Evolution strategies

Javad Salimi

Fall 2019

ES quick overview

- Developed: Germany in the 1970's
- Early names: I. Rechenberg, H.-P. Schwefel
- Typically applied to:
  - numerical optimisation
- Attributed features:
  - fast
  - good optimizer for real-valued optimisation
  - relatively much theory
- Special:
  - self-adaptation of (mutation) parameters standard

ES technical summary tableau

Representation	Real-valued vectors
Recombination	Discrete or intermediary
Mutation	Gaussian perturbation
Parent selection	Uniform random
Survivor selection	(μ,λ) or $(\mu+\lambda)$
Specialty	Self-adaptation of mutation step sizes

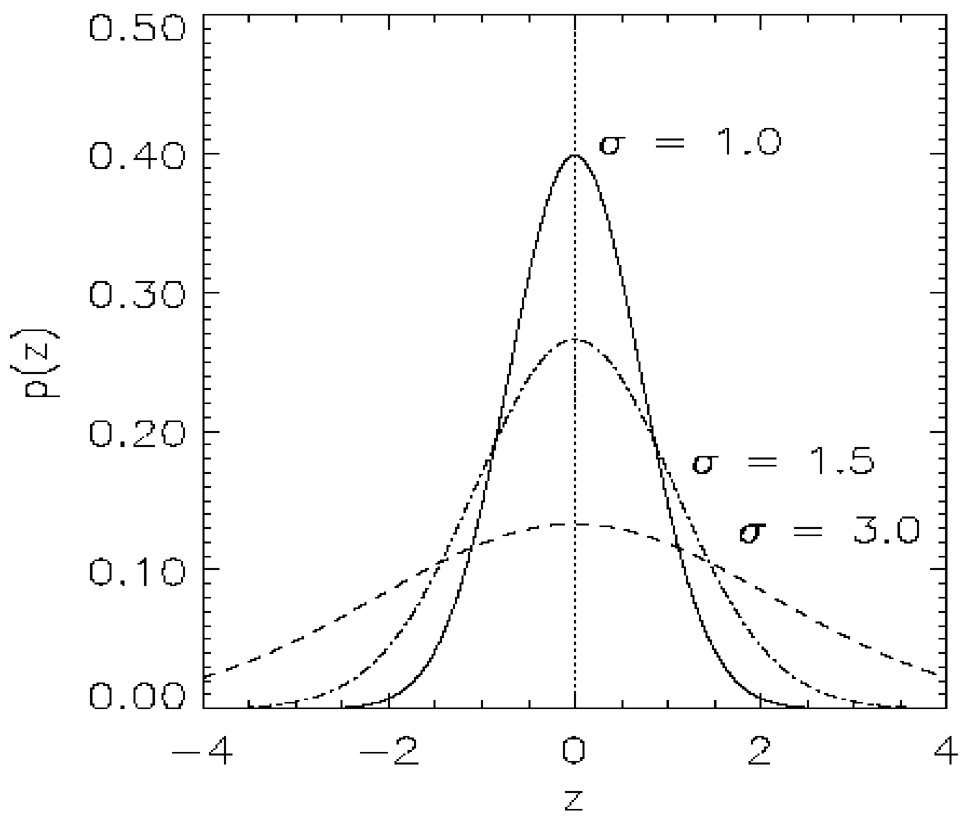
Introductory example: pseudo code

```
• Set $t = 0$
• Create initial point $x^t = \langle x_1^t,\ldots,x_n^t \rangle$
• REPEAT UNTIL (*TERMIN.COND* satisfied) DO
    – Draw $z_i$ from a normal distr. for all $i = 1,\ldots,n$
    – $y_i^t = x_i^t + z_i$
    – IF $f(x^t) < f(y^t)$ THEN $x^{t+1} = x^t$
    – ELSE $x^{t+1} = y^t$
    – FI
    – Set $t = t+1$
• OD
```

