UNIVERSITÉ CÔTE D'AZUR

Algorithmes Évolutionnaires (M2 MIAGE IA²)

Andrea G. B. Tettamanzi
Laboratoire I3S – Équipe SPARKS
andrea.tettamanzi@univ-cotedazur.fr

univ-cotedazur.fr

Séance 3
Le grandes familles du calcul évolutionniste

Plan

- Algorithmes génétiques
- Programmation évolutionnaire
- Stratégies d’évolution
- Programmation génétique

Plain Genetic Algorithms

- Encoding: bit strings
- Mutation: transcription error
- Recombination: crossover
- Selection:
  - Fitness-proportionate
  - Linear-ranking
- Replacement:
  - Generational replacement
  - Steady-state

Evolutionary Programming

- Purpose: to solve prediction tasks
- Individuals are finite-state automata
- Encoding: state-transition table
- Mutation: uniform random perturbation of entries
- Recombination: not used
- Selection:
  - tournament
  - truncation

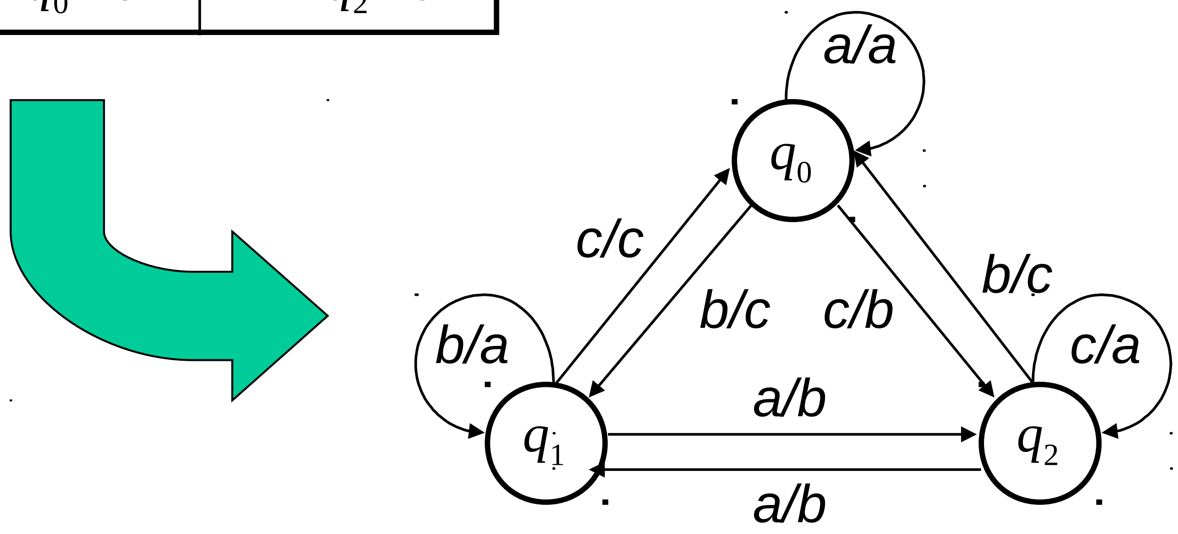
Evolutionary Programming: Individuals

Finite-state automaton: (Q, q_0, A, Σ, δ, ω)

- set of states Q;
- initial state q_0;
- set of accepting states A;
- alphabet of symbols Σ;
- transition function δ: $Q \times \Sigma \rightarrow Q$;
- output mapping function ω: $Q \times \Sigma \rightarrow \Sigma$;

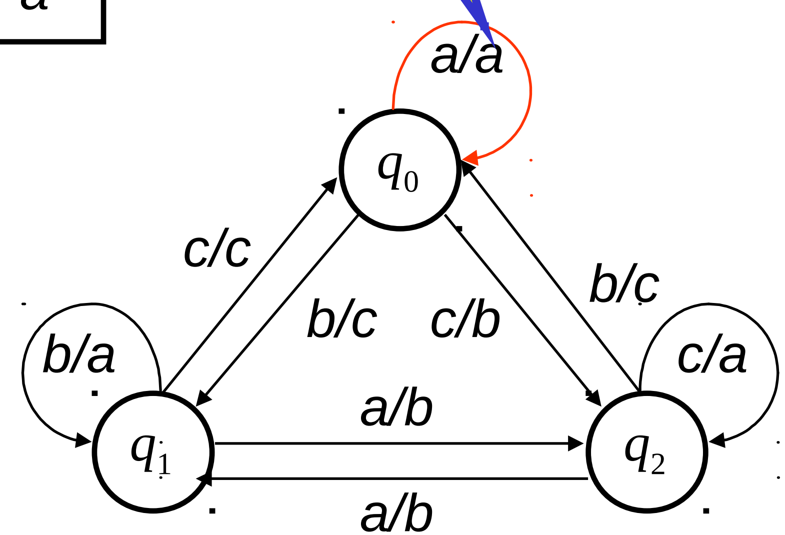
Evolutionary Programming: Encoding

input \ state	q_0	q_1	q_2
a	q_0 *a*	q_2 *b*	q_1 *b*
b	q_1 *c*	q_1 *a*	q_0 *c*
c	q_2 *b*	q_0 *c*	q_2 *a*



