```
section ‹2PL+2PC Refinement Proof Invariants (and important lemmas)›

theory Serializable_2PC_2PL_Invariants
  imports Serializable_2PC_2PL
begin

— ‹Invariant about future and past transactions kms›

definition TIDFutureKm where
  "TIDFutureKm s cl ⟷ (∀n k. n > tm_sn (tm s cl) ⟶ km_status (kms s k) (Tn_cl n cl) = working)"

definition TIDPastKm where
  "TIDPastKm s cl ⟷ (∀n k. n < tm_sn (tm s cl) ⟶ km_status (kms s k) (Tn_cl n cl) ∈ {committed, aborted})"

lemma other_sn_idle:
  assumes "TIDFutureKm s cl" and "TIDPastKm s cl"
    and "get_cl_txn t = cl" and "get_sn_txn t ≠ tm_sn (tm s cl)"
  shows "⋀k. km_status (kms s k) t ∈ {working, committed, aborted}"


— ‹Lock Invariants›

definition RLockInv where
  "RLockInv s k ⟷ (∀t. km_status (kms s k) t = read_lock ⟶ (∀t. km_status (kms s k) t ≠ write_lock))"

definition WLockInv where
  "WLockInv s k ⟷ (∀t. km_status (kms s k) t ≠ write_lock) ∨ (∃! t. km_status (kms s k) t = write_lock)"

— ‹Invariants for fingerprint, knowing the lock (km status)›

definition RLockFpInv where
  "RLockFpInv s k ⟷ (∀t. km_status (kms s k) t = read_lock ⟶
    km_key_fp (kms s k) t W = None ∧
    km_key_fp (kms s k) t R ≠ None)"

definition WLockFpInv where
  "WLockFpInv s k ⟷ (∀t. km_status (kms s k) t = write_lock ⟶ km_key_fp (kms s k) t W ≠ None)"

definition NoLockFpInv where
  "NoLockFpInv s k ⟷ (∀t. km_status (kms s k) t = no_lock ⟶
    km_key_fp (kms s k) t W = None ∧
    km_key_fp (kms s k) t R = None)"

— ‹Invariants about kv store›
definition KVSNonEmp where
  "KVSNonEmp s ⟷ (∀k. km_vl (kms s k) ≠ [])"

definition KVSGSNonEmp where
  "KVSGSNonEmp s ⟷ (∀k. kvs_of_gs s k ≠ [])"

definition KVSLen where
  "KVSLen s cl ⟷ (∀k. length (km_vl (kms s k)) ≤ length (kvs_of_gs s k))"

subsubsection ‹Lemmas for kvs_of_gs changing by different events›

lemma kvs_of_gs_km_inv:
  assumes "WLockInv s k" and "RLockInv s k"
    and "(∀t. km_status (kms s k) t ≠ write_lock) ∨
              km_status (kms s' k) t ≠ write_lock"
    and "tm_status (tm s (get_cl_txn t)) ≠ tm_committed"
    and "⋀k. km_vl (kms s' k) = km_vl (kms s k)"
    and "tm_km_k'_t'_unchanged k s s' t"
  shows "kvs_of_gs s' = kvs_of_gs s"


lemma kvs_of_gs_tm_inv:
  assumes "TIDFutureKm s cl" and "TIDPastKm s cl"
    and "tm_status (tm s cl) ≠ tm_committed ∨
         (∀k. km_status (kms s k) (get_txn_cl cl s) = committed)"
    and "tm_status (tm s' cl) ≠ tm_committed"
    and "km_tm_cl'_unchanged cl s s'"
  shows "kvs_of_gs s' = kvs_of_gs s"
```

```
lemma update_kv_all_tm_commit_no_lock_inv:
  assumes "TIDPastKm s cl" and "TIDFutureKm s cl"
    and "tm_status (tm s cl) = tm_prepared"
    and "tm_status (tm s' cl) = tm_committed"
    and "other_insts_unchanged cl (tm s) (tm s')"
    and "km_status (kms s k) (get_txn_cl cl s) = no_lock"
