1　Sheeran らの方法 1

$$\Big((\neg\exists s_{[0..k]}.\, I(s_0) \wedge loopFree(s_{[0..k]})) \vee (\neg\exists s_{[0..k]}.\, loopFree(s_{[0..k]}) \vee \neg P(s_k)) \Big) \wedge \bigwedge_{0\le i\le k} \big(\neg\exists s_{[0..i]}.\, I(s_0) \wedge path(s_{[0..i]}) \wedge \neg P(s_i)\big) \quad (1)$$

1.1　証明の概要

ここでは, Sheeran らの方法 1 の健全性を証明する. つぎに, システム安全であるとき以下の条件式が成り立つ.

$$\forall i.\, \forall s_{[0..i]}.\, \neg(I(s_0) \wedge path(s_{[0..i]}) \wedge \neg P(s_i)) \quad (2)$$

さらに, 小さい i から順に検査していく場合, 閉路を含む実行パスの到達する先はすでに検証済みであることを考慮に入れると式 (2) は以下の式と等価と言える.

$$\forall i.\, \forall s_{[0...i]}.\, \neg(I(s_0) \wedge loopFree(s_{[0..k]}) \wedge \neg P(s_i)) \quad (3)$$

この式は, 初期状態 s_0 から性質 P を満たさない状態 s_i に到達するような閉路を含まない実行パスは, 存在しないことを表す. 従って, 「式 (1) ならば, 式 (3) が成り立つ」を証明すれば, Sheeran らの方法 1 の健全性を示せたことになる.

以下, 証明の流れを述べる. まず, $i < k$ と $i \ge k$ に場合分けを行い, それぞれ

$$\forall s_{[0..i]}.\, \neg(I(s_0) \wedge loopFree(s_{[0..i]}) \wedge \neg P(s_i)) \quad (4)$$

が成り立つことを証明する.

$i < k$ の場合, 式 (1) の 3 行目より, k ステップ目まで安全であることが分かっているので, 式 (9) が成り立つ.

つぎに, $i \ge k$ の場合について証明する. 式 (1) より, (a) $\forall s_{[0..k]}.\, \neg(I(s_0) \wedge loopFree(s_{[0..k]}))$ (b) $\forall s_{[0..k]}.\, \neg(loopFree(s_{[0..k]}) \wedge \neg P(s_k))$ のどちらかが成り立つことがわかっている. ここで, 閉路を含まない実行パス $loopFree(s_{[0..i]})$ は分割できることを利用すると, 式 (3) の部分式 $I(s_0) \wedge loopFree(s_{[0..i]}) \wedge \neg P(s_i)$ は以下のように書くことができる.

$$I(s_0) \wedge loopFree(s_{[0..k]}) \wedge loopFree(s_{[k..i]}) \wedge \neg P(s_i) \quad (5)$$

式 (5) は, (a) が成り立つとき否定されるので, 式 (9) が成り立つ. また, $I(s_0) \wedge loopFree(s_{[0..i]}) \wedge \neg P(s_i)$ は, 以下のように分割することもできる.

$$I(s_0) \wedge loopFree(s_{[0..(i-k)]}) \wedge loopFree(s_{[(i-k)..i]}) \wedge \neg\, P(s_i) \quad (6)$$

そして, $i \ge k$ の場合, (b) は以下のように考えて良い.

$$\Big(\forall s_{[0..k]}.\, \neg(loopFree(s_{[0..k]}) \wedge \neg P(s_k))\Big) \Leftrightarrow \Big(\forall s_{[(i-k)..i]}.\, \neg(loopFree(s_{[(i-k)..i]}) \wedge \neg P(s_i))\Big) \quad (7)$$

(b) が成り立つときは, 式 (6) が否定されるので, 式 (9) が成り立つ. よって, $i \le k$ の場合も, 式 (9) が成り立つ.

