语义等价与精化

1 定义与例子

表达式语义等价

```
Record eequiv (e1 e2: expr): Prop := {
  nrm_eequiv:
    ⟦ e1 ⟧.(nrm) == ⟦ e2 ⟧.(nrm);
  err_eequiv:
    ⟦ e1 ⟧.(err) == ⟦ e2 ⟧.(err);
}.
```

表达式精化关系

```
Record erefine (e1 e2: expr): Prop := {
  nrm_erefine:
    ⟦ e1 ⟧.(nrm) ⊆ ⟦ e2 ⟧.(nrm) ∪ (⟦ e2 ⟧.(err) × int64);
  err_erefine:
    ⟦ e1 ⟧.(err) ⊆ ⟦ e2 ⟧.(err);
}.
```

程序语句语义等价

```
Record cequiv (c1 c2: com): Prop := {
  nrm_cequiv: ⟦ c1 ⟧.(nrm) == ⟦ c2 ⟧.(nrm);
  err_cequiv: ⟦ c1 ⟧.(err) == ⟦ c2 ⟧.(err);
  inf_cequiv: ⟦ c1 ⟧.(inf) == ⟦ c2 ⟧.(inf);
}.
```

程序语句精化关系

```
Record crefine (c1 c2: com): Prop := {
  nrm_crefine:
    ⟦ c1 ⟧.(nrm) ⊆ ⟦ c2 ⟧.(nrm) ∪ (⟦ c2 ⟧.(err) × state);
  err_crefine:
    ⟦ c1 ⟧.(err) ⊆ ⟦ c2 ⟧.(err);
  inf_crefine:
    ⟦ c1 ⟧.(inf) ⊆ ⟦ c2 ⟧.(inf) ∪ ⟦ c2 ⟧.(err);
}.
```

精化的例子

```
Lemma const_plus_const_refine: forall n m: Z,
  EConst (n + m) <<= [[n + m]].
```

证明见 Coq 代码。

- 定理：$c_1; (c_2; c_3) \equiv (c_1; c_2); c_3$。

- 证明：程序正常终止的情况

$$
\begin{aligned}
& [\![c_1; (c_2; c_3)]\!] .\mathrm{nrm} \\
= \quad & [\![c_1]\!] .\mathrm{nrm} \circ ([\![c_2]\!] .\mathrm{nrm} \circ [\![c_3]\!] .\mathrm{nrm}) \\
= \quad & ([\![c_1]\!] .\mathrm{nrm} \circ [\![c_2]\!] .\mathrm{nrm}) \circ [\![c_3]\!] .\mathrm{nrm} \\
= \quad & [\![(c_1; c_2); c_3]\!] .\mathrm{nrm}
\end{aligned}
$$

- 上面证明用到集合运算性质：$A \circ (B \circ C) = (A \circ B) \circ C$。

```
Theorem CSeq_assoc: forall (c1 c2 c3: com),
  [[c1; (c2; c3)]] ~=~ [[(c1; c2); c3]].
Proof.
  intros.
  split.
  + simpl.
    rewrite Rels_concat_assoc.
    reflexivity.
  + simpl.
    rewrite Rels_concat_union_distr_l.
    rewrite Sets_union_assoc.
    rewrite Rels_concat_assoc.
    reflexivity.
  + simpl.
    rewrite Rels_concat_union_distr_l.
    rewrite Sets_union_assoc.
    rewrite Rels_concat_assoc.
    reflexivity.
Qed.
```

- 定理：$\text{if } (e) \text{ then}\{c_1\} \text{ else } \{c_2\}; c_3 \equiv \text{if } (e) \text{ then}\{c_1; c_3\} \text{ else } \{c_2; c_3\}$。