  shows "update_kv_all_txn (λt. tm_status (tm s' (get_cl_txn t))) (km_status (kms s k))
          (km_key_fp (kms s k)) (km_vl (kms s k)) =
         update_kv_all_txn (λt. tm_status (tm s (get_cl_txn t))) (km_status (kms s k))
          (km_key_fp (kms s k)) (km_vl (kms s k))"


(*All events*)
abbreviation not_tm_commit where
  "not_tm_commit e ≡ ∀cl sn u F. e ≠ TM_Commit cl sn u F"

abbreviation invariant_list_kvs where
  "invariant_list_kvs s ≡ ∀cl k. TIDFutureKm s cl ∧ TIDPastKm s cl ∧ RLockInv s k ∧ WLockInv s k ∧
                          RLockFpInv s k ∧ NoLockFpInv s k ∧ KVSNonEmp s"

lemma kvs_of_gs_inv:
  assumes "gs_trans s e s'"
    and "invariant_list_kvs s"
    and "not_tm_commit e"
  shows "kvs_of_gs s' = kvs_of_gs s"


— ‹More specific lemmas about TM commit›
lemma kvs_of_gs_commit_length_increasing:
  assumes "tm_status (tm s cl) = tm_prepared"
    and "tm_status (tm s' cl) = tm_committed"
    and "km_tm_cl'_unchanged cl s s'"
  shows "length (kvs_of_gs s k) ≤ length (kvs_of_gs s' k)"


lemma kvs_of_gs_length_increasing:
  assumes "gs_trans s e s'"
    and "invariant_list_kvs s"
  shows "⋀k. length (kvs_of_gs s k) ≤ length (kvs_of_gs s' k)"


— ‹Fingerprint content invariant and Lemmas for proving the fp_property›

lemma km_vl_read_lock_commit_eq_length:
  assumes "RLockFpInv s k"
    and "km_status (kms s k) t = read_lock"
    and "km_vl (kms s' k) =
          update_kv_key t (km_key_fp (kms s k) t) (full_view (km_vl (kms s k))) (km_vl (kms s k))"
  shows "length (km_vl (kms s' k)) = length (km_vl (kms s k))"


definition RLockFpContentInv where
  "RLockFpContentInv s k ⟷ (∀t. km_status (kms s k) t = read_lock ⟶
    km_key_fp (kms s k) t R =
      Some (v_value (last_version (km_vl (kms s k)) (full_view (km_vl (kms s k))))))"

definition WLockFpContentInv where
  "WLockFpContentInv s k ⟷ (∀t. km_status (kms s k) t = write_lock ⟶
    km_key_fp (kms s k) t R = None ∨
    km_key_fp (kms s k) t R =
      Some (v_value (last_version (km_vl (kms s k)) (full_view (km_vl (kms s k))))))"

lemma km_vl_kvs_eq_length:
  assumes "WLockInv s k" and "RLockInv s k"
    and "tm_status (tm s cl) = tm_prepared"
    and "km_status (kms s k) (get_txn_cl cl s) ∈ {read_lock, write_lock}"
  shows "length (kvs_of_gs s k) = length (km_vl (kms s k))"


— ‹Lemmas for view growth after commit›

lemma committed_kvs_view_grows:
  assumes "tm_status (tm s cl) = tm_prepared"
    and "tm_status (tm s' cl) = tm_committed"
```

```
    and "km_tm_cl'_unchanged cl s s'"
  shows "(λk. full_view (kvs_of_gs s k)) ⊑ (λk. full_view (kvs_of_gs s' k))"


lemma updated_vl_view_grows:
  assumes "km_vl (kms s' k) =
    update_kv_key t (km_key_fp (kms s k) t) (full_view (km_vl (kms s k))) (km_vl (kms s k))"
    and "other_insts_unchanged k (kms s) (kms s')"
  shows "(λk. full_view (km_vl (kms s k))) ⊑ (λk. full_view (km_vl (kms s' k)))"