以上により, すべての i について式 (9) が成り立つので, Sheeran らのエンコード方法 1 の健全性が示された.

1.2 Coq 上での証明

```
Threorem Proof_Sheeran_method1
 ├─ Lemma Proof_Sheeran_method1_case1
 │   └─ Lemma case1_t1
 │       └─ Lemma lt_big_and_incl
 └─ Lemma Proof_Sheeran_method1_case2
     ├─ Lemma divide_loop_free
     │   ├─ Lemma divide_path
     │   │   └─ Lemma shift_path
     │   │       ├─ Lemma divide_tl_path
     │   │       └─ Lemma divide_hd_path
     │   └─ Lemma divide_loop_check
     │       ├─ Lemma divide_lc1
     │       └─ Lemma divide_lc2
     │           └─ Lemma divide_lc2'
     │               └─ Lemma divide_lc2''
     └─ Lemma case2_t1 (Admitted!)
```

※下に証明の概要を書く.

```
Theorem Proof_Sheeran_method1 :
  ∀ (k : ℕ) (I : init) (T : trans) (P : property) (l : list ℕ),
     Sheeran_method1 I T P l k → (∀ (i : ℕ) (s : ss), ¬ (I (s 0) ∧ loop_free T s 0 i l ∧ ¬ P (s i))).
Proof.
  intros.
  assert (H0 : i < k ∨ i >= k).
  omega.
  destruct H0.
  - revert H0.  (* i < k → ¬(I (s 0) ∧ loop_free T s 0 i l ∧ ¬P (s i))). *)
    apply Proof_Sheeran_method1_case1.
    tauto.
  - revert H0.  (* i >= k → ¬(I (s 0) ∧ loop_free T s 0 i l ∧ ¬P (s i))). *)
    apply Proof_Sheeran_method1_case2.
    tauto.
Qed.
```

まず, Proof_Sheeran_method1_case1 を証明する.

```
Lemma case1_t1 : ∀ (i k : ℕ) (s : ss) (I : init) (T : trans) (P : property),
  (i < k) ∧ kth_P_safe I T P k → ¬ (I (s 0) ∧ path T s 0 i ∧ ¬ P (s i)).

Theorem Proof_Sheeran_method1_case1 :
  ∀ (k : ℕ) (I : init) (T : trans) (P : property) (l : list ℕ),
    Sheeran_method1 I T P l k
    → (∀ (i : ℕ) (s : ss), (i < k)%ℕ → ¬(I (s 0) ∧ loop_free T s 0 i l ∧ ¬ P (s i))).
Proof.
  unfold Sheeran_method1.
  intros.
  assert((i < k) ∧ kth_P_safe I T P k).
  tauto.
  apply case1_t1 with (s := s) in H1.
  unfold loop_free.
  tauto.
Qed.
```

case1_t1 を apply するために, assert を使って, 式変形を行っているだけ. kth_P_safe は, 式 (1) の 2 行目の部分を表している.

次に, Proof_Sheeran_method1_case2 を証明する.

```
Lemma case2_t1 : ∀ (i k : ℕ) (T : trans) (P : property) (l : list ℕ),
  (i >= k)%ℕ →
  (∀ s1 : ss, ¬ (loop_free T s1 (i−k) k l ∧ ¬ P (s1 i))) ↔
  (∀ s2 : ss, ¬ (loop_free T s2 0 k l ∧ ¬ P (s2 k))).
 Proof. Admitted.

Theorem divide_loop_free : ∀ (i j: ℕ) (s : ss) (T : trans) (l : list ℕ),
  loop_free T s 0 (i+j) l → loop_free T s 0 i l ∧ loop_free T s i j l.

Theorem Proof_Sheeran_method1_case2 :
  ∀ (k : ℕ) (I : init) (T : trans) (P : property) (l : list ℕ),
    Sheeran_method1 I T P l k →
    (∀ (i : ℕ) (s : ss), (i >= k)%ℕ → ¬(I (s 0) ∧ loop_free T s 0 i l ∧ ¬ P (s i))).
 Proof.
  unfold Sheeran_method1.
  intros.
  apply neg_false. (*結論部分を I (s 0) ∧ loop_free T s 0 i l ∧ ¬P (s i)) ↔ False に変形*)
  split. (* → と ← に分ける. *)
  − intros.
    destruct H.
    destruct H.
    + assert (H3 : i = k + (i − k)). omega.
      unfold lasso in H.
      destruct H1.
      destruct H4.
```

```
    rewrite H3 in H4.
    apply divide_loop_free in H4.
    firstorder.
  + unfold violate_loop_free in H.
    simpl in H.
    destruct H1.
    destruct H3.
    assert (H5 : i = i - k + k). omega.
    rewrite H5 in H3.
    apply divide_loop_free in H3.
    apply (case2_t1 i k T P l) in H0.
    rewrite <- H0 in H.
    firstorder.
- tauto.
Qed.
```

