Author: [Your Name]

Table of Contents

1. Introduction

This project implements a bidirectional translator between simple English quantifier statements and first-order predicate logic. The tool parses natural-language sentences of the form “All X are Y,” “Some X are (not) Y,” and “No X are Y,” converting them into corresponding logical formulas, and vice versa.

2. Problem Statement

Users often need to formalize or interpret statements involving universal (`∀`) and existential (`∃`) quantification. Manual translation is error-prone. This script automates:

- **English → Predicate Logic:** Handle quantifiers `all` , `every` , `each` , `some` , `a` , `an` , `no` , `none` , including negation.
- **Predicate Logic → English:** Support formulas in the form `∀x (is_X(x) → [¬]is_Y(x))` and `∃x (is_X(x) ∧ [¬]is_Y(x))` .

The translator must correctly identify the quantifier, subject class, property, and negation to produce accurate conversions.

3. Methodology

1. **Parsing English:**

   - Tokenize and lowercase input.
   - Locate quantifier and subject (`word[0]` , `word[1]`).
   - Detect `are not` vs. `are` to determine negation.
   - Map to predicate syntax: subject → `is_subject(x)` , property → `is_property(x)` .
   - Assemble:
      - Universal: `∀x (is_subject(x) → [¬]is_property(x))` .
      - Existential: `∃x (is_subject(x) ∧ [¬]is_property(x))` .
2. **Parsing Logic:**
   - Identify leading `∀x` or `∃x` .
   - Split inner formula on `→` (universal) or `∧` (existential).
   - Extract subject and property predicates, detect `¬` prefix for negation.
   - Reconstruct English phrase: e.g., “all dinosaurs are extinct” or “some animals are not endangered.”
3. **Testing:**
   - Three test cases validate both directions:

	English Input
	All dinosaurs are extinct
	Some animals are endangered
	No dinosaurs are alive

   - Reverse translations compare against lowercase originals.

4. Results

All predefined test cases passed as expected:

Direction	Input	Expected	Result	Pass
English → Logic	All dinosaurs are extinct	∀x (is_dinosaur(x) → is_extinct(x))	∀x (is_dinosaur(x) → is_extinct(x))	✅
	Some animals are endangered	∃x (is_animal(x) ∧ is_endangered(x))	∃x (is_animal(x) ∧ is_endangered(x))	✅
	No dinosaurs are alive	∀x (is_dinosaur(x) → ¬is_alive(x))	∀x (is_dinosaur(x) → ¬is_alive(x))	✅
Logic → English	∀x (is_dinosaur(x) → is_extinct(x))	all dinosaurs are extinct	all dinosaurs are extinct	✅
	∃x (is_animal(x) ∧ is_endangered(x))	some animals are endangered	some animals are endangered	✅

Direction	Input	Expected	Result	Pass
	∀x (is_dinosaur(x) → ¬is_alive(x))	no dinosaurs are alive	no dinosaurs are alive	✅

5. Discussion

- **Robustness:** Works for standard quantifiers and simple negation but does not support compound English phrases (e.g., “not all”).
- **Limitations:** Subject and property names must be single words. Plurals handled by stripping trailing `s`, which may mis-handle irregular plurals.
- **Error Handling:** Returns informative messages for missing keywords (`are`), malformed logic, or unsupported quantifiers.

6. Conclusion

The predicate translator successfully automates routine English–logic conversions for basic quantifier statements. All tests passed, demonstrating correct parsing and generation for a representative sample.

7. Recommendations

- **Extend Vocabulary:** Handle multi-word subjects/properties (e.g., `large mammals`).
- **Support Complex Logic:** Incorporate nested quantifiers and additional logical connectives (`∨`, `↔`).
- **Internationalization:** Allow different natural languages by modularizing parsing rules.