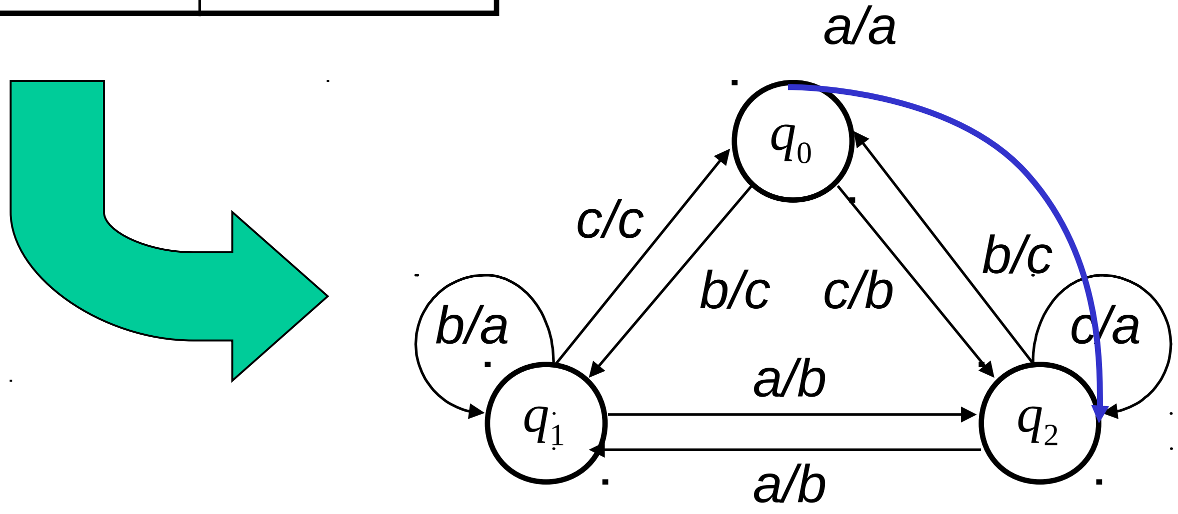
Evolutionary Programming: Mutation

state / input	q_0	q_1	q_2
a	q_0 *a*	q_2 *b*	q_1 *b*
b	q_1 *c*	q_1 *a*	q_0 *c*
c	q_2 *b*	q_0 *c*	q_2 *a*



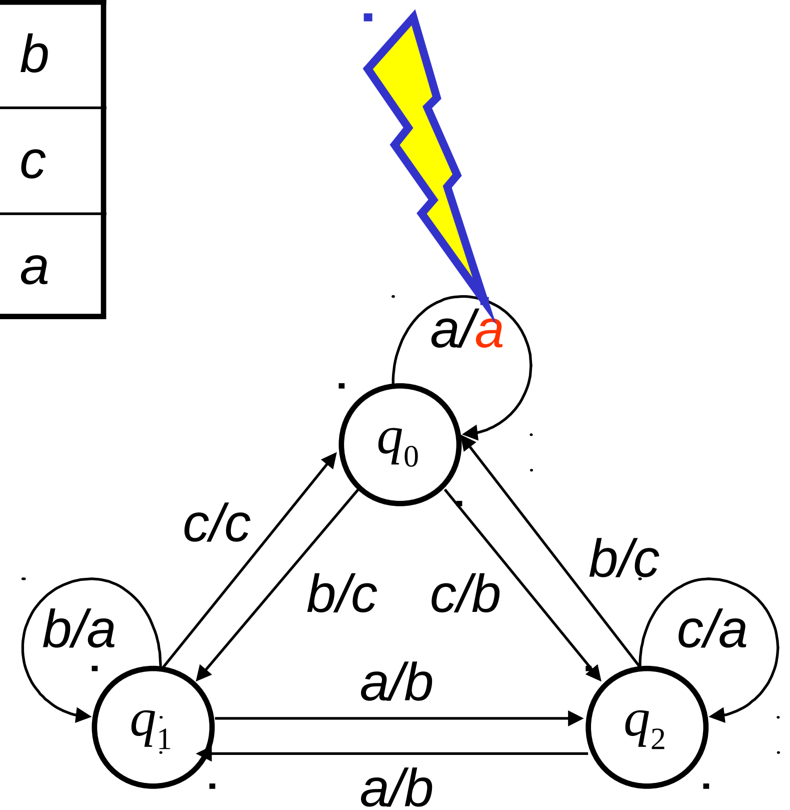
Evolutionary Programming: Mutation

state / input	q_0	q_1	q_2
a	$\mathbf{q_2}$ *a*	q_2 *b*	q_1 *b*
b	q_1 *c*	q_1 *a*	q_0 *c*
c	q_2 *b*	q_0 *c*	q_2 *a*



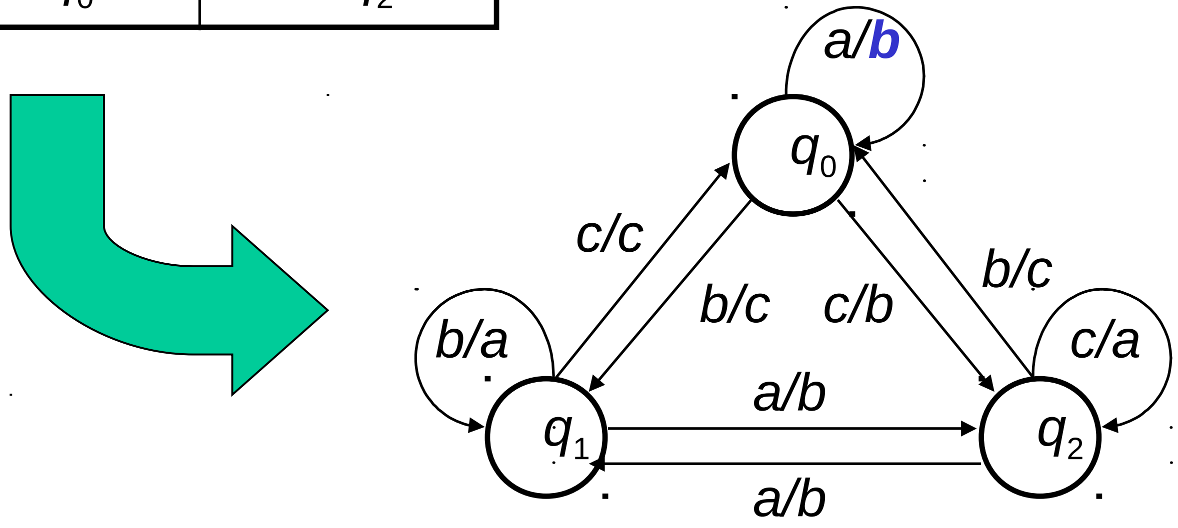
Evolutionary Programming: Mutation

state / input	q_0	q_1	q_2
a	q_0 *a*	q_2 *b*	q_1 *b*
b	q_1 *c*	q_1 *a*	q_0 *c*
c	q_2 *b*	q_0 *c*	q_2 *a*