---

case2_t1 と divide_loop_free を使用して, 証明した. この証明は, 前節の $k \leq i$ の場合の証明と同じ流れで証明している. case2_t1 は, まだ未証明.

以下, 使用した補題一覧.

```
Lemma divide_path : ∀ (i j: ℕ) (s : ss) (T : trans),
 path T s 0 (i+j) → path T s 0 i ∧ path T s i j.

Lemma divide_loop_check : ∀ (i j: ℕ) (s : ss) (l : list ℕ),
 loop_check s 0 (i+j) l → loop_check s 0 i l ∧ loop_check s i j l.

Lemma divide_loop_free : ∀ (i j: ℕ) (s : ss) (T : trans) (l : list ℕ),
  loop_free T s 0 (i+j) l → loop_free T s 0 i l ∧ loop_free T s i j l.

Lemma divide_tl_path : ∀ (i : ℕ) (s : ss) (T : trans),
  path T s 0 (S i) ↔ path T s 0 i ∧ T (s i) (s (S i)).

Lemma divide_hd_path : ∀ (i j : ℕ) (s : ss) (T : trans),
  T (s i) (s (S i)) ∧ path T s (S i) j ↔ path T s i (S j).

Lemma shift_path : ∀ (i j : ℕ) (s : ss) (T : trans),
  path T s 0 i ∧ path T s i (S j) ↔ path T s 0 (S i) ∧ path T s (S i) j .

Lemma divide_path : ∀ (i j: ℕ) (s : ss) (T : trans),
  path T s 0 (i+j) → path T s 0 i ∧ path T s i j.

Lemma divide_lc1 : ∀ (j i : ℕ) (s : ss) (l : list ℕ),
 loop_check s 0 (i+j) l → loop_check s 0 i l.

Lemma divide_lc2'' : ∀ (i j k : ℕ) (s : ss) (l : list ℕ),
 loop_check' s i (S k) (S j) l ↔
 neq_nth_mth (s (i + (S k))) (s i) l 0 ∧ loop_check' s (S i) k j l.

Lemma divide_lc2' : ∀ (j i : ℕ) (s : ss) (l : list ℕ),
 loop_check s i (S j) l → loop_check s (S i) j l.

Lemma divide_lc2 : ∀ (i j : ℕ) (s : ss) (l : list ℕ),
 loop_check s 0 (i+j) l → loop_check s i j l.

Lemma divide_loop_check : ∀ (i j: ℕ) (s : ss) (l : list ℕ),
 loop_check s 0 (i+j) l → loop_check s 0 i l ∧ loop_check s i j l.

Lemma lt_big_and_incl : ∀ (i k : ℕ) (P : ℕ → Prop),
   i < k ∧ big_and P 0 (S k) → big_and P 0 (S i).
```

2　Sheeran らの方法 2

k-induction として知られる検査法.