Introductory example: mutation mechanism

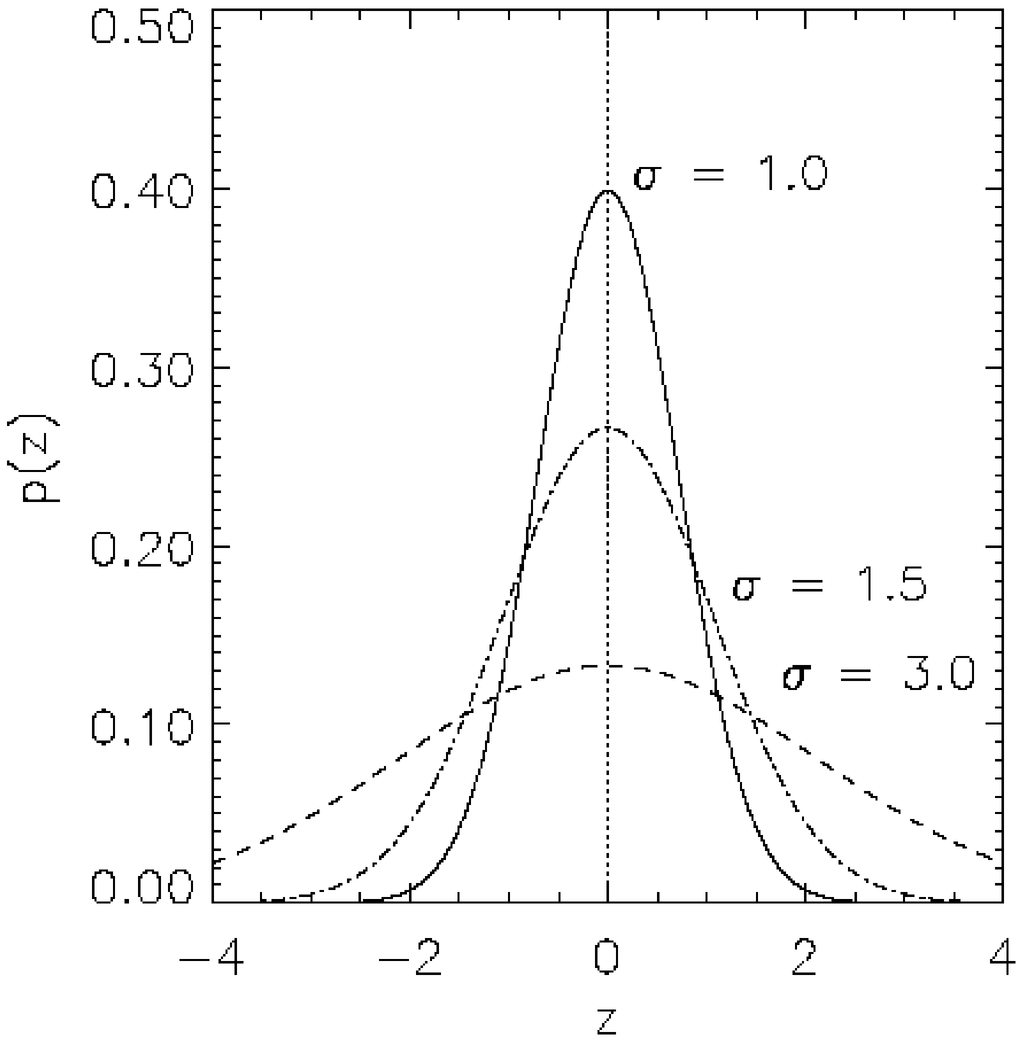
- z values drawn from normal distribution $N(\xi,\sigma)$
  - mean ξ is set to 0
  - variation σ is called mutation step size
- σ is varied on the fly by the "1/5 success rule":
- This rule resets σ after every k iterations by
  - $\sigma = \sigma / c$ if $p_s > 1/5$
  - $\sigma = \sigma \cdot c$ if $p_s < 1/5$
  - $\sigma = \sigma$ if $p_s = 1/5$
- where p_s is the % of successful mutations, $0.8 \leq c \leq 1$

