th clock.

Condition 6 (nonoverlap)

$$\beta \leq r_{min} \tag{2.2.14}$$

Another corollary of the *bounded interval* and *bounded delay* conditions is that for any two nonfaulty clocks p and q, we can derive,

$$0 \leq t_p^{i+1} - t_q^i \leq r_{max} + \beta. \tag{2.2.15}$$

For nonfaulty clocks p and q, $\Theta_p^{i+1}(q)$ represents p's observation of q's i'th clock reading at time t_p^{i+1}, i.e., it is p's estimate of $IC_q^i(t_p^{i+1})$. The error

Condition 1 (initial skew) *For nonfaulty processors p and q*

$$|PC_p(0) - PC_q(0)| \leq \delta_S \tag{2.2.8}$$

The nonfaulty physical clocks must keep good enough time so that they do not drift away from real time by a rate greater than ρ.

Condition 2 (bounded drift) *There is a nonnegative constant ρ such that if clock p is nonfaulty at time s, $s \geq t$, then*

$$(1-\rho)(s-t) \leq PC_p(s) - PC_p(t) \leq (1+\rho)(s-t) \tag{2.2.9}$$

A useful corollary to *bounded drift* is that two physical clocks p and q that are not faulty at time s,[2] for $s \geq t$, can drift further apart over the interval $s-t$ by $2\rho(s-t)$, since both p and q can drift by $\rho(s-t)$ with respect to real time, but in opposite directions.

$$|PC_p(s) - PC_q(s)| \leq |PC_p(t) - PC_q(t)| + 2\rho(s-t) \tag{2.2.10}$$

Each protocol has some mechanism for triggering the resynchronization of the clocks. Schneider postulates the existence of a global synchronization signal, t_G^i, which occurs at a period bounded from above and below. One can usually interpret t_G^i as the real-time instant when the first nonfaulty processor decides to resynchronize for the i'th time. Schneider's conditions on t_G^i are stated in terms of positive constants which we name lo, hi, and wid. His first condition is that the period $t_G^{i+1} - t_G^i$ is bounded from below by lo, and from above by hi. The second condition bounds the delay in receiving the trigger so that $t_p^i - t_G^i \leq wid$, for nonfaulty p.

Our description of the proof uses a slightly different set of parameters in order to dispense with the notion of a global synchronization signal used in Schneider's formulation. The parameters below seem easier to identify

[2] In the description of the machine verification, great pains are taken to indicate the times at which the clocks are required to be nonfaulty. The rest of the informal outline of the proof makes the simplifying assumption that clocks are either faulty or nonfaulty, and disregards the time at which clocks are asserted as being nonfaulty.

It is easy to derive the following from Definitions (2.1.1), (2.1.3), and (2.1.4).

$$VC_p(t_p^{i+1}) = IC_p^{i+1}(t_p^{i+1}) = cfn(p, \Theta_p^{i+1}) \tag{2.1.5}$$

$$IC_p^{i+1}(t) = cfn(p, \Theta_p^{i+1}) + PC_p(t) - PC_p(t_p^{i+1}) \tag{2.1.6}$$

So far we have merely defined the virtual and interval clock functions in terms of the physical clock function $PC_p(t)$, the synchronization times t_p^i, and the convergence function *cfn* applied to the clock readings Θ_p^i. In the next section, we enumerate Schneider's constraints on these quantities when p is a *nonfaulty*, or *correct,* processor. The main result we obtain from these constraints and the above definitions is a bound δ on the skew between the logical clocks of two correct processors p and q.

Theorem 2.1.1 (bounded skew) *For any two clocks p and q that are nonfaulty at time t,*

$$|VC_p(t) - VC_q(t)| \leq \delta \tag{2.1.7}$$

The proof of Theorem 2.1.1 is outlined in Section 2.3.1.

2.2 Clock conditions

In formalizing the laws constraining the behavior of individual clocks, we must ensure that no assumptions are made regarding the faulty clocks since we are dealing with Byzantine failures. These laws which are conditions on the behavior of clocks are enumerated as axioms within the boxes below. Individual protocols and clock implementations are expected to satisfy these conditions.

The conditions constraining the behaviour of clocks employ a number of constants represented by lowercase Greek letters. All of these constants are taken to be non-negative.