- 证明：程序正常终止的情况

$$
\begin{aligned}
& [\![\text{if } (e) \text{ then}\{c_1\} \text{ else } \{c_2\}; c_3]\!] .\mathrm{nrm} \\
= \quad & (\text{test\_true}([\![e]\!]) \circ [\![c_1]\!] .\mathrm{nrm} \cup \text{test\_false}([\![e]\!]) \circ [\![c_2]\!] .\mathrm{nrm}) \circ \\
& [\![c_3]\!] .\mathrm{nrm} \\
= \quad & \text{test\_true}([\![e]\!]) \circ [\![c_1]\!] .\mathrm{nrm} \circ [\![c_3]\!] .\mathrm{nrm} \cup \\
& \text{test\_false}([\![e]\!]) \circ [\![c_2]\!] .\mathrm{nrm} \circ [\![c_3]\!] .\mathrm{nrm} \\
= \quad & [\![\text{if } (e) \text{ then}\{c_1; c_3\} \text{ else } \{c_2; c_3\}]\!] .\mathrm{nrm}
\end{aligned}
$$

- 上面证明用到集合运算性质：$(A \cup B) \circ C = (A \circ C) \cup (B \circ C)$。

```
Theorem CIf_CSeq: forall e c1 c2 c3,
  [[ if e then { c1 } else { c2 }; c3 ]] ~=~
  [[ if e then { c1; c3 } else { c2; c3 } ]].
Proof.
  intros.
  split.
  + simpl.
    rewrite <- ! Rels_concat_assoc.
    apply Rels_concat_union_distr_r.
  + simpl.
    rewrite ! Rels_concat_union_distr_r.
    rewrite ! Rels_concat_union_distr_l.
    rewrite <- ! Rels_concat_assoc.
    sets_unfold; intros s; tauto.
  + simpl.
    rewrite ! Rels_concat_union_distr_r.
    rewrite ! Rels_concat_union_distr_l.
    rewrite <- ! Rels_concat_assoc.
    sets_unfold; intros s; tauto.
Qed.
```

2 语义等价于精化的性质

接下去，我们介绍语义等价的两条重要性质。其一：语义等价是一种等价关系。

- 对于任意表达式 E, $E \equiv E$。
- 证明：
  - 求值成功的情况：$\llbracket E \rrbracket.\mathrm{nrm} = \llbracket E \rrbracket.\mathrm{nrm}$ （集合相等的自反性）；
  - 求值失败的情况：$\llbracket E \rrbracket.\mathrm{err} = \llbracket E \rrbracket.\mathrm{err}$ （集合相等的自反性）；
- 对于任意表达式 E_1 与 E_2，如果 $E_1 \equiv E_2$，那么 $E_2 \equiv E_1$。
- 证明：
  - 求值成功的情况：由 $E_1 \equiv E_2$ 这一假设可知 $\llbracket E_1 \rrbracket.\mathrm{nrm} = \llbracket E_2 \rrbracket.\mathrm{nrm}$，故 $\llbracket E_2 \rrbracket.\mathrm{nrm} = \llbracket E_1 \rrbracket.\mathrm{nrm}$ （集合相等的对称性）。
  - 求值失败的情况：由 $E_1 \equiv E_2$ 这一假设可知 $\llbracket E_1 \rrbracket.\mathrm{err} = \llbracket E_2 \rrbracket.\mathrm{err}$，故 $\llbracket E_2 \rrbracket.\mathrm{err} = \llbracket E_1 \rrbracket.\mathrm{err}$ （集合相等的对称性）。
- 对于任意表达式 E_1，E_2 与 E_3, 如果 $E_1 \equiv E_2$ 且 $E_2 \equiv E_3$，那么 $E_1 \equiv E_3$。
- 证明：
  - 求值成功的情况：由 $E_1 \equiv E_2$ 与 $E_2 \equiv E_3$ 这两条假设可知 $\llbracket E_1 \rrbracket.\mathrm{nrm} = \llbracket E_2 \rrbracket.\mathrm{nrm}$ 并且 $\llbracket E_2 \rrbracket.\mathrm{nrm} = \llbracket E_3 \rrbracket.\mathrm{nrm}$，故 $\llbracket E_1 \rrbracket.\mathrm{nrm} = \llbracket E_3 \rrbracket.\mathrm{nrm}$ （集合相等的传递性）。
  - 求值失败的情况：由 $E_1 \equiv E_2$ 与 $E_2 \equiv E_3$ 这两条假设可知 $\llbracket E_1 \rrbracket.\mathrm{err} = \llbracket E_2 \rrbracket.\mathrm{err}$ 并且 $\llbracket E_2 \rrbracket.\mathrm{err} = \llbracket E_3 \rrbracket.\mathrm{err}$，故 $\llbracket E_1 \rrbracket.\mathrm{err} = \llbracket E_3 \rrbracket.\mathrm{err}$ （集合相等的传递性）。