Illustration of normal distribution



Genetic operators: mutations (2)

The one dimensional case



Representation

- Chromosomes consist of three parts:
  - Object variables: $x_1,\ldots,x_n$
  - Strategy parameters:
    - Mutation step sizes: $\sigma_1,\ldots,\sigma_{n_\sigma}$
    - Rotation angles: $\alpha_1,\ldots, \alpha_{n_\alpha}$
- Not every component is always present
- Full size: $\langle x_1,\ldots,x_n, \sigma_1,\ldots,\sigma_n ,\alpha_1,\ldots, \alpha_k \rangle$
- where k = n(n-1)/2 (no. of i,j pairs)

Mutation

- Main mechanism: changing value by adding random noise drawn from normal distribution
- $x'_i = x_i + N(0,\sigma)$
- Key idea:
  - σ is part of the chromosome $\langle x_1,\ldots,x_n, \sigma \rangle$
  - σ is also mutated into σ' (see later how)
- Thus: mutation step size σ is coevolving with the solution x

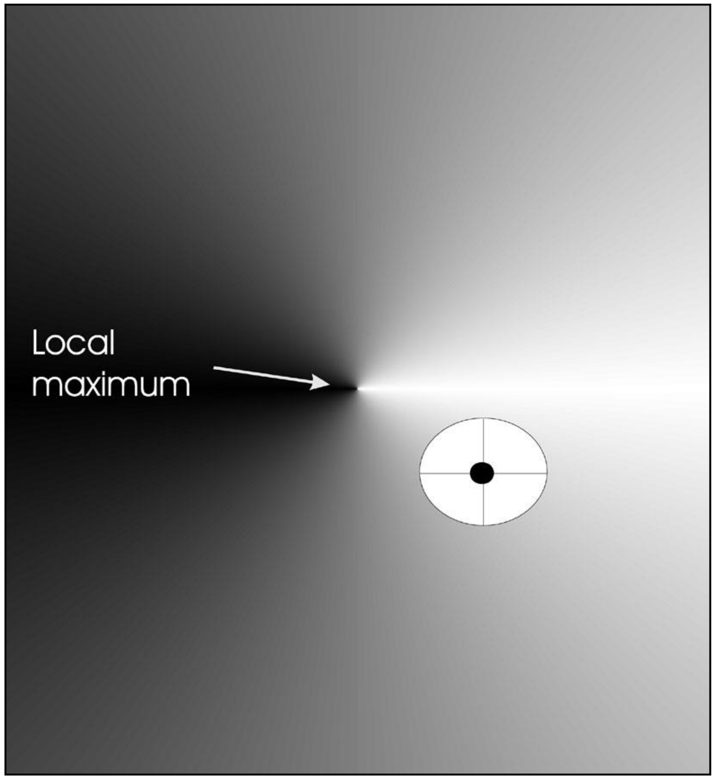
Mutate σ first

- Net mutation effect: $\langle x, \sigma \rangle \rightarrow \langle x', \sigma' \rangle$
- Order is important:
  – first $\sigma \rightarrow \sigma'$ (see later how)
  – then $x \rightarrow x' = x + N(0,\sigma')$
- Rationale: new $\langle x', \sigma' \rangle$ is evaluated twice
  – Primary: x' is good if f(x') is good
  – Secondary: σ' is good if the x' it created is good
- Reversing mutation order this would not work

Mutation case 1: Uncorrelated mutation with one σ

- Chromosomes: $\langle\, x_1, \ldots, x_n, \sigma \,\rangle$
- $\sigma' = \sigma \cdot \exp(\tau \cdot N(0,1))$
- $x'_i = x_i + \sigma' \cdot N(0,1)$
- Typically the “learning rate” $\tau \propto 1/ n^{½}$
- And we have a boundary rule $\sigma' < \varepsilon_0 \Rightarrow \sigma' = \varepsilon_0$