Section 2.1 above described how the processors go through rounds of synchronization. The proof of Theorem 2.1.7 is by induction on the number of rounds. The main idea of the proof is to show that the virtual clocks are within δ_S immediately following a round of synchronization, and the skew between them does not exceed δ in the following period until the next round of synchronization. To start, the following condition asserts that the nonfaulty clocks are synchronized to within the quantity δ_S at time 0.

field axioms that are used are satisfied by both the real numbers and the rationals. The term $PC_p(t)$ is the reading of p's physical clock at real time t. The adjusted virtual clock reading at time t_p^i is computed by applying an adjustment adj_p^i to the physical clock reading $PC_p(t_p^i)$. In its i'th *interval* of operation, i.e., when $t_p^i \leq t < t_p^{i+1}$, the virtual clock reading, $VC_p(t)$ is given by $PC_p(t) + adj_p^i$. At round 0, the adjustment adj_p^0 is taken to be 0 so that for $t < t_p^1$, the reading $VC_p(t)$ is just $PC_p(t)$. In other words, in the first period of operation, each clock takes its physical clock reading as its virtual clock reading. This means that for synchronization over the first period, we need as a condition, a bound on the initial skews between the physical clocks of nonfaulty processors.

For $i > 0$, we let Θ_p^i be an array of clock readings so that $\Theta_p^i(q)$ is p's reading of q's clock at time t_p^i. In the EHDM formalization, the array of observed clock readings Θ_p^i, is actually represented as a function from clocks to readings. The corrected value of $VC_p(t_p^i)$ is computed by a *convergence function*, $cfn(p, \Theta_p^i)$. The adjustment adj_p^i to be applied to the physical clock is therefore given by the difference $cfn(p, \Theta_p^i) - PC_p(t_p^i)$. Since Θ_p^i is a function, cfn is a higher-order function.

The above explanation of $\Theta_p^i(q)$ does not specify whether q's physical or virtual clock is the one that is read by clock p. Note that if t_q^i preceded t_p^i, then q's virtual clock has already been adjusted for the i'th time at time t_p^i. In Schneider's model, $\Theta_p^i(q)$ is a reading of q's virtual clock at time t_p^i but ignoring the i'th correction that may have already been applied to q's clock. This value is represented by an abstraction called the *interval clock*. The interval clock reading $IC_q^i(t)$ is given by $PC_q(t) + adj_q^i$. Thus for $i > 0$, the value $\Theta_p^i(q)$ is p's reading of $IC_q^{i-1}(t_p^i)$. The rationale for introducing an interval clock is that the observed clock readings in the protocol are based on readings exchanged prior to synchronization. The interval clock is an abstraction that is useful for describing the protocol and it need not actually be implemented. The physical and virtual clocks are of course both implemented.

The above description leads to following definitions where i ranges over the natural numbers and $t > 0$.

$$adj_p^{i+1} = cfn(p, \Theta_p^{i+1}) - PC_p(t_p^{i+1}) \quad (2.1.1)$$

$$adj_p^0 = 0 \quad (2.1.2)$$

$$IC_p^i(t) = PC_p(t) + adj_p^i \quad (2.1.3)$$

$$VC_p(t) = IC_p^i(t), \text{ for } t_p^i \leq t < t_p^{i+1} \quad (2.1.4)$$

Chapter 2

Schneider's Schema for Clock Synchronization

Schneider shows that a number of known algorithms for synchronizing Byzantine clocks can be presented in a uniform manner so that their individual proofs are greatly simplified [1]. The exposition below follows Schneider's outline quite closely, but revises a number of the details in the description of the protocol as well as the proof. Section 2.1 describes how the logical clock is computed from the physical clock using the convergence function. Section 2.2 describes the conditions on the behavior of clocks and on suitable convergence functions. The proof of correctness of clock synchronization from the conditions of Section 2.2 is outlined in Section 2.3.

2.1 Defining Clocks

The physical and logical clocks are presented as functions from real time (as given by some external standard) to clock readings. This real time thus forms the frame of reference and is often referred to simply as "time."[1] The variable t ranges over this real time. Synchronization takes place in *rounds*. The time at which processor p adjusts its clock following the i'th round of synchronization is represented by t_p^i. The starting time t_p^0 which is the time from which the system is observed, is taken to be zero.

In our abstraction, both the real time and the clock readings can be interpreted as ranging over the real numbers or the rationals. The ordered

[1] In the original presentation of the interactive convergence algorithm, clocks are represented as functions from clock time to the external standard time [3, 8].

machine proof clears up some minor inaccuracies in Schneider's derivation of these relations.

