∃xistentialism
(Existential Quantification)

What is Existential Quantification?

- An existentially quantified proposition asserts that there is some value of some type for which some proposition involving that value is true. Here are a couple of examples:
  - $\exists\, m \in \mathbb{N},\, m + m = 8$
  - $\exists\, m \in \mathbb{N},\, m > 10$
  - $\exists\, m \in \mathbb{N},\, m - m = 3$
- In Lean, we might write:

```
def anExistsProp :=
  exists m, m + m = 8
```

- Or:

```
def anotherExistsProp :=
  exists m, m > 10
def yetAnotherExistsProp :=
  exists m, m - m = 3
```

A Familiar Theme

- You can fool all of the people some of the time…
  - $\forall\ p \in \text{People}, \exists\ t \in \text{time}, \text{fool}(p, t)$ — everybody can be fooled at one time or another
  - $\exists\ t \in \text{time}, \forall\ p \in \text{People}, \text{fool}(p, t)$ — there exists a time when all of the people can be fooled simultaneously
  - $\exists\ t \in \text{time}, \forall\ p \in \text{People}, \text{fool}(p, t) \rightarrow \forall\ p \in \text{People}, \exists\ t \in \text{time}, \text{fool}(p, t)$
- …and some of the people all of the time.
  - $\exists\ p \in \text{People}, \forall\ t \in \text{time}, \text{fool}(p, t)$ — there exists somebody who can be fooled all of the time
  - $\forall\ t \in \text{time}, \exists\ p \in \text{People}, \text{fool}(p, t)$ — at any given moment, there exists somebody who can be fooled
  - $\exists\ p \in \text{People}, \forall\ t \in \text{time}, \text{fool}(p, t) \rightarrow \forall\ t \in \text{time}, \exists\ p \in \text{People}, \text{fool}(p, t)$

Existential Proofs

- Existential proofs have two components
  - Witness: value for which the sub-proposition holds (e.g., 4)
    - We only need a single witness, even if multiple are available
  - Proof that the sub-proposition holds for the specified witness
- The introduction rule for existential quantifiers is:

```
(T : Type) (pred: T → Prop) (w : T) (e : pred w)
------------------------------------------------
              ∃ a : T, pred a
```

Deconstructing exists.intro

```
def existsIntro
   (T: Type)
   (pred: T → Prop)
   (witness : T)
   (proof : pred witness)
   : ∃(w), pred w
   := exists.intro witness proof
```

Abstract Example

```
example{T: Type}{witness: T}
  {predicate: T → Prop}
  {proof: predicate witness}:
  ∃ m, predicate m :=
    ⟨ witness, proof ⟩
```

Concrete Example

```
def isEven(n : nat) : Prop :=
  ∃(m : nat), m + m = n

lemma pf8is4twice: 4 + 4 = 8 := rfl

theorem even8: isEven 8 :=
  exists.intro 4 pf8is4twice

theorem even8': ∃(m : nat), m + m = 8 :=
  exists.intro 4 pf8is4twice
```

Concrete Example (2)

```
theorem even8'': eightEven :=
  ⟨ 4, pf8is4twice ⟩

theorem even8''' : isEven 8 :=
begin
  unfold isEven, -- not necessary
  exact ⟨ 4, pf8is4twice ⟩
end
```

Exercise

- Construct a proof, `isNonZ`, of the proposition that there exists a natural number n such that 0 ≠ n.

```
theorem isNonZ : exists n : nat, 0 ≠ n :=
  exists.intro 1 (λ pf : (0 = 1),
    nat.no_confusion pf)

theorem isNonZ' : exists n : nat, 0 ≠ n :=
begin
  have pf0isnt1: (0 ≠ 1),
    apply nat.no_confusion,

  exact ⟨ 1, pf0isnt1 ⟩
end
```

Existential Elimination

- If one assumes that there exists an x such that $P(x)$ is true, then one can assume there is an arbitrary value, w, such that $P(w)$ is true. If one can then show, without making additional assumptions about w, that some conclusion, Q that does not depend on w, follows, that one has shown that Q follows from the mere existence of a w with property P, and thus from $\exists\, x, P\, x$.

Existential Elimination Inference Rule

```
Q: Prop; T: Type; P: (T → Prop); ∃(x: T), P x; (∀(a: T), P a) → Q
-------------------------------------------------------------------
                                Q
```

- This rule says that we can conclude that any proposition, *Q*, is true, if
  1. T is any type of value;
  2. P is any property of values of this type;
  3. there is some value, x, of this type that has property P; and
  4. from any such value, w, Q then follows.

Deconstructing Existential Elimination

```
def existsElim
  { Q : Prop }
  { T : Type }
  { P : T → Prop }
  ( ex : exists x, P x)
  ( pw2q : ∀ w : T, P w → Q)
  : Q
  := exists.elim ex pw2q
```

Existential Elimination Example

```
theorem forgetAProperty :
  (∃ n, P n ∧ S n) → (∃ n, P n) :=
   -- here Q, the conclusion, is (exists n, P n)
begin
  assume ex,
  show ∃ (n : ℕ), P n,
  from
    begin
      apply exists.elim ex, -- give one arg, build other
      assume w Pw, -- assume w and proof of P w
      show ∃ (n : ℕ), P n,
      from exists.intro w Pw.left,
    end,
end
```

Exercise

- Assuming n is a natural number and P and S are properties of natural numbers, prove that $(\exists\ n, P\ n \land S\ n) \to (\exists\ n, S\ n \land P\ n)$.