Mutants with equal likelihood

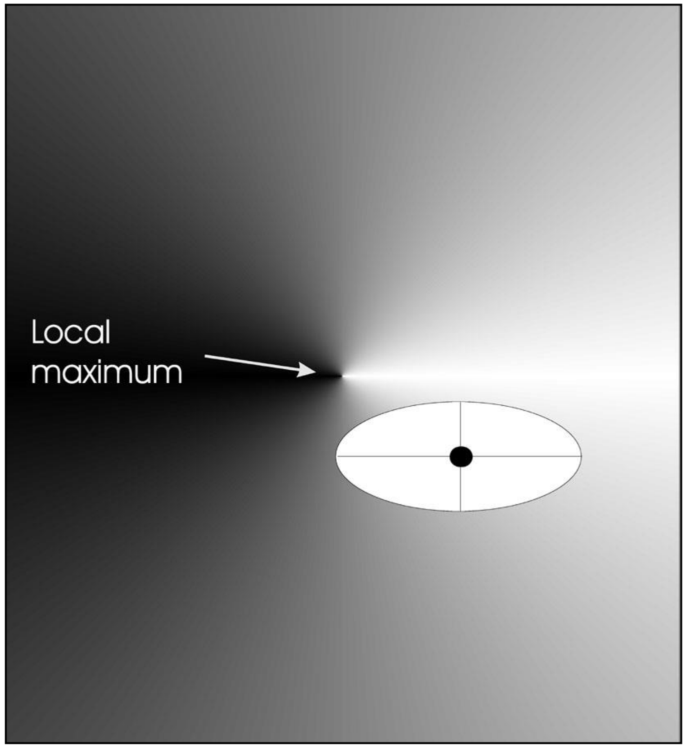


Circle: mutants having the same chance to be created

Mutation case 2: Uncorrelated mutation with n σ's

- Chromosomes: $\langle x_1,\dots,x_n, \sigma_1,\dots, \sigma_n \rangle$
- $\sigma'_i = \sigma_i \cdot \exp(\tau' \cdot N(0,1) + \tau \cdot N_i(0,1))$
- $x'_i = x_i + \sigma'_i \cdot N_i(0,1)$
- Two learning rate parameters:
  - τ' overall learning rate
  - τ coordinate wise learning rate
- $\tau' \propto 1/(2n)^{½}$ and $\tau \propto 1/(2n^{½})^{½}$
- And $\sigma_i' < \varepsilon_0 \Rightarrow \sigma_i' = \varepsilon_0$

Mutants with equal likelihood



Ellipse: mutants having the same chance to be created

Mutation case 3: Correlated mutations

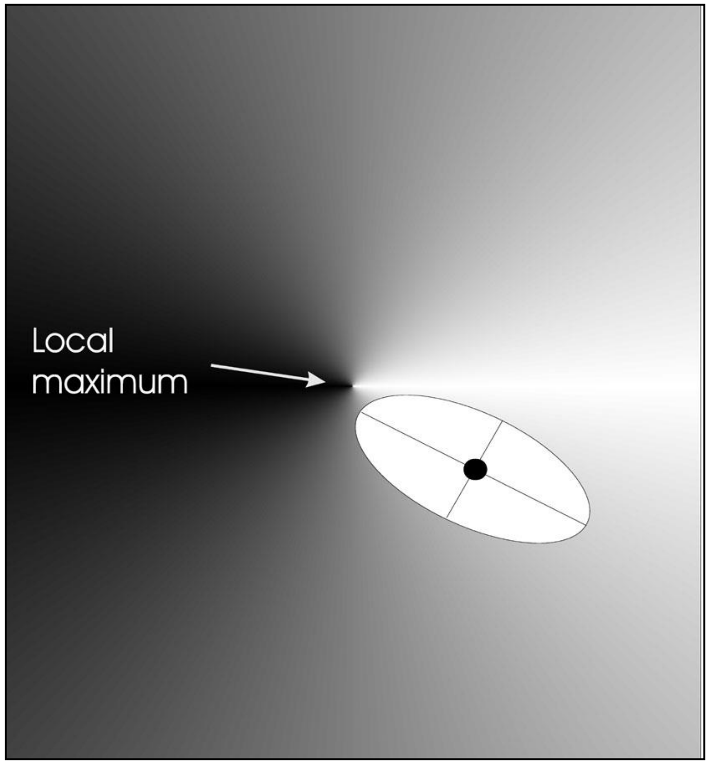
- Chromosomes: $\langle x_1,\ldots,x_n, \sigma_1,\ldots, \sigma_n ,\alpha_1,\ldots, \alpha_k \rangle$
- where $k = n \cdot (n-1)/2$
- and the covariance matrix C is defined as:
  - $c_{ii} = \sigma_i^2$
  - $c_{ij} = 0$ if i and j are not correlated
  - $c_{ij} = ½ \cdot (\sigma_i^2 - \sigma_j^2) \cdot \tan(2\, \alpha_{ij})$ if i and j are correlated
- Note the numbering / indices of the α's