lemma tm_view_inv:
  assumes "gs_trans s e s'"
    and "not_tm_commit e"
  shows "tm_view (tm s' cl) = tm_view (tm s cl)"


definition TMFullView where
  "TMFullView s cl ⟷ tm_view (tm s cl) ⊑ (λk. full_view (kvs_of_gs s k))"
```

— ‹TM_commit updating kv›

```
lemma kvs_of_gs_tm_commit:
  assumes "TIDFutureKm s cl" and "TIDPastKm s cl"
    and "WLockInv s k" and "WLockFpInv s k"
    and "RLockInv s k" and "RLockFpInv s k"
    and "NoLockFpInv s k" and "KVSNonEmp s"
    and "tm_status (tm s cl) = tm_prepared"
    and "km_status (kms s k) (get_txn_cl cl s) ∈ {read_lock, write_lock, no_lock}"
    and "tm_status (tm s' cl) = tm_committed"
    and "other_insts_unchanged cl (tm s) (tm s')"
  shows "update_kv_all_txn (λt. tm_status (tm s' (get_cl_txn t))) (km_status (kms s k))
          (km_key_fp (kms s k)) (km_vl (kms s k)) =
    update_kv_key (get_txn_cl cl s) (km_key_fp (kms s k) (get_txn_cl cl s))
      (full_view (update_kv_all_txn (λt. tm_status (tm s (get_cl_txn t))) (km_status (kms s k))
        (km_key_fp (kms s k)) (km_vl (kms s k))))
      (update_kv_all_txn (λt. tm_status (tm s (get_cl_txn t))) (km_status (kms s k))
        (km_key_fp (kms s k)) (km_vl (kms s k)))"
```

— ‹Lemmas for showing transaction id freshness›

```
lemma get_sqns_other_cl_inv:
  assumes "TIDFutureKm s cl" and "TIDPastKm s cl"
    and "⋀k. WLockInv s k" and "⋀k. WLockFpInv s k"
    and "⋀k. RLockInv s k" and "⋀k. RLockFpInv s k"
    and "⋀k. NoLockFpInv s k" and "KVSNonEmp s"
    and "tm_status (tm s cl) = tm_prepared"
    and "tm_status (tm s' cl) = tm_committed"
    and "⋀k. km_status (kms s k) (get_txn_cl cl s) = read_lock ∨
             km_status (kms s k) (get_txn_cl cl s) = write_lock ∨
             km_status (kms s k) (get_txn_cl cl s) = no_lock"
    and "km_tm_cl'_unchanged cl s s'"
    and "cl' ≠ cl"
  shows "get_sqns (kvs_of_gs s') cl' = get_sqns (kvs_of_gs s) cl'"


lemma new_t_is_in_writers:
  assumes "TIDFutureKm s cl" and "TIDPastKm s cl"
    and "⋀k. WLockInv s k" and "⋀k. WLockFpInv s k"
    and "⋀k. RLockInv s k" and "⋀k. RLockFpInv s k"
    and "⋀k. NoLockFpInv s k" and "KVSNonEmp s"
    and "tm_status (tm s cl) = tm_prepared"
    and "tm_status (tm s' cl) = tm_committed"
    and "km_status (kms s k) (get_txn_cl cl s) = write_lock"
    and "other_insts_unchanged cl (tm s) (tm s')"
    and "kms s' = kms s"
  shows "vl_writers_sqns (kvs_of_gs s' k) cl = vl_writers_sqns (kvs_of_gs s k) cl ∪ {tm_sn (tm s cl)}"


lemma new_t_is_in_writers2:
  assumes "TIDFutureKm s cl" and "TIDPastKm s cl"
    and "⋀k. WLockInv s k" and "⋀k. WLockFpInv s k"
    and "⋀k. RLockInv s k" and "⋀k. RLockFpInv s k"
    and "⋀k. NoLockFpInv s k" and "KVSNonEmp s"
```

```
    and "tm_status (tm s cl) = tm_prepared"
    and "tm_status (tm s' cl) = tm_committed"
    and "km_status (kms s k) (get_txn_cl cl s) = read_lock"