$$\Big((\neg\exists s_{[0..k]}.\ I(s_0) \wedge loopFree(s_{[0..k]})) \vee (\neg\exists s_{[0..k]}.\ loopFree(s_{[0..k]}) \vee all.P(s_{[0..(k-1)]}) \vee \neg P(s_k)) \Big) \wedge \bigwedge_{0\le i\le k} \big(\neg\exists s_{[0..i]}.\ I(s_0) \wedge path(s_{[0..i]}) \wedge \neg P(s_i)\big) \tag{8}$$

2.1　証明の概要

以下, 証明の流れを述べる. まず, $i < k$ と $i \geq k$ に場合分けを行い, それぞれ

$$\forall s_{[0..i]}.\ \neg(I(s_0) \wedge loopFree(s_{[0..i]}) \wedge\neg\ P(s_i)) \tag{9}$$

が成り立つことを証明する.

$i < k$ の場合, 式 (8) の 2 行目より, k ステップ目まで安全であることが分かっているので, 式 (9) が成り立つ.

つぎに, $i \geq k$ の場合について証明する. 式 (1) より,

(a) $\forall s_{[0..k]}.\ \neg(I(s_0) \wedge loopFree(s_{[0..k]}))$

(b) $\forall s_{[0..k]}.\ \neg(loopFree(s_{[0..k]}) \wedge all.P(s_{[0..(k-1)]}) \wedge \neg P(s_k))$

のどちらかが成り立つことがわかっている. (a) が成り立つ場合の証明は, 方法 1 のときと同じなのでここでは省略し, (b) が成り立つ場合についての証明のみを行う.

(b) が成り立つ場合の証明には, 完全帰納法 (complete induction) を使用して証明する. 完全帰納法とは, 以下の式が妥当であることをいう.

$$\Big(\ \forall i.\Big(\forall m.m < i \to P(m)\Big) \to P(i)\ \Big)\ \to\ \forall i.P(i) \tag{10}$$

$k = 0$ は,

次に, $k \geq 1$ に対しての証明を行う. 式 (9) を以下のように変形する.

$$\forall s_{[0..i]}.\ I(s_0) \wedge loopFree(s_{[0..i]}) \to P(s_i) \tag{11}$$

そして, $P(s_i)$ の部分に式 (10) を適用して整理すると, 仮定は以下のようになる.

$$\forall s_{[0..i]}.\ I(s_0) \wedge loopFree(s_{[0..i]}) \tag{12}$$

$$\forall s_{[0..k]}.\ loopFree(s_{[0..k]}) \wedge all.P(s_{[0..(k-1)]}) \to P(s_k) \tag{13}$$

$$\forall m.\ \forall s_m.\ m < i \to P(s_m) \tag{14}$$

上の 3 つの式から, $P(s_i)$ を導出できればよい. 上の式を以下のように変形する. $(i-k \geq 0)$

$$\forall s_{[0..i]}.\ I(s_0) \wedge loopFree(s_{[0..(i-k)]}) \wedge loopFree(s_{[(i-k)..i]}) \tag{15}$$

$$\forall s_{[(i-k)..i]}.\ loopFree(s_{[(i-k)..i})} \wedge all.P(s_{[(i-k)..(i-1)]}) \to P(s_i) \tag{16}$$

$$\forall m.\ \forall s_{[0..m]}.\ m < i \to I(s_0) \wedge loopFree(s_{[0..(i-m)]}) \to P(s_m) \tag{17}$$

式 (16) から, $P(s_i)$ を $loopFree(s_{[(i-k)..i})} \wedge\ all.P(s_{[(i-k)..(i-1)]})$ に変形できる. これは, $loopFree(s_{[(i-k)..i})} \wedge P(s_{(i-k)}) \wedge P(s_{(i-k+1)}) \wedge .. \wedge P(s_{(i-1)})$ であるので, 式 (15), 式 (17) から成り立つ. よって. (b) が成り立つとき場合も, 式 (3) は成り立つ.