Formalizing some results related to ordered and unordered pairs

KEVIN COLLINS, JANITHA ASWEDIGE, and MINH NGUYEN, University of Hawaiʻi at Mānoa, USA

1 INTRODUCTION

We formalize the proofs of several theorems related to ordered and unordered pairs in Chapter I in the Kunen's textbook [1].

The preamble:

```
open classical
namespace Kunen

constant U : Type -- the universe of our model ; sets, but we cannot use Lean's idea of set
constant el : U → U → Prop
local infix ∈ := el
constant set_existence_witness : U -- sneaky
axiom el_ext {x y : U} : (∀ z: U, z ∈ x ↔ z ∈ y) → x=y
axiom ext_el {x y : U} : x=y → (∀ z: U, z ∈ x →  z ∈ y)∧ (∀ z: U, z ∈ y →  z ∈ x)
def empty_set (x:U) : Prop := (∀ y, ¬  y ∈ x)

def φ_1 (x: U): Prop := false
def φ_2 (x: U): Prop := ¬ x ∈ x


axiom comprehension_1 (x : U): ∃ y:U, ∀ z, z ∈ y ↔ z ∈ x ∧ φ_1(z)
axiom comprehension_2 (x : U): ∃ y:U, ∀ z, z ∈ y ↔ z ∈ x ∧ φ_2(z)
```

2 MAIN DEVELOPMENT

We proceed the proofs of several theorems of ordered and unordered pairs as follows.

```
theorem at_most_one_empty_set :
    ∀ emp1:U, ∀ emp2:U, (empty_set emp1 → empty_set emp2 →  emp1 = emp2) :=
        λ emp1 emp2, λ h1: empty_set emp1, λ h2: empty_set emp2,
        have h: (∀ z:U, z ∈ emp1 ↔ z ∈ emp2), from
            λ z,
            have g1: z ∈ emp1 → z ∈ emp2, from λ g2: z ∈ emp1, false.elim (h1 z g2),
            have g2: z ∈ emp2 → z ∈ emp1, from λ g2: z ∈ emp2, false.elim (h2 z g2),
            iff.intro g1 g2,
        el_ext h

theorem at_least_one_empty_set :
```

. Manuscript submitted to ACM

```
  ∃  emp:U, empty_set emp :=
    have h : ∃ y:U, ∀ z, z ∈ y ↔ z ∈ set_existence_witness ∧ φ_1(z), from
        comprehension_1 set_existence_witness,
    exists.elim h (
        assume emp,
        assume g: ∀ z, z ∈ emp ↔ z ∈ set_existence_witness ∧ φ_1(z),
        have g1: empty_set emp, from
            assume y,
            assume g2: y ∈ emp,
            have g3: y ∈ emp ↔ y ∈ set_existence_witness ∧ φ_1(y), from g y,
            have g4: y ∈ emp → y ∈ set_existence_witness ∧ φ_1(y), from iff.elim_left g3,
            have g5: y ∈ set_existence_witness ∧ φ_1(y), from g4 g2,
            show false, from and.elim_right g5,
        exists.intro emp g1
    )

theorem no_universal_set :
  ∀ x, ∃ y, ¬ y ∈ x :=
  assume univ,
  have h : ∃ y, ∀ z, z ∈ y ↔ z ∈ univ ∧ φ_2(z), from comprehension_2 univ,
  exists.elim h (
      assume diag,
      assume g: ∀ z, z ∈ diag ↔ z ∈ univ ∧ φ_2(z),
      have g0: diag ∈ diag ↔ diag ∈ univ ∧ ¬ diag ∈ diag, from g diag,
      have g1: ¬ diag ∈ univ, from
          assume g2: diag ∈ univ,
              -- it's just propositional logic at this point,
              -- we don't need excluded middle but it helps
              (em (diag ∈ diag)).elim (
                  assume h0: diag ∈ diag,
                  (and.elim_right ((iff.elim_left g0) h0) h0) -- nice!
              )(
                  assume h0: ¬ diag ∈ diag,
                  h0 ((iff.elim_right g0) (and.intro (g2)(h0))) -- nice!
              ),
      exists.intro diag g1
  )
```