    and "other_insts_unchanged cl (tm s) (tm s')"
    and "kms s' = kms s"
  shows "vl_writers_sqns (kvs_of_gs s' k) cl = vl_writers_sqns (kvs_of_gs s k) cl"


lemma new_t_is_in_readers:
  assumes "TIDFutureKm s cl" and "TIDPastKm s cl"
    and "⋀k. WLockInv s k" and "⋀k. WLockFpInv s k"
    and "⋀k. RLockInv s k" and "⋀k. RLockFpInv s k"
    and "⋀k. NoLockFpInv s k" and "KVSNonEmp s"
    and "tm_status (tm s cl) = tm_prepared"
    and "tm_status (tm s' cl) = tm_committed"
    and "km_status (kms s k) (get_txn_cl cl s) = read_lock"
    and "other_insts_unchanged cl (tm s) (tm s')"
    and "kms s' = kms s"
  shows "vl_readers_sqns (kvs_of_gs s' k) cl = vl_readers_sqns (kvs_of_gs s k) cl ∪ {tm_sn (tm s cl)}"


lemma new_t_is_in_readers2:
  assumes "TIDFutureKm s cl" and "TIDPastKm s cl"
    and "⋀k. WLockInv s k" and "⋀k. WLockFpInv s k"
    and "⋀k. RLockInv s k" and "⋀k. RLockFpInv s k"
    and "⋀k. NoLockFpInv s k" and "KVSNonEmp s"
    and "tm_status (tm s cl) = tm_prepared"
    and "tm_status (tm s' cl) = tm_committed"
    and "km_status (kms s k) (get_txn_cl cl s) = write_lock"
    and "km_key_fp (kms s k) (get_txn_cl cl s) R ≠ None"
    and "other_insts_unchanged cl (tm s) (tm s')"
    and "kms s' = kms s"
  shows "vl_readers_sqns (kvs_of_gs s' k) cl = vl_readers_sqns (kvs_of_gs s k) cl ∪ {tm_sn (tm s cl)}"


lemma new_t_is_in_readers3:
  assumes "TIDFutureKm s cl" and "TIDPastKm s cl"
    and "⋀k. WLockInv s k" and "⋀k. WLockFpInv s k"
    and "⋀k. RLockInv s k" and "⋀k. RLockFpInv s k"
    and "⋀k. NoLockFpInv s k" and "KVSNonEmp s"
    and "tm_status (tm s cl) = tm_prepared"
    and "tm_status (tm s' cl) = tm_committed"
    and "km_status (kms s k) (get_txn_cl cl s) = write_lock"
    and "km_key_fp (kms s k) (get_txn_cl cl s) R = None"
    and "other_insts_unchanged cl (tm s) (tm s')"
    and "kms s' = kms s"
  shows "vl_readers_sqns (kvs_of_gs s' k) cl = vl_readers_sqns (kvs_of_gs s k) cl"


lemma kvs_writers_tm_commit_grows:
  assumes "TIDFutureKm s cl" and "TIDPastKm s cl"
    and "⋀k. WLockInv s k" and "⋀k. WLockFpInv s k"
    and "⋀k. RLockInv s k" and "⋀k. RLockFpInv s k"
    and "⋀k. NoLockFpInv s k" and "KVSNonEmp s"
    and "tm_status (tm s cl) = tm_prepared"
    and "tm_status (tm s' cl) = tm_committed"
    and "∀k. km_status (kms s k) (get_txn_cl cl s) ∈ {read_lock, write_lock, no_lock}"
    and "km_status (kms s k) (get_txn_cl cl s) = write_lock"
    and "other_insts_unchanged cl (tm s) (tm s')"
    and "kms s' = kms s"
  shows "kvs_writers_sqns (kvs_of_gs s') cl = kvs_writers_sqns (kvs_of_gs s) cl ∪ {tm_sn (tm s cl)}"


lemma kvs_writers_tm_commit_doesnt_grow:
  assumes "TIDFutureKm s cl" and "TIDPastKm s cl"
    and "⋀k. WLockInv s k" and "⋀k. WLockFpInv s k"
    and "⋀k. RLockInv s k" and "⋀k. RLockFpInv s k"