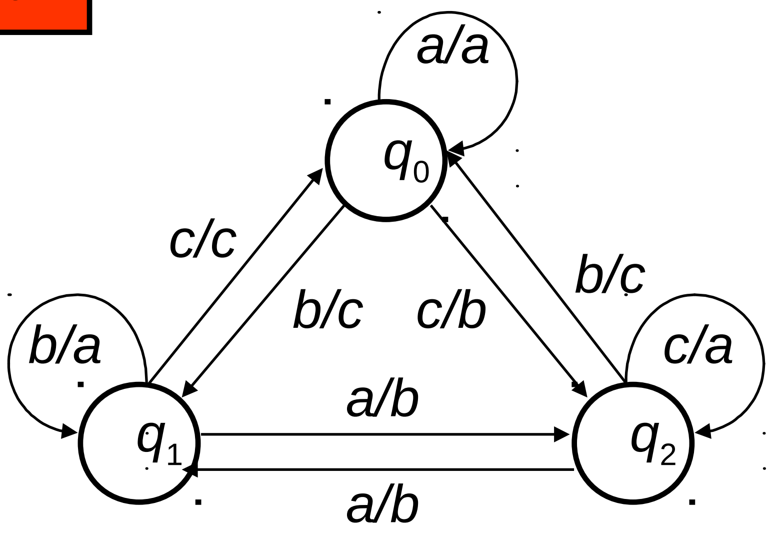
Evolutionary Programming: Mutation

state / input	q_0	q_1	q_2
a	q_0 ***b***	q_2 *b*	q_1 *b*
b	q_1 *c*	q_1 *a*	q_0 *c*
c	q_2 *b*	q_0 *c*	q_2 *a*



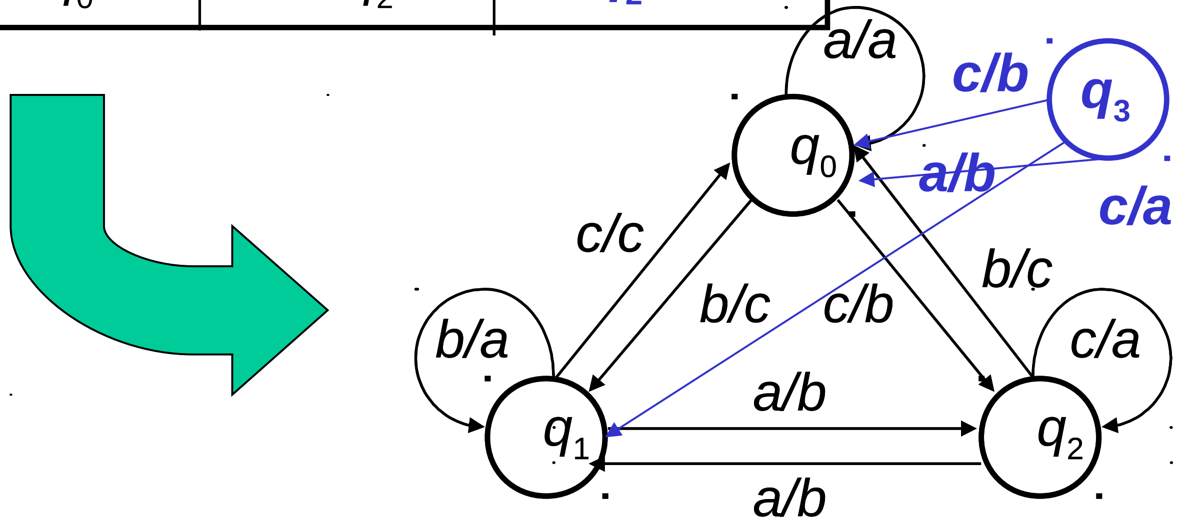
Evolutionary Programming: Mutation

input \ state	q_0	q_1	q_2
a	q_0 *a*	q_2 *b*	q_1 *b*
b	q_1 *c*	q_1 *a*	q_0 *c*
c	q_2 *b*	q_0 *c*	q_2 *a*