在 Coq 标准库中，`Reflexive`、`Symmetric`、`Transitive`以及`Equivalence`定义了自反性、对称性、传递性以及等价关系。下面证明中，我们统一使用了`Instance`关键字，而非之前证明中常用的`Theorem`与`Lemma`，我们将稍后再解释`Instance`关键字的特殊作用。

```
Instance eequiv_refl: Reflexive eequiv.
Proof.
  unfold Reflexive; intros.
  split.
  + reflexivity.
  + reflexivity.
Qed.
```

```
Instance eequiv_sym: Symmetric eequiv.
Proof.
  unfold Symmetric; intros.
  split.
  + rewrite H.(nrm_eequiv).
    reflexivity.
  + rewrite H.(err_eequiv).
    reflexivity.
Qed.
```

```
Instance eequiv_trans: Transitive eequiv.
Proof.
  unfold Transitive; intros.
  split.
  + rewrite H.(nrm_eequiv), H0.(nrm_eequiv).
    reflexivity.
  + rewrite H.(err_eequiv), H0.(err_eequiv).
    reflexivity.
Qed.
```

```
Instance eequiv_equiv: Equivalence eequiv.
Proof.
  split.
  + apply eequiv_refl.
  + apply eequiv_sym.
  + apply eequiv_trans.
Qed.
```

下面还可以证明精化关系也具有自反性和传递性。

```
Instance erefine_refl: Reflexive erefine.
Proof.
  unfold Reflexive; intros.
  split.
  + apply Sets_included_union1.
  + reflexivity.
Qed.
```

```
Instance erefine_trans: Transitive erefine.
Proof.
  unfold Transitive; intros.
  split.
  + rewrite H.(nrm_erefine).
    rewrite H0.(nrm_erefine).
    rewrite H0.(err_erefine).
    sets_unfold; intros s1 s2; tauto.
  + rewrite H.(err_erefine).
    rewrite H0.(err_erefine).
    reflexivity.
Qed.
```

并且精化关系在语义等价变换下不变。

```
Instance erefine_well_defined:
  Proper (eequiv ==> eequiv ==> iff) erefine.
Proof.
  unfold Proper, respectful; intros.
  split; intros.
  + split.
    - rewrite <- H.(nrm_eequiv).
      rewrite <- H0.(nrm_eequiv).
      rewrite <- H0.(err_eequiv).
      apply H1.(nrm_erefine).
    - rewrite <- H.(err_eequiv).
      rewrite <- H0.(err_eequiv).
      apply H1.(err_erefine).
  + split.
    - rewrite H.(nrm_eequiv).
      rewrite H0.(nrm_eequiv).
      rewrite H0.(err_eequiv).
      apply H1.(nrm_erefine).
    - rewrite H.(err_eequiv).
      rewrite H0.(err_eequiv).
      apply H1.(err_erefine).
Qed.
```

两条重要性质之二是：三种语法连接词能保持语义等价关系（congruence）。下面先证明加法、减法、乘法的情况。

```
Lemma arith_sem1_nrm_congr: forall Zfun D11 D12 D21 D22,
  D11 == D12 ->
  D21 == D22 ->
  arith_sem1_nrm Zfun D11 D21 ==
  arith_sem1_nrm Zfun D12 D22.
Proof.
  sets_unfold.
  intros ? ? ? ? ? ? ? ? s1 s2.
  unfold arith_sem1_nrm.
  apply ex_iff_morphism; intros i1.
  apply ex_iff_morphism; intros i2.
  rewrite H, H0.
  reflexivity.
Qed.
```

很多其它情况的证明是类似的。