    and "⋀k. NoLockFpInv s k" and "KVSNonEmp s"
    and "tm_status (tm s cl) = tm_prepared"
    and "tm_status (tm s' cl) = tm_committed"
    and "∀k. km_status (kms s k) (get_txn_cl cl s) ∈ {read_lock, no_lock}"
    and "other_insts_unchanged cl (tm s) (tm s')"
    and "kms s' = kms s"
  shows "kvs_writers_sqns (kvs_of_gs s') cl = kvs_writers_sqns (kvs_of_gs s) cl"
```

```
lemma kvs_readers_sqns_tm_commit_grows:
  assumes "TIDFutureKm s cl" and "TIDPastKm s cl"
    and "⋀k. WLockInv s k" and "⋀k. WLockFpInv s k"
    and "⋀k. RLockInv s k" and "⋀k. RLockFpInv s k"
    and "⋀k. NoLockFpInv s k" and "KVSNonEmp s"
    and "tm_status (tm s cl) = tm_prepared"
    and "tm_status (tm s' cl) = tm_committed"
    and "∀k. km_status (kms s k) (get_txn_cl cl s) ∈ {read_lock, write_lock, no_lock}"
    and "km_status (kms s k) (get_txn_cl cl s) = read_lock ∨
         (km_status (kms s k) (get_txn_cl cl s) = write_lock ∧
          km_key_fp (kms s k) (get_txn_cl cl s) R ≠ None)"
    and "other_insts_unchanged cl (tm s) (tm s')"
    and "kms s' = kms s"
  shows "kvs_readers_sqns (kvs_of_gs s') cl = kvs_readers_sqns (kvs_of_gs s) cl ∪ {tm_sn (tm s cl)}"


lemma kvs_readers_sqns_tm_commit_doesnt_grow:
  assumes "TIDFutureKm s cl" and "TIDPastKm s cl"
    and "⋀k. WLockInv s k" and "⋀k. WLockFpInv s k"
    and "⋀k. RLockInv s k" and "⋀k. RLockFpInv s k"
    and "⋀k. NoLockFpInv s k" and "KVSNonEmp s"
    and "tm_status (tm s cl) = tm_prepared"
    and "tm_status (tm s' cl) = tm_committed"
    and "∀k. km_status (kms s k) (get_txn_cl cl s) ∈ {write_lock, no_lock}"
    and "∀k. km_status (kms s k) (get_txn_cl cl s) ≠ write_lock ∨
              km_key_fp (kms s k) (get_txn_cl cl s) R = None"
    and "other_insts_unchanged cl (tm s) (tm s')"
    and "kms s' = kms s"
  shows "kvs_readers_sqns (kvs_of_gs s') cl = kvs_readers_sqns (kvs_of_gs s) cl"


lemma get_sqns_tm_commit_grows:
  assumes "TIDFutureKm s cl" and "TIDPastKm s cl"
    and "⋀k. WLockInv s k" and "⋀k. WLockFpInv s k"
    and "⋀k. RLockInv s k" and "⋀k. RLockFpInv s k"
    and "⋀k. NoLockFpInv s k" and "KVSNonEmp s"
    and "tm_status (tm s cl) = tm_prepared"
    and "tm_status (tm s' cl) = tm_committed"
    and "⋀k. km_status (kms s k) (get_txn_cl cl s) ∈ {read_lock, write_lock, no_lock}"
    and "other_insts_unchanged cl (tm s) (tm s')"
    and "kms s' = kms s"
  shows "get_sqns (kvs_of_gs s') cl =
         (if ∀k. km_status (kms s k) (get_txn_cl cl s) = no_lock then
          get_sqns (kvs_of_gs s) cl else
          get_sqns (kvs_of_gs s) cl ∪ {tm_sn (tm s cl)})"


definition SqnInv where
  "SqnInv s cl ⟷
    (tm_status (tm s cl) ≠ tm_committed ⟶ (∀m ∈ get_sqns (kvs_of_gs s) cl. m < tm_sn (tm s cl))) ∧
    (tm_status (tm s cl) = tm_committed ⟶ (∀m ∈ get_sqns (kvs_of_gs s) cl. m ≤ tm_sn (tm s cl)))"


— ‹Lemmas for proving view wellformedness of tm_view›

