Bi↔implication

What do we mean by “bi-implication”

- $P \leftrightarrow Q$ is a bi-implication, meaning that $P \rightarrow Q$ and $Q \rightarrow P$
  - Bi-implication is equality for propositions
  - Bi-implication is also a proposition
  - Can also call it “iff” or “if and only if”
- So, how do we prove $P \leftrightarrow Q$?
  - First we prove $P \rightarrow Q$, and then we prove $Q \rightarrow P$

Introduction rule

- Lean provides the iff.intro rule that works just like you would expect

```
lemma bi_implication :
  ∀{ P Q : Prop },
    (P → Q) → (Q → P) → (P ↔ Q) :=
      λ P Q pfPQ pfQP,
        iff.intro pfPQ pfQP
```

Elimination rules

- As we said earlier, P ↔ Q means that P → Q ***<u>and</u>*** Q → P
- Thus, we should expect iff to have left elimination (elim_left) and right elimination (elim_right) rules

```
#check iff.elim_left (iff.intro forward backward)
#check iff.elim_right (iff.intro forward backward)
```

- See connected example in implication_properties.lean

Exercise

- If we have bi-implication between P and Q and bi-implication between Q and R then we can derive derive bi-implication between P and R by way of Q.

$$\frac{\{\ P\ Q : \mathrm{Prop}\ \}\ (pq : P \leftrightarrow Q)\ (qr : Q \leftrightarrow R)}{pr : (P \leftrightarrow R)}\ \text{chain}$$

- Exercise: Prove it

Exercise

- Construct a proof, pqequiv, of the proposition, $P \land Q \leftrightarrow Q \land P$. Note: we don't need to know whether P and Q are true, false, or unknown to provide such a proof

```
theorem pqequiv : P ∧ Q ↔ Q ∧ P :=
```

Exercise

- Construct a proof, a_imp_b_imp_c_iff_a_and_b_imp_c, that $A \to B \to C \leftrightarrow A \land B \to C$.

```
lemma a_imp_b_imp_c_iff_a_and_b_imp_c:
  ∀ A B C: Prop, (A → B → C) ↔ ((A ∧ B) → C) :=
  λ A B C: Prop,
    begin
      sorry
    end
```

- Given that you can prove this, does this mean that $A \to B = A \land B$?

Storytime…

```
segment_intersect_kernel(s1, s2: segment_2d):
    [segment_intersection_type, point_2d] =
  LET p: point_2d = s1`p1 IN
  LET r: vector_2d = vector_from_point_to_point(s1`p1, s1`p2) IN
  LET q: point_2d = s2`p1 IN
  LET s: vector_2d = vector_from_point_to_point(s2`p1, s2`p2) IN
  LET r_cross_s: real = cross(r, s) IN
  LET q_minus_p_cross_r: real = cross((q - p), r) IN
  IF ((r_cross_s = 0) AND (q_minus_p_cross_r = 0)) THEN
    LET t0: real = (s2`p1 - s1`p1) * r IN
    LET t1: real = (s2`p2 - s1`p1) * r IN
    LET norm_sq: nnreal = r * r IN
    IF ((0 <= t0 AND t0 <= norm_sq) OR (0 <= t1 AND t1 <= norm_sq)) THEN
      (Collinear_Overlapping, zero_point)
    ELSE
      (Collinear_Non_Overlapping, zero_point)
    ENDIF
  ELSIF ((r_cross_s = 0) AND (q_minus_p_cross_r /= 0)) THEN
    (Parallel, zero_point)
  ELSE
    LET q_minus_p_cross_s: real = cross((q - p), s) IN
    LET t: real = q_minus_p_cross_s / r_cross_s IN
    LET u: real = q_minus_p_cross_r / r_cross_s IN
    IF ((0 <= t AND t <= 1) AND (0 <= u AND u <= 1)) THEN
      (Intersecting, mk_vect2(p`x + t * r`x, p`y + t * r`y))
    ELSE
      (Non_Parallel_Not_Intersecting, zero_point)
    ENDIF
  ENDIF;
```

Storytime… (2)

```
are_segments_intersecting_alt?(s1, s2: segment_2d): bool =
  EXISTS(p1: (is_point_on_segment?(s1))):
    is_point_on_segment?(s2)(p1);

are_segments_intersecting?_defs_same: LEMMA
  FORALL(s1, s2: segment_2d):
    are_segments_intersecting?(s1)(s2) iff
    are_segments_intersecting_alt?(s1, s2);
```

Bi-Implication Properties

- Reflexive? Is $A \leftrightarrow A$ true?
  - Yes
- Symmetric? Does $(A \leftrightarrow B) \rightarrow (B \leftrightarrow A)$?
  - Yes
- Associative? Does $((A \leftrightarrow B) \leftrightarrow C) \leftrightarrow (A \leftrightarrow (B \leftrightarrow C))$?
  - Yes, classically
- Transitive? Does $(A \leftrightarrow B) \rightarrow (B \leftrightarrow C) \rightarrow (A \leftrightarrow C)$?
  - Yes
- Connected? Does $(A \neq B) \rightarrow ((A \leftrightarrow B) \vee (B \leftrightarrow A))$?
  - No

Fin