Answer to exercise

```
theorem reverseProperty :
  (∃ n, P n ∧ S n) → (∃ n, S n ∧ P n) :=
begin
  assume ex,
  show ∃ (n : ℕ), S n ∧ P n,
  from
    begin
      apply exists.elim ex, -- give one arg, build other
      assume w Pw, -- assume w and proof of P w
      show ∃ (n : ℕ), S n ∧ P n,
      from exists.intro w ⟨ Pw.right, Pw.left ⟩
    end,
end
```

Exercises

- Express the property, of natural numbers, of being a perfect square. For example, 9 is a perfect square, because 3 is a natural number such that 3 * 3 = 9. By contrast, 12 is not a perfect square, as there does not *exist* a natural number that squares to 12.

```
def isASquare: ℕ → Prop :=
  λ n, exists m, n = m ^ 2
```

- Prove the proposition that 9 is a perfect square

```
theorem isPS9 : isASquare 9 :=
begin
  unfold isASquare,
  exact exists.intro 3 (eq.refl 9)
end
```

Fooling All of the People Again

- Remember this claim:
  - ∃ t ∈ time, ∀ p ∈ People, fool(p, t) → ∀ p ∈ People, ∃ t ∈ time, fool(p, t)
- Let's look at a general proof

```
theorem existsforall_impl_forallexists:
  ∀ (S T: Type) (pred: (S → T → Prop)),
    (∃ (t: T), ∀ (p: S), pred(p)(t)) →
      (∀ (p: S), ∃ (t: T), pred(p)(t)) :=
begin
  …
end
```

Negating Existential and Universal Quantifiers

- What happens when you negate an existential quantifier?
- What does this mean:
  - $\neg(\exists\ t \in \text{time}, \text{fool}(\text{me}, t))$ — there does not exist a time when you can fool me
  - $\forall\ t \in \text{time}, \neg\text{fool}(\text{me}, t)$ — at any time, you will not fool me
  - Are these equivalent?
- How about this:
  - $\neg(\forall\ t \in \text{time}, \text{fool}(\text{me}, t))$ — you cannot fool me all of the time
  - $\exists\ t \in \text{time}, \neg\text{fool}(\text{me}, t)$ — there exists a time when you cannot fool me
  - Are these equivalent?

Proof of Existential Negation (1 of 2)

```
theorem not_exists_t_iff_always_not_t:
  {T: Type}{pred: (T → Prop)}:
    (¬(∃ t: T, pred(t))) ↔
      ∀ t: T, ¬pred(t) :=
begin
  apply iff.intro,
    -- ¬(∃ t: T, pred(t)) → ∀ t: T, ¬pred(t)
    assume pf_not_exists_t,
    assume t,
    assume Q,
    have pf_exists_t := exists.intro t Q,
    exact (pf_not_exists_t pf_exists_t)…
```

Proof of Existential Negation (2 of 2)

```
    -- ∀ t: T, ¬pred(t) → ¬(∃ t: T, pred(t))
    assume pf_forall_t_not,
    assume pf_not_exists_t,
    apply exists.elim pf_not_exists_t,
    assume a pf_a,
    have pf_not_a := pf_forall_t_not a,
    exact pf_not_a pf_a
end
```

Satisfiability

- Satisfiability is about finding values for sub-propositions such that a larger proposition is true
- Applies to propositional logic, not predicate logic and assumes the axiom of the excluded middle to be true
- Typically, we like to phrase the larger proposition in what is referred to as Conjunctive Normal Form, or CNF
  - E.g., $(x1 \lor x2 \lor \neg x3 \lor x4) \land (\neg x1 \lor x2 \lor \neg x3)$
- Do there exist values for propositions P and Q such that:
  - (P or Q) and (¬P or ¬Q)
  - See proof

Exercises

- Do there exist values for P and Q such that:
  $(P \lor Q) \land (\neg P \lor \neg Q) \land (\neg P \lor Q)$ is true?
  If so, prove it.
- Do there exist values for P and Q such that:
  $(P \lor Q) \land (\neg P \lor \neg Q) \land (\neg P \lor Q) \land (\neg Q)$ is true?
  If so, prove it.

3-SAT

- 3-SAT is a special kind of satisfiability (SAT) problem where each of the disjunctions have <u>no more than</u> 3 terms in each disjunction
  - E.g., $(P \vee Q \vee R) \wedge (\neg P \vee \neg Q \vee \neg S) \wedge (\neg P \vee Q \vee T) \wedge \ldots$
- 3-SAT is SAT, and 2-SAT is 3-SAT
- All SAT problems can be reduced to 3-SAT problems
- 3-SAT (and hence SAT) is NP-complete

Exercise

- Do there exist values for P, Q, and R such that:
  $(P \lor Q \lor R) \land (\neg P \lor \neg Q \lor \neg R) \land (\neg P \lor Q \lor R) \land (\neg Q \lor \neg R)$ is true?
  If so, prove it.

Fin