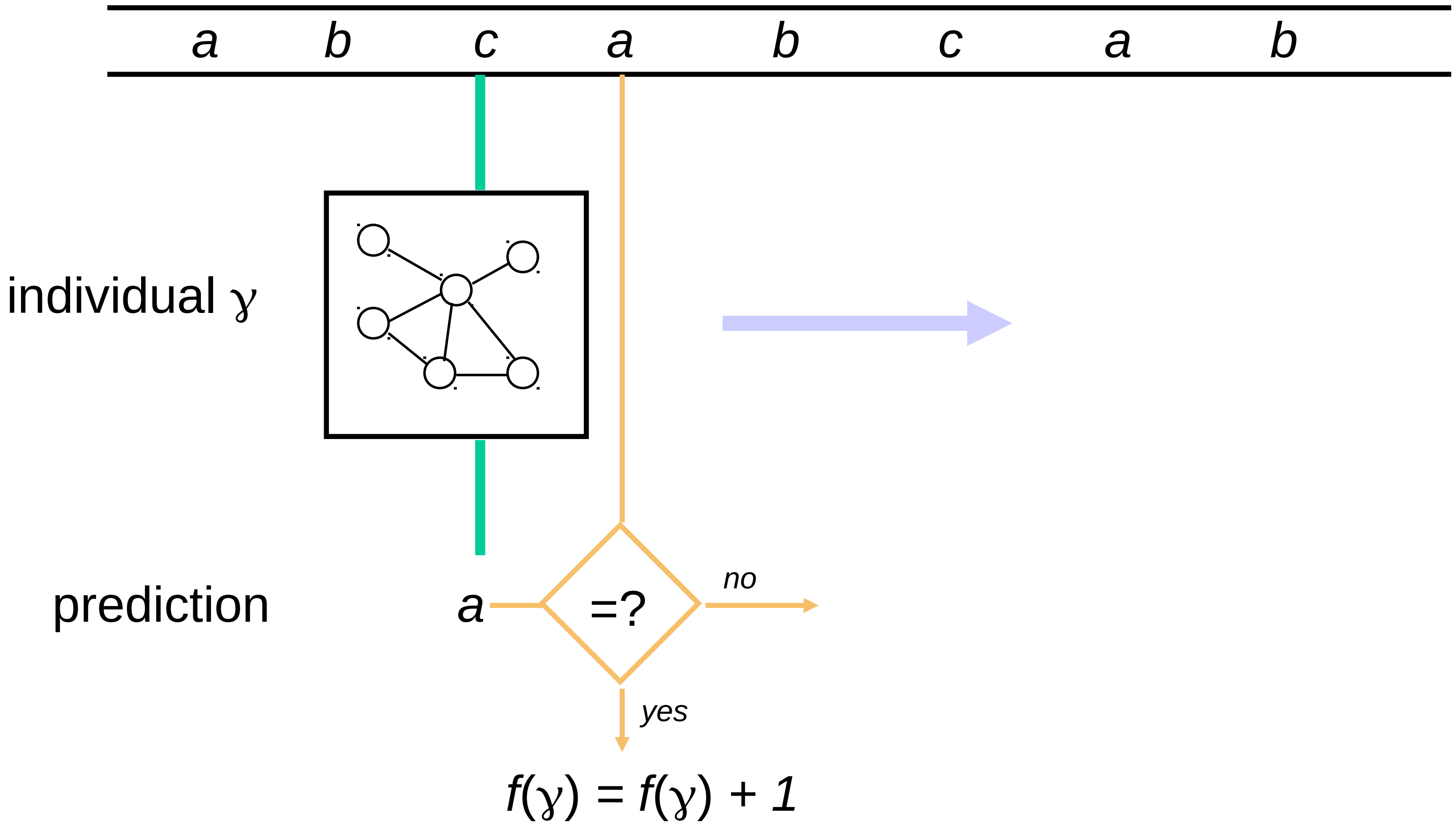


Evolutionary Programming: Mutation

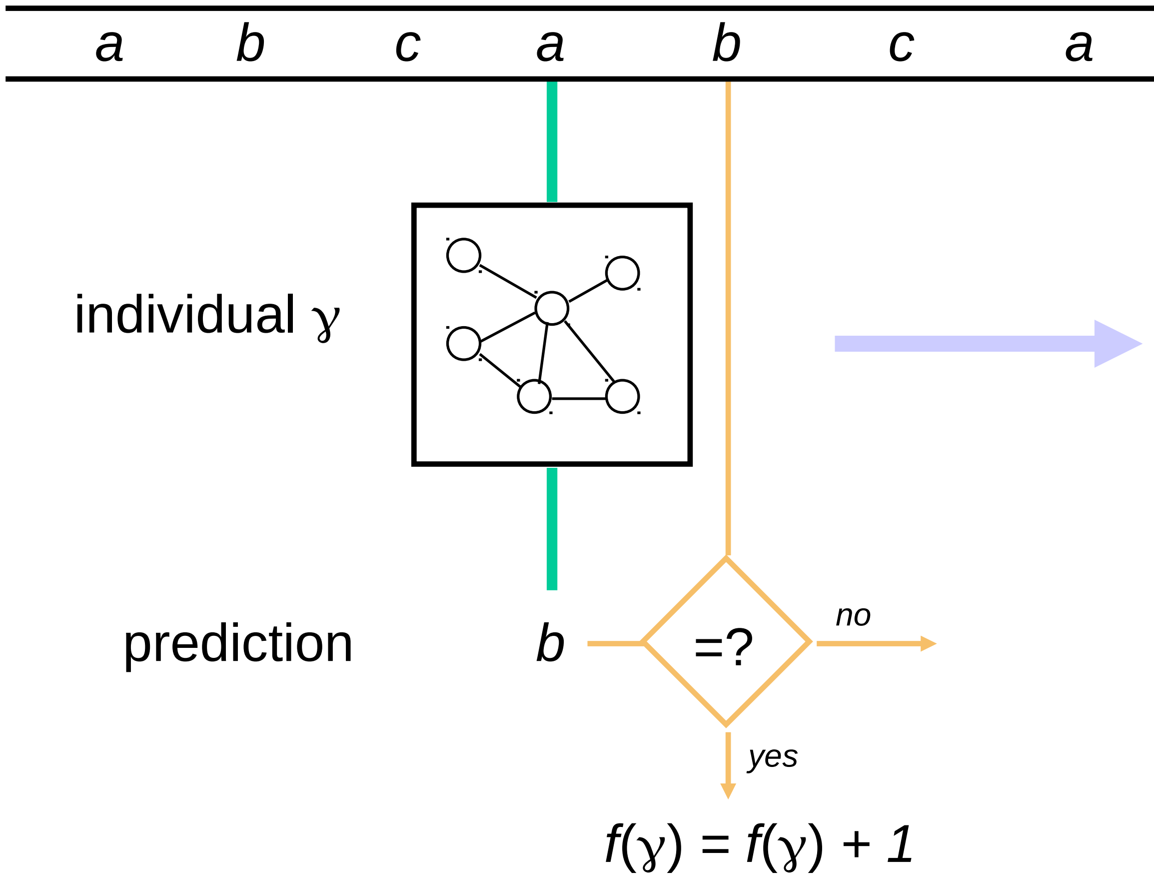
state / input	q_0	q_1	q_2	q_3
a	q_0 a	q_2 b	q_1 b	q_1 b
b	q_1 c	q_1 a	q_0 c	q_0 c
c	q_2 b	q_0 c	q_2 a	q_2 a



Evolutionary Programming: Fitness

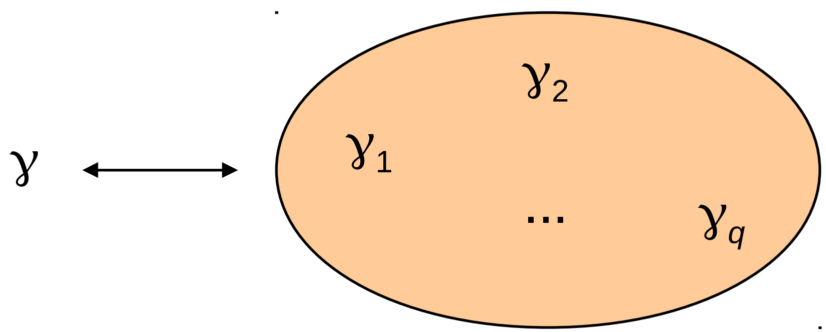


Evolutionary Programming: Fitness



Evolutionary Programming: Selection

Variant of stochastic *q*-tournament selection:



$\text{score}(\gamma) = \#\{\gamma_i \mid f(\gamma) > f(\gamma_i)\}$

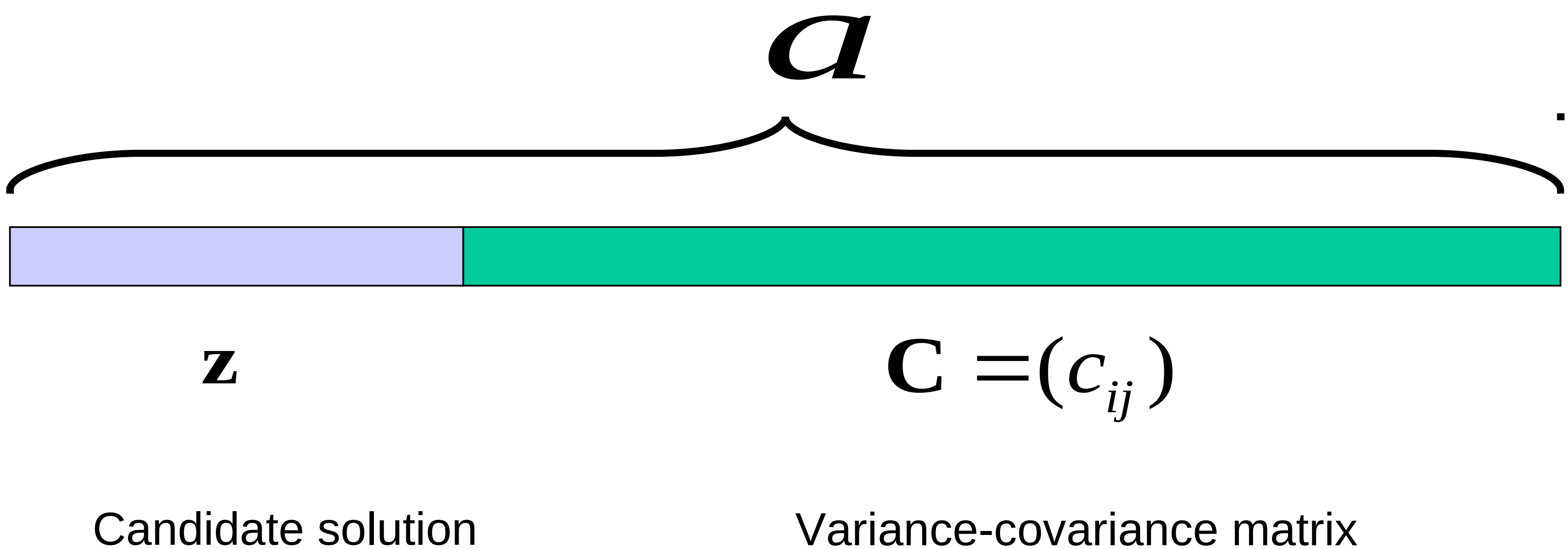
Order individuals by decreasing score
Select first half (Truncation selection)

Evolution Strategies

- Encoding: *n*-dimensional vectors of reals
- Mutation: Gaussian perturbation
- Self-adaptation: mutation distribution in the genotype

  (standard deviations and covariances evolve with solutions)
- Recombination:
  - Discrete, intermediate
  - multi-parent
- Selection: truncation
- Fitness is the objective function

Evolution Strategies: Encoding



Evolution Strategies: Encoding

l-dimensional normal joint distribution (for mutation)

$$p(\mathbf{z}) = \sqrt{\frac{\det \mathbf{C}^{-1}}{(2\pi)^l}}\, e^{-\frac{1}{2}\mathbf{z}^T \mathbf{C}^{-1} \mathbf{z}}$$

Evolution Strategies: Mutation

$$\gamma = \langle \mathbf{z}, \mathbf{C} \rangle \to \gamma' = \langle \mathbf{z}', \mathbf{C}' \rangle$$

where $$\mathbf{z}' = \mathbf{z} + N(\mathbf{0}, \mathbf{C}')$$

Elementary Rotation Matrix

$$\mathbf{R}_{ij}(\alpha) = \begin{array}{cc}
\begin{bmatrix}
1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\
0 & \ddots & & & & ⋰ & \vdots \\
\vdots & & \cos\alpha & & -\sin\alpha & & \vdots \\
\vdots & & & \ddots & & & \vdots \\
\vdots & & \sin\alpha & & \cos\alpha & & \vdots \\
\vdots & ⋰ & & & & \ddots & 0 \\
0 & \cdots & \cdots & \cdots & \cdots & 0 & 1
\end{bmatrix} & \begin{matrix} \\ \\ i \\ \\ j \\ \\ \\ \end{matrix} \\
\begin{matrix} & & i & & j & & \end{matrix} &
\end{array}$$