Acknowledgements. John Rushby supplied much of the background and guidance for this work. Friedrich von Henke helped me get started with EHDM. I am also grateful to Fred Schneider and Rick Butler for their encouragement.

that these conditions are sufficient for bounding the skew between the resulting logical clocks. He also shows that the convergence functions used by a number of existing protocols satisfy these restrictions. Such a schematic presentation of Byzantine clock synchronization provides an elegant framework for understanding various individual protocols, and greatly simplifies the proofs of their correctness.

Since the SCS protocol captures the mathematics behind Byzantine clock synchronization in an abstract and schematic manner, it makes an interesting candidate for verification. The schematic nature of the SCS protocol makes it convenient to subsequently verify a number of specific protocols as instances of the SCS protocol. Also, Schneider's analysis employs a global "real time" rather than clock time as its frame of reference, i.e., clocks map real time to clock time. Lamport and Melliar-Smith's analysis [3] of ICA and the verification by Rushby and von Henke [8] were both carried out in terms of clocks that mapped clock time to real time. The use of clock time as a frame of reference makes some of the mathematics is fairly cumbersome and also makes the specification harder to understand. It seems reasonable to assume that to each real time instant, there is a unique clock reading, but not quite as reasonable to insist that there is a unique real time instant corresponding to a clock reading since a failed clock could exhibit the same reading at different real time instants. It is, of course, possible to explain away such objections. The question of what is the best framework for specifying such protocols is, to our knowledge, still open.

The mechanical verification of the SCS protocol was carried out using the EHDM verification system developed at the Computer Science Laboratory of SRI International. The egocentric mean function of the ICA protocol was also verified as satisfying Schneider's restrictions. The SCS protocol and its informal proof are presented in Chapter 2. An overview of the mechanically checked proof is presented in Chapter 3. The appendices contain the complete listing of the proof that was presented as input to EHDM.

The use of EHDM to check the proof led to the clarification of a number of details from Schneider's original presentation without tampering unduly with the outline and intent of his argument. Schneider's proof employs a monotonicity condition on convergence functions that was found to be inessential for the proof. The monotonicity condition actually fails for ICA and other similar convergence functions (see Section 2.4). Schneider's proof requires certain relations to hold between the convergence behavior of the convergence function, the drift rate of the physical clocks, the error in communicating clock values, and the time between synchronization rounds. The an hour, δ can be no less than two minutes.

The above outline obviously makes a great many simplifying assumptions, but it does capture the basic process of clock synchronization. The most significant invalid assumption is that clocks and processors do not fail. Clock synchronization protocols ought to be able to tolerate a certain number of processor failures since they are often used to synchronize multiple processors in fault-tolerant architectures. When processors do fail, they could do so in the worst possible way by exhibiting arbitrarily different behaviors towards different processors, e.g., by "maliciously" communicating different clock values to different processors. Such failures are known as *Byzantine* failures [7]. Consider the case of three clocks a, b, and c, when a reads 12 noon, b reads 11:59 am, and c has failed. To resynchronize, they exchange clock values and c maliciously communicates its value as 12:01 pm to a and as 11:58 am to b. Suppose each clock is resynchronized by taking the average of all the clock values observed by it, then a resets itself to 12 noon and b resets itself to 11:59 am. The clocks are thus no closer following resynchronization than immediately prior to resynchronization. Thus the clocks can continue to drift even further apart until the next round of synchronization.

The above scenario illustrates one of the earliest clock synchronization protocols capable of tolerating Byzantine processor failures: the Interactive Convergence Algorithm (ICA) of Lamport and Melliar-Smith [3]. ICA tolerates up to $\lfloor (N-1)/3 \rfloor$ failures for N processors. In ICA, a processor p resynchronizes for the i'th time when its clock reads iR. Processor p then reads the difference between the other clock readings and its own clock reading. By ignoring clock differences larger than a certain value Δ, processor p computes the *egocentric mean* of the acceptable clock differences as the correction required to resynchronize its clock. Rushby and von Henke [8] have subjected Lamport and Melliar-Smith's proof of correctness to mechanical scrutiny using EHDM. As is often the case with fault-tolerant distributed protocols, the original proof is both subtle and complex. The mechanical verification was able to identify and correct several minor flaws, and to significantly streamline the proof.