lemma kvs_of_gs_version_order:
  assumes "TIDPastKm s cl" and "TIDFutureKm s cl" and "WLockInv s k" and "RLockInv s k" and "KVSNonEmp s"
    and "i ∈ full_view (kvs_of_gs s k)"
    and "tm_status (tm s cl) = tm_prepared"
    and "tm_status (tm s' cl) = tm_committed"
    and "km_status (kms s k) (get_txn_cl cl s) ∈ {read_lock, write_lock, no_lock}"
    and "km_tm_cl'_unchanged cl s s'"
  shows "kvs_of_gs s k ! i ⊑ver kvs_of_gs s' k ! i"


lemma new_version_index:
  assumes "TIDPastKm s cl" and "TIDFutureKm s cl"
    and "WLockInv s k" and "KVSNonEmp s"
    and "tm_status (tm s cl) = tm_prepared"
    and "tm_status (tm s' cl) = tm_committed"
    and "km_status (kms s k) (get_txn_cl cl s) = write_lock"
    and "other_insts_unchanged cl (tm s) (tm s')"
    and "i ∈ full_view (update_kv_all_txn (λt. tm_status (tm s' (get_cl_txn t)))
    (km_status (kms s k)) (km_key_fp (kms s k)) (km_vl (kms s k)))"
```

```
    and "i ∉ full_view (update_kv_all_txn (λt. tm_status (tm s (get_cl_txn t)))
    (km_status (kms s k)) (km_key_fp (kms s k)) (km_vl (kms s k)))"
  shows "i = length (km_vl (kms s k))"


lemma t_is_fresh:
  assumes "SqnInv s cl"
    and "tm_status (tm s cl) = tm_prepared"
  shows "get_txn_cl cl s ∈ next_txids (kvs_of_gs s) cl"


lemma kvs_of_gs_view_atomic:
  assumes "TIDPastKm s cl" and "TIDFutureKm s cl"
    and "⋀k. WLockInv s k" and "⋀k. WLockFpInv s k"
    and "⋀k. RLockInv s k" and "⋀k. NoLockFpInv s k"
    and "SqnInv s cl" and "KVSNonEmp s"
    and "tm_status (tm s cl) = tm_prepared"
    and "tm_status (tm s' cl) = tm_committed"
    and "∀k. km_status (kms s k) (get_txn_cl cl s) ∈ {read_lock, write_lock, no_lock}"
    and "km_tm_cl'_unchanged cl s s'"
  shows "view_atomic (kvs_of_gs s') (λk. full_view (kvs_of_gs s k))"


lemma reach_kvs_expands [simp, intro]:
  assumes "reach tps s" and "gs_trans s e s'"
    and "⋀cl. TIDFutureKm s cl" and "⋀cl. TIDPastKm s cl"
    and "⋀k. RLockInv s k" and "⋀k. WLockInv s k"
    and "⋀k. RLockFpInv s k" and "⋀k. NoLockFpInv s k"
    and "KVSNonEmp s" and "KVSLen s cl"
  shows "kvs_of_gs s ⊑kvs kvs_of_gs s'"


definition KVSView where
  "KVSView s cl ⟷ view_wellformed (kvs_of_gs s) (tm_view (tm s cl))"

— ‹CanCommit›

lemma writers_visible:
  assumes "u = (λk. full_view (K k))"
  shows "visTx K u = kvs_writers K"


lemma WW_writers_id_helper:
  assumes "(x, v_writer x') ∈ {(xa, x). ∃xb i.
            i ∈ full_view (K xb) ∧
            (∃i'. i' ∈ full_view (K xb) ∧
              x = v_writer (K xb ! i) ∧ xa = v_writer (K xb ! i') ∧ i < i')}* "
    and "x' ∈ set (K k)"
  shows "∃xa. x ∈ v_writer ` set (K xa)"


lemma WW_writers_id:
  "((((⋃ (range (WW K)))⁻¹)*)⁻¹ `` kvs_writers K = kvs_writers K"


lemma full_view_satisfies_ET_SER_canCommit:
  "u = (λk. full_view (K k)) ⟹ ET_SER.canCommit K u F"


end
```