We now define some additional definitions and axioms.

```
axiom pairing (x y: U): ∃ w:U, (x∈ w∧ y∈ w) ∧ (∀ a:U,a ∈ w → a=x ∨ a=y)
def unordered_pair (x y w: U): Prop :=  (x∈ w ∧ y∈ w)∧ (∀ a: U, a∈ w → (a=x ∨ a=y))
def set_of_x (x u : U): Prop :=(x∈ u) ∧ (∀ a:U, a∈ u→ a=x)
def ordered_pair (x y u w z: U): Prop :=
```

```
set_of_x x u ∧ unordered_pair x y w ∧ ((u∈ z ∧ w∈ z)∧ (∀ a:U, a∈ z ⟶ a= u ∨ a=w))


theorem uop_sox: ∀ x w u: U, (unordered_pair x x w) ⟶ (set_of_x x u)⟶ (u=w):=
  assume x w z:U,
  assume h1: unordered_pair x x w,
  assume h,
  have h2:  (x∈ w ∧ x∈ w)∧ (∀ a: U, a∈ w ⟶ (a=x ∨ a=x)), from h1,
  have h3:  x∈ w ∧ (∀ a:U, a∈ w ⟶ a=x), from
  (
    have h4: x∈ w, from and.elim_left (and.elim_left h2),
    have h5: ∀ a: U, a∈ w ⟶ (a=x ∨ a=x), from and.elim_right h2,
    have h6: ∀ a:U, a∈ w ⟶ a=x, from
    (assume b:U,
      assume bw: b∈ w,
      or.elim (h5 b bw)
        (assume h7,
        h7
        )
        (assume h7,
        h7
        )
    ),
  and.intro h4 h6),
 have h4: (x∈ z) ∧ (∀ a:U, a∈ z⟶ a=x), from h,
 have h5: ∀ a:U, a∈ z ↔ a∈ w, from
 (assume a:U,
  have hazw: a∈ z ⟶ a∈ w, from
    (
    assume haz: a∈ z,
    have hix: a=x,from h4.right a haz,
    have xiw: x∈ w, from and.elim_left h3,
    have aiw: a∈ w, from eq.subst (symm hix) xiw,
    aiw
    ),
  have hawz: a∈ w ⟶ a∈ z, from
    (assume aiw: a∈ w,
    have xiz: x∈ z, from and.elim_left h4,
    have hix: a=x, from h3.right a aiw,
     eq.subst (symm hix) xiz
    ),
  iff.intro hazw hawz
  ),
```

```
  el_ext (h5)


theorem exists_setofx: ∀ x: U, ∃ u: U, set_of_x x u :=
assume x:U,
have h: ∃ w:U, (x∈ w∧ x∈ w) ∧ (∀ a:U,a ∈ w → a=x ∨ a=x), from pairing x x,
have h1: ∃ u:U, x∈ u ∧ ∀ a:U, a∈ u → a=x, from
  (exists.elim h
    (
    assume u,
    assume h2,
    have h3: x∈ u, from and.elim_left (and.elim_left h2),
    have h4: ∀ a:U, a∈ u→ a=x ∨ a=x, from (and.elim_right h2),
    have h5: ∀ a:U, a∈ u → a=x, from
      assume a:U,
      assume hau,
      or.elim (h4 a hau)
        (assume hax,
        hax
        )
        (assume hax,
        hax
        ),
        exists.intro u (and.intro h3 h5)
    )
  ),
  h1


theorem exists_unordered_pair: ∀ x:U, ∀ y:U, ∃ w: U, unordered_pair x y w :=
assume x y: U,
have h1: ∃ w:U, (x∈ w∧ y∈ w) ∧ (∀ a:U,a ∈ w → a=x ∨ a=y), from pairing x y,
exists.elim h1
  (assume w,
  assume h2,
  exists.intro w h2
  )


theorem exists_ordered_pair: ∀ x y:U, ∃  u w z:U, ordered_pair x y u w z :=
assume x y:U,
have usox: ∃ u:U, set_of_x x u, from exists_setofx x,
have wuop: ∃ w:U, unordered_pair x y w, from exists_unordered_pair x y,
```

```
exists.elim usox
  (assume u:U,
  assume soxu,
  exists.elim wuop
    (assume w:U,
    assume uopw,
    have zop: ∃ z:U, ((u∈ z ∧ w∈ z)∧ (∀ a:U, a∈ z ⟶ a= u ∨ a=w)), from pairing u w,
    exists.elim zop
      (assume z,
      assume opz,
      have op: set_of_x x u ∧ unordered_pair x y w ∧ ((u∈ z ∧ w∈ z)∧ (∀ a:U, a∈ z ⟶ a= u ∨
     a=w)), from and.intro soxu (and.intro (uopw)           opz),
      exists.intro u (exists.intro w (exists.intro z op))
      ))
  )
```