Correlated mutations cont'd

The mutation mechanism is then:

- $\sigma'_i = \sigma_i \cdot \exp(\tau' \cdot N(0,1) + \tau \cdot N_i\,(0,1))$
- $\alpha'_j = \alpha_j + \beta \cdot N\,(0,1)$
- $\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{N}(\boldsymbol{0},\boldsymbol{C'})$
  - ***x*** stands for the vector $\langle x_1,\dots,x_n \rangle$
  - ***C'*** is the covariance matrix ***C*** after mutation of the α values
- $\tau' \propto 1/(2\,n)^{½}$ and $\tau \propto 1/(2\,n^{½})^{½}$ and $\beta \approx 5°$
- $\sigma_i' < \varepsilon_0 \Rightarrow \sigma_i' = \varepsilon_0$ and

Mutants with equal likelihood



Ellipse: mutants having the same chance to be created

Recombination

- Creates one child
- Acts per variable / position by either
  - Averaging parental values, or
  - Selecting one of the parental values
- From two or more parents by either:
  - Using two selected parents to make a child
  - Selecting two parents for each position anew

Names of recombination's

	Two fixed parents	Two parents selected for each i
$z_i = (x_i + y_i)/2$	Local intermediary	Global intermediary
z_i is x_i or y_i chosen randomly	Local discrete	Global discrete

Parent selection

- Parents are selected by uniform random distribution whenever an operator needs one/some
- Thus: ES parent selection is unbiased - every individual has the same probability to be selected
- Note that in ES “parent” means a population member.