Rotation Angles

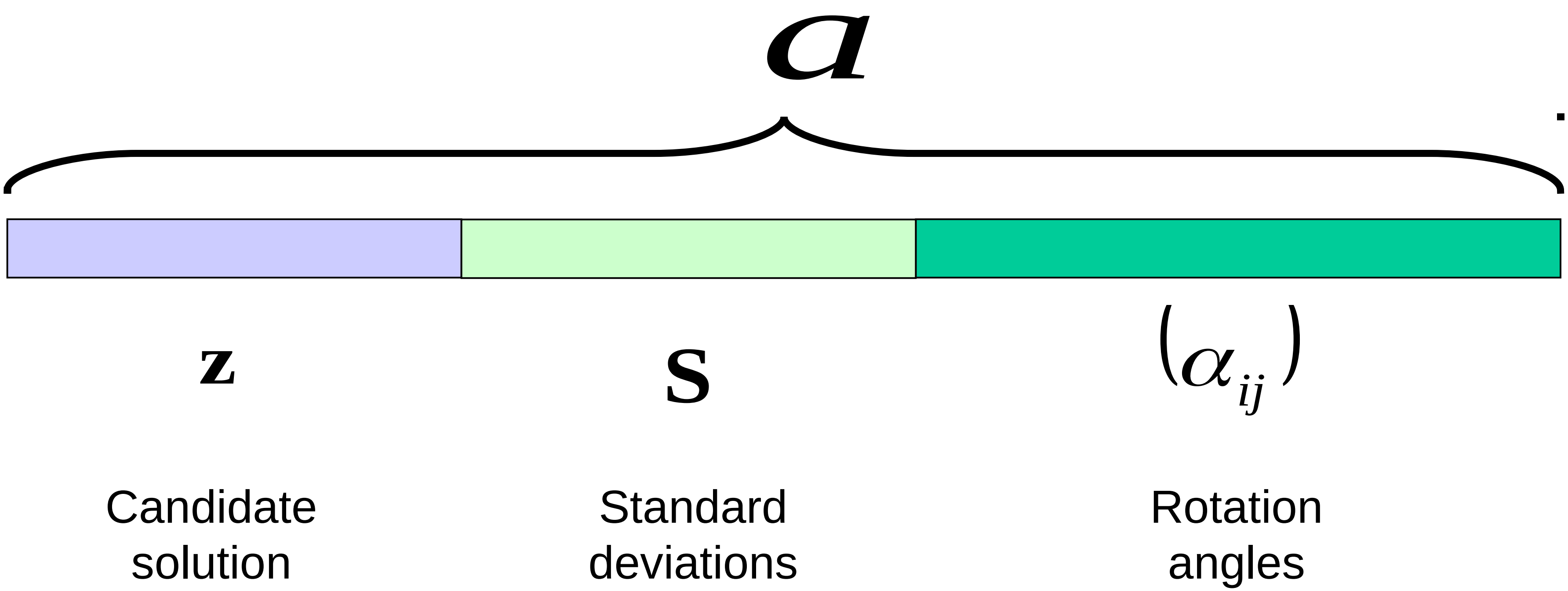
- Modifying **C** directly would yield invalid matrices
- Use the decomposition

$$\mathbf{C} = (\mathbf{ST})^T (\mathbf{ST})$$

$$\mathbf{S} = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \sigma_l \end{bmatrix} \qquad \mathbf{T} = \prod_{i=1}^{l-1} \prod_{j=i+1}^{l} \mathbf{R}_{ij}(\alpha_{ij})$$

$$\alpha_{ij} \in (0, 2\pi)$$

Evolution Strategies: Actual Encoding



Evolution Strategies: Mutation

$$\sigma_i' = \sigma_i \exp(\tau' N(0,1) + \tau N_i(0,1))$$

$$\alpha_{ij}' = \alpha_{ij} + \beta N_{ij}(0,1)(\bmod 2\pi)$$

self-adaptation

$$\mathbf{z}' = \mathbf{z} + \mathbf{T}^T(\alpha') \cdot \mathbf{S}^T(\sigma') \cdot \mathbf{N}(\mathbf{0}, \mathbf{I})$$

Hans-Paul Schwefel suggests:

$$\tau \propto \left(\sqrt{2\sqrt{n}}\right)^{-1}$$

$$\tau' \propto \left(\sqrt{2n}\right)^{-1}$$

$$\beta \approx 0.0873 = 5^\circ$$

Evolution Strategies: Recombination

- Discrete recombination: each component in the child is copied from either parent
- Intermediate recombination: each component in the child is a linear combination of the corresponding components in the parents
  - Internal: linear combination in the convex hull
  - External: linear combination out of the convex hull
- Best results:
  - Discrete recombination for object problem parameters
  - Intermediate recombination for strategy parameters

Evolution Strategies: Selection and Replacement

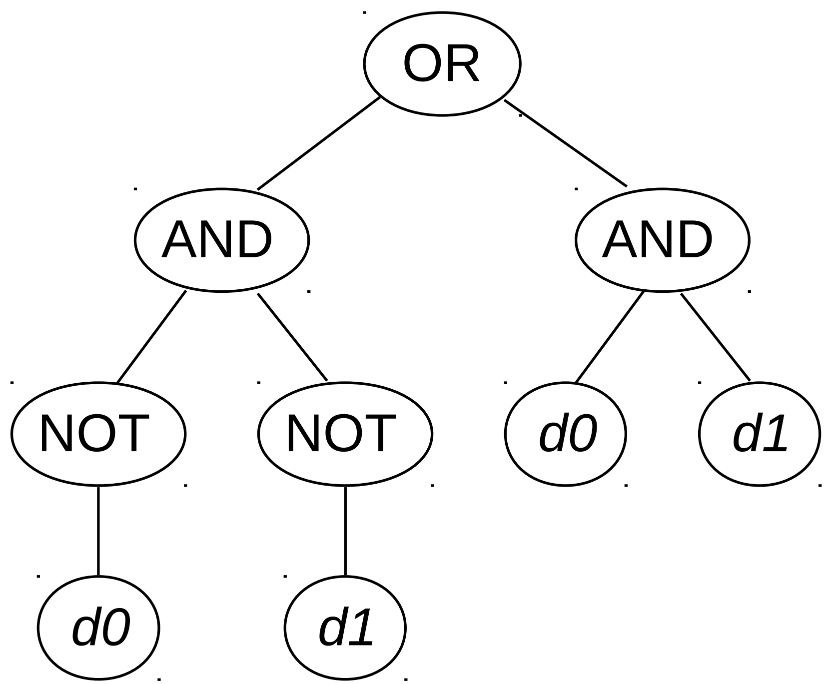
- (n, m) strategies:
  - $m>n$ offspring are produced;
  - The best n of them replace the previous population.
- $(n + m)$ strategies:
  - $m>n$ offspring are produced;
  - They are injected into the current population;
  - The best n individuals in the population survive.