Main result:

```
theorem ordered_pair_equiv: ∀ x y u w z a b s t: U, ordered_pair x y u w z ⟶ ordered_pair a b s t
     z ⟶ (x=a ∧ y=b):=
assume x y u w z a b s t: U,
assume h1 h2,
have h3: set_of_x x u, from and.left (h1),
have h4: unordered_pair x y w, from and.left (and.right h1),
have h5: set_of_x a s, from and.left (h2),
have h6: unordered_pair a b t, from and.left (and.right h2),
have uiz: u∈ z, from and.left (and.left (and.right (and.right h1))),
have tinz: t∈ z, from and.right (and.left (and.right (and.right h2))),
have tuv: t=u ∨ t=w, from and.right(and.right (and.right h1)) t tinz,
by_cases
  (
  assume hxy: x=y,
  have uopxysox: w=u, from
    have uopxx: unordered_pair x x w, from eq_uop x y w hxy h4,
    symm(uop_sox x w u uopxx h3),
  have tisu: t=u, from or.elim tuv
    (assume h,
    h
    )
```

```
    (assume h,
    eq.subst uopxysox h
    ),
  have aint: a∈ t, from h6.left.left,
  have bint: b∈ t, from h6.left.right,
  have ainu: a∈ u, from (ext_el tisu).left a aint,
  have binu: b∈ u, from (ext_el tisu).left b bint,
  have aisx: a=x, from h3.right a ainu,
  have bisx: b=x, from h3.right b binu,
  have bisy: b=y, from eq.subst hxy bisx,
  and.intro (symm aisx) (symm bisy)
  )
(
assume hxy,
have uiss: u=s, from
  (have uissort: u=s ∨ u=t, from h2.right.right.right u uiz,
  or.elim uissort
    (assume h,
    h
    )
    (assume h,
    have ainu: a∈ u, from (ext_el h).right a h6.left.left,
    have binu: b∈ u, from (ext_el h).right b h6.left.right,
    have aisx: a=x, from h3.right a ainu,
    have bisx: b=x, from h3.right b binu,
    have winz: w∈ z, from h1.right.right.left.right,
    have wst: w=s∨ w=t, from h2.right.right.right w winz,
    or.elim wst
      (assume ws,
      have yinw: y∈ w, from h4.left.right,
      have yins: y∈ s, from eq.subst ws yinw,
      have ya: y=a, from h5.right y yins,
      have yisx: y=x, from eq.subst aisx ya,
      false.elim (hxy (symm yisx))
      )
      (assume wist,
      have yinw: y∈ w, from h4.left.right,
      have yint: y∈ t, from eq.subst wist yinw,
      have yisaorb: y=a∨ y=b, from h6.right y yint,
      have yisx: y=x, from or.elim yisaorb
        (assume yisa,
        eq.subst aisx yisa
        )
```

```
        (assume yisb,
        eq.subst bisx yisb
        ),
        false.elim (hxy (symm yisx))
      )
    )
  ),
  have xinu: x∈ u, from h3.left,
  have xins: x∈ s, from (ext_el uiss).left x xinu,
  have xisa: x=a, from h5.right x xins,
  have winz: w∈ z, from h1.right.right.left.right,
  have wissort: w=s ∨ w=t, from h2.right.right.right w winz,
  have yinw: y∈ w, from h4.left.right,
  or.elim wissort
    (assume wiss,
    have yins: y∈ s, from eq.subst wiss yinw,
    have yisa: y=a, from h5.right y yins,
    have yisx: y=x, from eq.subst (symm xisa) yisa,
    false.elim (hxy (symm yisx))
    )
    (assume wist,
    have yint: y∈ t, from eq.subst wist yinw,
    have yisaorb: y=a ∨ y=b, from h6.right y yint,
    or.elim yisaorb
      (assume yisa,
      have yisx: y=x, from eq.subst (symm xisa) yisa,
      false.elim (hxy (symm yisx))
      )
      (assume yisb,
      and.intro xisa yisb
      )
    )
)


-- Below This Line is our attempt at setting up Unions and Intersections and doing Exercise I.6.17

def Union (x y u:U): Prop := ∀ a:U, a∈ u ↔ (a∈ x ∨ a∈ y)
def Intersection (x y u:U): Prop := ∀ a:U, a∈ u ↔ (a∈ x ∧ a∈ y)
def uop_xyz (x y z w:U): Prop := ∀ a:U, a∈ w ↔ (a= x ∨ a= y ∨ a= z)
```

end Kunen

REFERENCES

[1] Kenneth Kunen. 2009. *The Foundations of Mathematics.* College Publications.