Schneider [1] presents a clock synchronization scheme that generalizes protocols such as ICA. Schneider's clock synchronization scheme (abbreviated here as SCS) regards each logical clock as being periodically reset to a value computed by a *convergence function.* The egocentric mean of ICA is an instance of such a convergence function. Schneider places certain natural conditions on the behavior of suitable convergence functions and shows with respect to each other and not with respect to some external standard clock. The primary requirement that any solution must satisfy is that at any instant, the absolute difference, or the *skew*, between two clock readings should be within some bound δ. The secondary requirement is that there must be a small bound on the correction required to keep clocks in synchrony. The latter requirement prevents trivial solutions that, for example, reset the clocks to zero at each round of synchronization. We restrict our focus to the primary requirement, since the secondary requirement turns out to be a straightforward consequence of one of the assumptions for the operation of the protocol studied here.

To implement synchronized clocks, each processor has a *physical* clock whose drift rate with respect to a fixed standard time is bounded. We refer to the fixed standard time as *real time*. In addition to the physical clock, each processor maintains a *logical,* or *virtual,* clock that is computed by periodically applying an adjustment to the reading of the physical clock. The adjustment to be applied at the end of each period is determined by means of a synchronization protocol. The application of such an adjustment could be *continuous* so that the individual clock ticks are either sped up or slowed down, but no clock ticks are dropped or repeated. Alternately, the adjustment could be applied in an *instantaneous* manner, in which case, some clock ticks might be dropped or repeated. In the latter situation, critical events should not be scheduled during these clock ticks. This report only considers the case of instantaneous clock adjustments. These results are therefore applicable to the class of systems that have a synchronization phase followed by a period of normal operation in each cycle of synchronization. The results here can be extended to the case of continuous clock adjustments. Schneider [1] presents an analysis of continuous adjustments.

To take a somewhat coarse look at clock synchronization, suppose that the various physical clocks start synchronized and drift apart from real time at a rate not exceeding ρ. For example, a clock might gain or lose up to a minute every hour. The processors operate normally for a period R of, say, an hour. The processors then engage in a round of synchronization during which they exchange clock values. Assume for simplicity that the communication between clocks occurs instantaneously. At some mutually agreeable instant, the processors reset their clocks to some mutually agreeable value such as the average of their clock readings. Thus at the end of such a round of synchronization, the skew between clocks vanishes. Clearly, if we want the clocks to be no more than δ apart, the period R between synchronizations should not exceed $\delta/2\rho$. Given that ρ is a minute per hour, and R is

Chapter 1

Introduction

Synchronizing clocks in the presence of faults is a classic problem in distributed computing. Even the most accurate clocks do drift at significant rates, both with respect to a time standard and relative to each other. In order for independent processors to exhibit cooperative behavior, it is often required that their local clocks be synchronized. Such synchrony is the basis for distributed algorithms that use timeouts, time stamps, and rounds of message passing. Synchronization is also assumed when the same computation is executed on multiple, independent processors in order to mask processor failures. Digital avionics systems constitute a typical example of the need for synchronized clocks. In these systems, the results of multiple redundant processors are voted to ensure a high degree of fault tolerance, and the processor clocks must be synchronized in order to carry this out. Clock synchronization problems led to the scrubbing of the first scheduled launch of the NASA Space Shuttle [4], and to anomalous behavior of the Voyager spacecraft [5]. Butler [6] presents a survey of various clock synchronization protocols.

Synchronizing clocks in the presence of faults is a difficult problem. If synchrony is maintained by periodically broadcasting a global clock value to each of the processors, the failure of the global clock then becomes critical. On the other hand, if each processor has its own local clock and these clocks are initially synchronized, they might slowly drift apart so that with time the system loses its ability to behave synchronously. It is therefore necessary to periodically resynchronize the clocks. We are concerned here with algorithms that perform this resynchronization in a fault tolerant manner. In the cases we consider, the clocks are required to be synchronized only

List of Figures

Contents

Abstract

Schneider [1] generalizes a number of protocols for Byzantine fault tolerant clock synchronization and presents a uniform proof for their correctness. We present a machine checked proof of this schematic protocol that revises some of the details in Schneider's original analysis. The verification was carried out with the EHDM system [2] developed at the SRI Computer Science Laboratory. The mechanically checked proofs include the verification that the *egocentric mean* function used in Lamport and Melliar-Smith's Interactive Convergence Algorithm [3] satisfies the requirements of Schneider's protocol.

Mechanical Verification of a Schematic Protocol for Byzantine Fault Tolerant Clock Synchronization

Natarajan Shankar
Computer Science Laboratory
SRI International
333 Ravenswood
Menlo Park, CA 94025

June 5, 1991