Genetic Programming

- Program induction
- LISP (historically), math expressions, machine language, ...
- Applications:
  - optimal control;
  - planning;
  - sequence induction;
  - symbolic regression;
  - modeling and forecasting;
  - symbolic integration and differentiation;
  - inverse problems

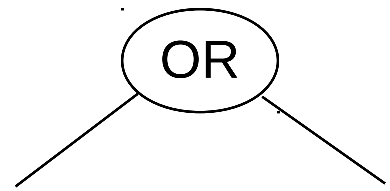
Genetic Programming: The Individuals

subset of LISP S-expressions

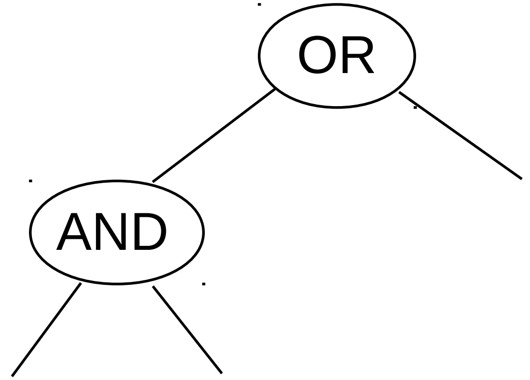


(OR (AND (NOT *d0*) (NOT *d1*)) (AND *d0 d1*))

