Dawkins Weasel 1.2.0 — ODD Protocol (Overview, Design Concepts, Details)

1. Purpose

Dawkins Weasel is a NetLogo model designed for instructional use. It demonstrates why **cumulative selection** (retaining partial improvements) can rapidly produce a specified target phrase, while **random change alone** is extraordinarily unlikely to do so for longer phrases. The model is inspired by Richard Dawkins's "Weasel" thought experiment in *The Blind Watchmaker*.

2. Entities, state variables, and scales

Entities

The model's entities are **strings** (phrases) of fixed length, represented as NetLogo strings.

- **Parent string (single entity):** the current best string, stored as parent-string.
- **Offspring strings (temporary entities):** strings generated each generation (either many offspring when selection is on, or an in-place mutated string when selection is off). Offspring are not stored as persistent agents; they exist as local variables during each step.

There is no spatial structure and no interaction among entities; comparison occurs only between each candidate string and the fixed target.

Global variables

- allowed-chars (global): "ABCDEFGHIJKLMNOPQRSTUVWXYZ " (uppercase letters plus space).
- parent-string (global): current best phrase.
- generation (global): current generation count.

User settings

- target-phrase (input): target phrase the model attempts to match.
- mutation-rate (slider): per-character probability of mutation during copying.
- offspring-per-generation (input): number of offspring generated per generation when selection is on.
- with-selection (switch): whether selection is applied.

Temporal scale

Time is discrete and measured in **generations**. Each call to go advances the model exactly **one generation**.

3. Process overview and scheduling

The model proceeds in the following sequence:

Setup phase

1. The model clears previous state and output.
2. It defines the allowed character set (allowed-chars).
3. It **normalizes** target-phrase so it contains only characters in allowed-chars:
    - lowercase letters are converted to uppercase
    - any other characters are replaced with spaces
    - if target-phrase is empty, a default phrase is assigned
4. It creates a random initial string of the same length as target-phrase and stores it as parent-string.
5. It sets generation = 0 and prints the initial string to the Output Area.

Runtime phase (one generation per tick)

On each call to go:

If with-selection is ON (cumulative selection):

1. Increment generation.
2. Generate offspring-per-generation offspring strings by copying parent-string.
3. For each offspring, iterate through all character positions:
    - with probability mutation-rate, replace the character with a random element of allowed-chars
4. Compute a score for each offspring (defined below) and select the **best** offspring (lowest score).
5. Set parent-string to the best offspring.
6. Print the generation and current parent-string to the Output Area.

If with-selection is OFF (random change only):

1. Increment generation.
2. Iterate through all character positions of parent-string:
    - with probability mutation-rate, replace the character with a random element of allowed-chars
3. Print the generation and current parent-string to the Output Area.

Stopping condition

After each generation, the model computes a score for the current string. If the score equals 0 (exact match), the model stops and prints a summary statement including:

- number of generations required

- target-phrase
- mutation-rate
- offspring-per-generation
- whether selection was on or off

4. Design concepts

Basic principles

The model implements Dawkins's contrast between:

- **single-step random typing** (no retention of partial improvements), and
- **cumulative selection** (retention of partial improvements across generations).

The "fitness" concept is reduced to a single operational measure: similarity to a fixed target phrase.

Emergence

When selection is on, the model typically produces rapid convergence to the target phrase under intermediate parameter settings. Key emergent behaviors include:

- **No progress when mutation-rate = 0:** no variation is produced; selection has nothing to act on.
- **Slowed or unstable progress when mutation-rate is too high:** improvements are frequently overwritten.
- **Faster convergence with larger offspring-per-generation (when selection is on):** more candidate variants increase the chance of producing a better match in each generation.
- **Near-impossibility of convergence when selection is off for long phrases:** the expected time to match grows explosively with phrase length.

Adaptation

Adaptation is represented as a reduction in the number of mismatched characters between the current string and the target phrase over generations, due to selection retaining higher-matching offspring.

Objectives

The objective function is to minimize mismatch to the target phrase, computed as: score = number of positions where parent-string differs from target-phrase

Learning

No learning occurs. Strings change only through mutation and (optionally) selection.

Prediction

No prediction in the forecasting sense is performed; however, the model produces expected qualitative outcomes:

- With selection on and moderate mutation-rate, convergence typically occurs in relatively few generations.
- Without selection, convergence is extremely unlikely for longer phrases.

Sensing

Strings do not “sense” in an agent-based sense. The model algorithm computes similarity to the target phrase and uses it for selection.

Interaction

There is no interaction among strings; each candidate is compared only to the target phrase.

Stochasticity

Randomness enters in two places:

- whether each character mutates (random-float 1.0 < mutation-rate)
- the replacement character chosen from allowed-chars

Observation

The Output Area reports one line per generation: generation number and the current best string. A final summary statement is printed in the Command Center upon reaching an exact match.

5. Initialization

At setup:

- allowed-chars is set to "ABCDEFGHIJKLMNOPQRSTUVWXYZ ".
- target-phrase is normalized to use only allowed-chars and to be non-empty.
- parent-string is initialized as a random string of the same length as target-phrase, with each character sampled uniformly from allowed-chars.
- generation starts at 0.

6. Input data

No external input data files are used. Inputs are provided through the NetLogo interface (target-phrase, mutation-rate, offspring-per-generation, with-selection).

7. Submodels

Mutation submodel

For each character position, with probability mutation-rate, the character is replaced by a uniformly random character from allowed-chars.

Selection submodel (conditional on with-selection)

Among offspring-per-generation mutated offspring, the model selects the offspring with the lowest mismatch score relative to target-phrase. (If ties occur, the first encountered tied best is retained under the current implementation.)

Scoring submodel

Similarity is computed position-by-position across the whole string:

- each exact character match reduces the mismatch count by 1
- the final score equals the number of mismatches