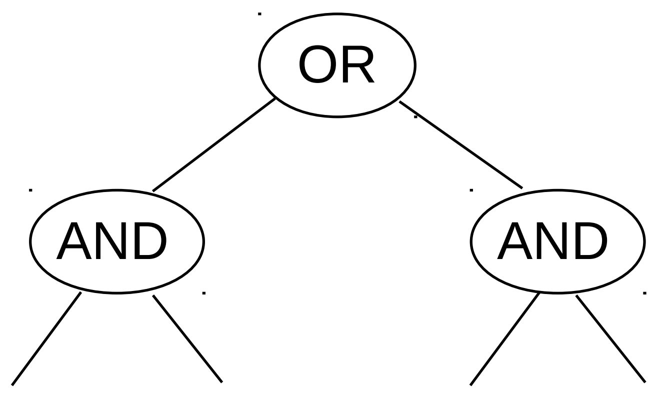
Genetic Programming: Initialization



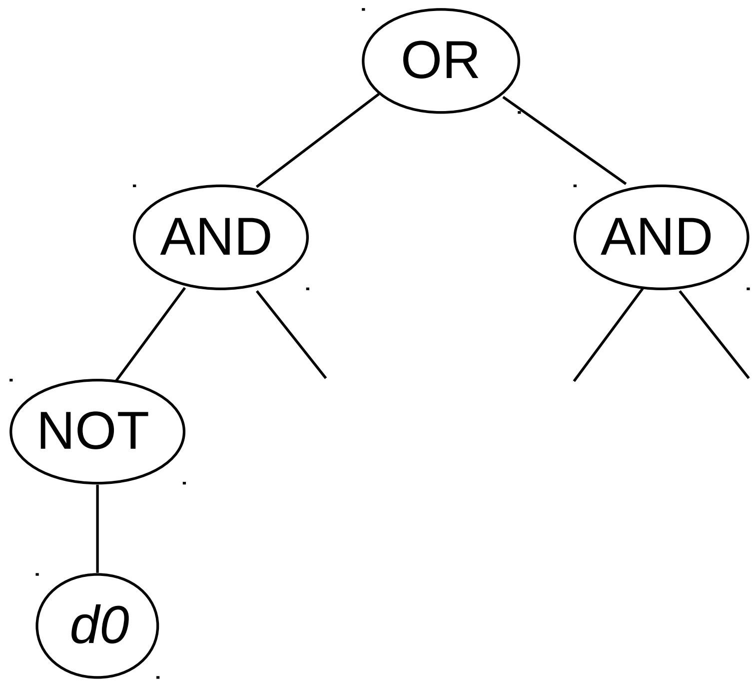
Genetic Programming: Initialization



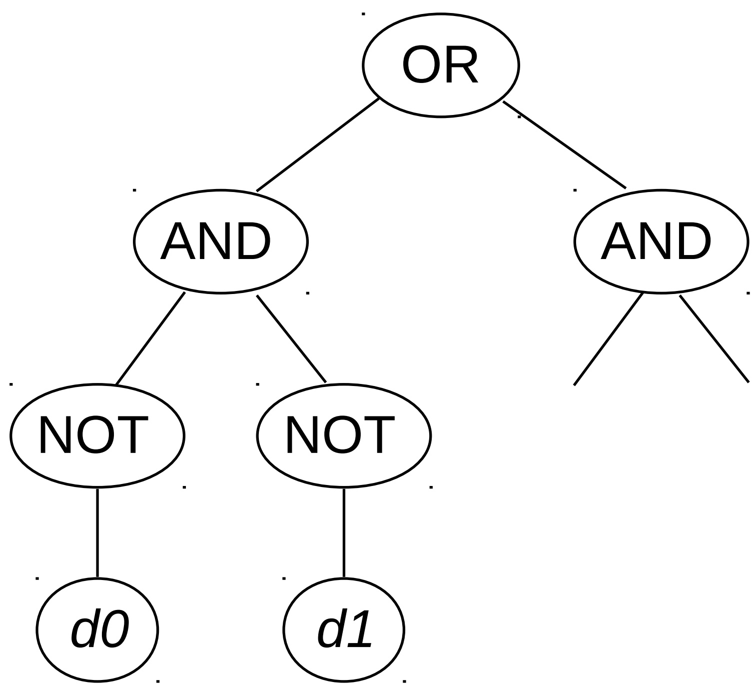
Genetic Programming: Initialization



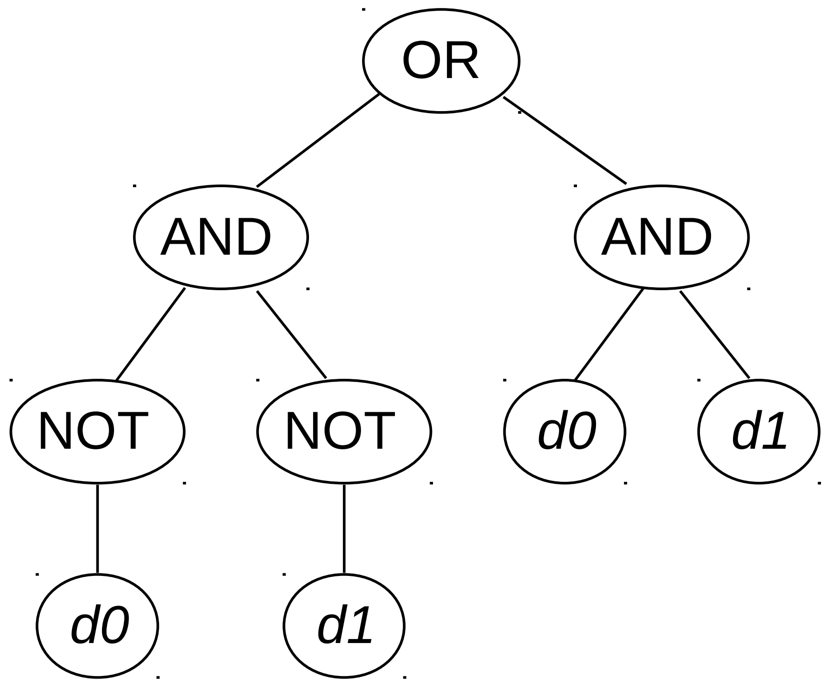
Genetic Programming: Initialization



Genetic Programming: Initialization



Genetic Programming: Initialization



Genetic Programming: Fitness

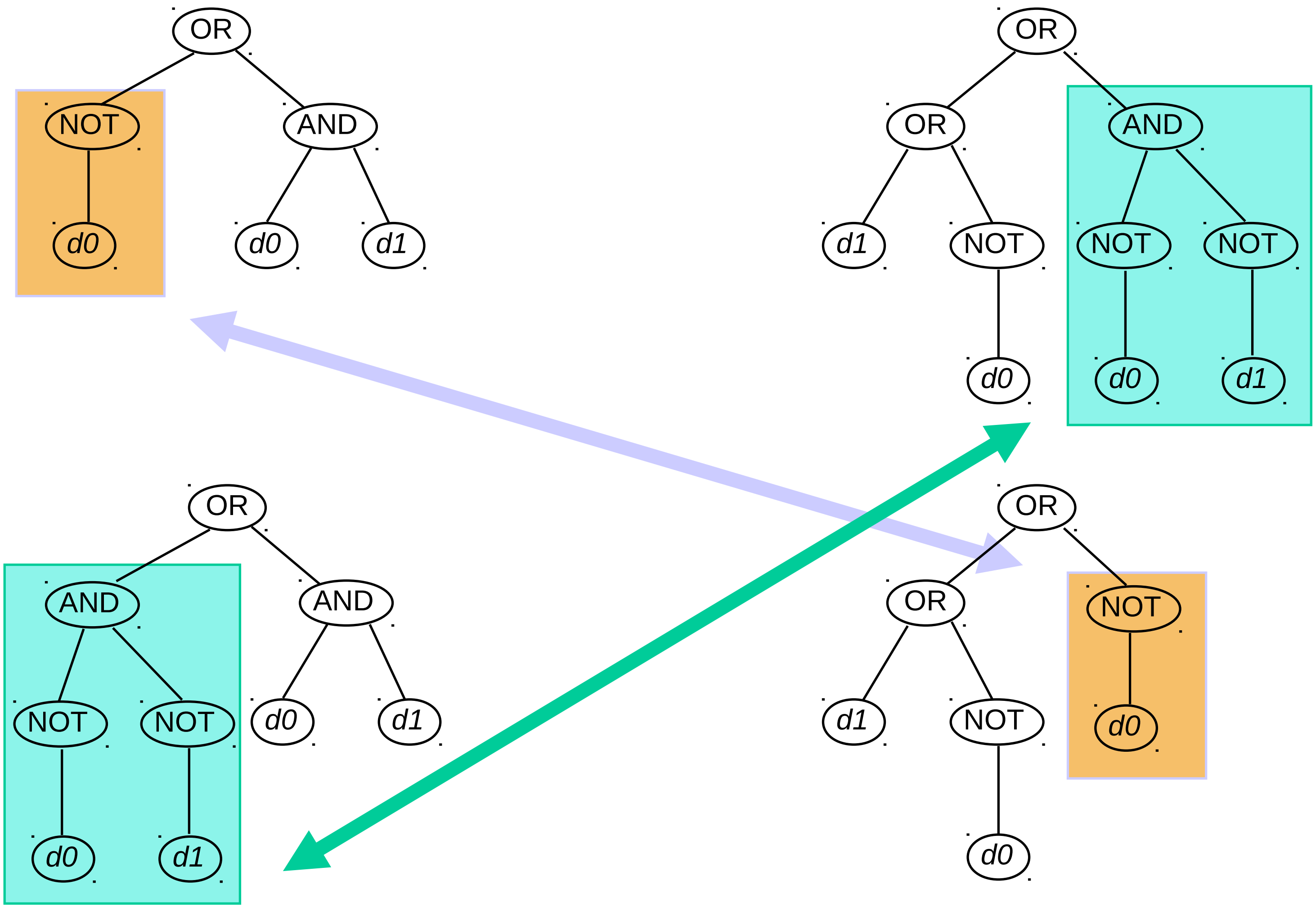
Fitness cases: $j = 1, ..., N_e$

"Raw" fitness: $$r(\gamma) = \sum_{j=1}^{N_e} \left| \mathrm{Output}(\gamma, j) - C(j) \right|$$

"Standardized" fitness: $s(\gamma) \in [0, +\infty)$

"Adjusted" fitness: $$a(\gamma) = \frac{1}{1 + s(\gamma)}$$

Genetic Programming: Crossover



Genetic Programming: Other Operators

- Mutation: replace a terminal with a subtree
- Permutation: change the order of arguments to a function
- Editing: simplify S-expressions, e.g. (AND X X) $\rightarrow$ X
- Encapsulation: define a new function using a subtree
- Decimation: throw away most of the population