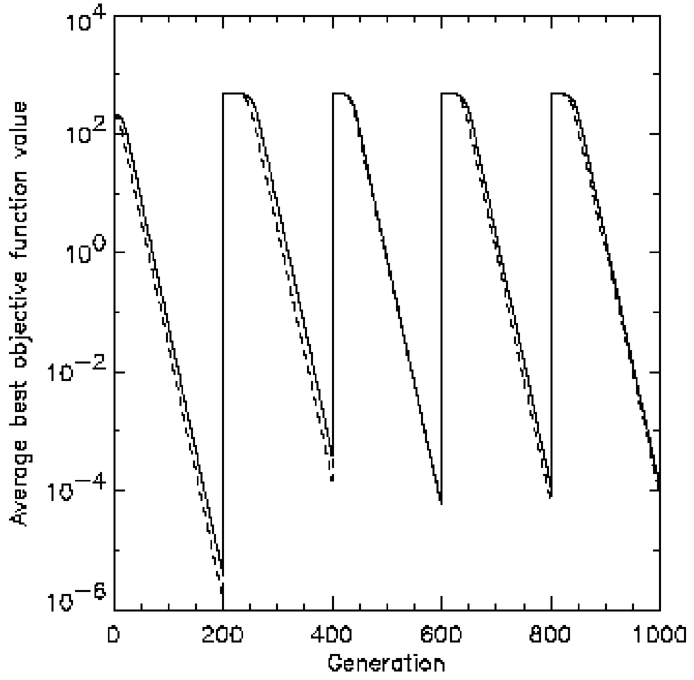
Survivor selection

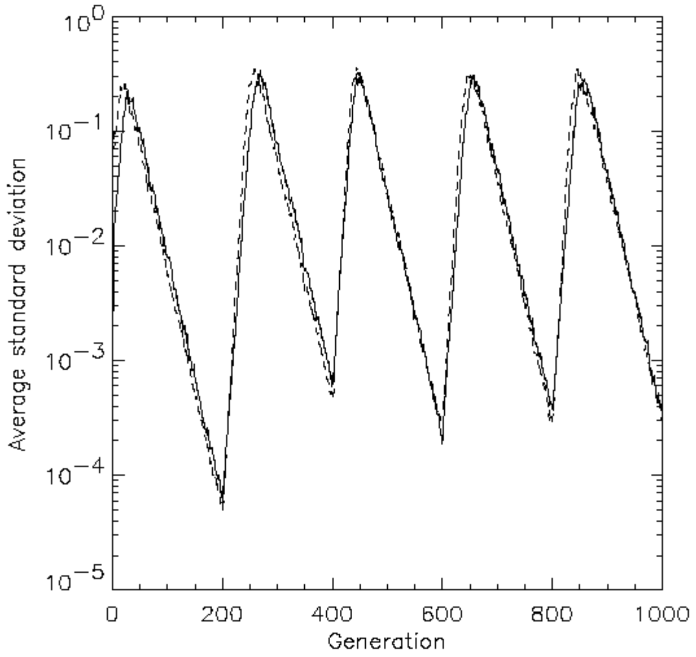
- (μ+λ)-selection is an elitist strategy
- (μ,λ)-selection can “forget”
- Often (μ,λ)-selection is preferred for:
  - Better in leaving local optima
  - Better in following moving optima
- Selective pressure in ES is very high ($\lambda \approx 7 \cdot \mu$ is the common setting)

Self-adaptation illustrated

- Given a dynamically changing fitness landscape (optimum location shifted every 200 generations)
- Self-adaptive ES is able to
  - follow the optimum and
  - adjust the mutation step size after every shift !

Self-adaptation illustrated cont'd





Changes in the fitness values (left) and the mutation step sizes (right)

Prerequisites for self-adaptation

- $\mu > 1$ to carry different strategies
- $\lambda > \mu$ to generate offspring surplus
- (μ,λ)-selection to get rid of miss adapted σ's
- Mixing strategy parameters by (intermediary) recombination on them

Example application: the Ackley function (Bäck et al '93)

- The Ackley function (here used with n =30):

$$f(x) = -20 \cdot \exp\left(-0.2\sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$$

- Evolution strategy:
  - Representation:
    - $-30 < x_i < 30$ (coincidence of 30's!)
    - 30 step sizes
  - (30,200) selection
  - Termination : after 200000 fitness evaluations
  - Results: average best solution is $7.48 \cdot 10^{-8}$ (very good)

Genetic Programming

GP quick overview

- Developed: USA in the 1990's
- Early names: J. Koza
- Typically applied to:
  - machine learning tasks (prediction, classification…)
- Attributed features:
  - competes with neural nets and alike
  - needs huge populations (thousands)
  - slow
- Special:
  - non-linear chromosomes: trees, graphs
  - mutation possible but not necessary (dispute!)

GP technical summary tableau

Representation	Tree structures
Recombination	Exchange of subtrees
Mutation	Random change in trees
Parent selection	Fitness proportional
Survivor selection	Generational replacement

Introductory example: credit scoring

- Bank wants to distinguish good from bad loan applicants
- Model needed that matches historical data

ID	No of children	Salary	Marital status	OK?
ID-1	2	45000	Married	0
ID-2	0	30000	Single	1
ID-3	1	40000	Divorced	1
…				

Introductory example: credit scoring

- A possible model:

IF (NOC = 2) AND (S > 80000) THEN good ELSE bad

- In general:

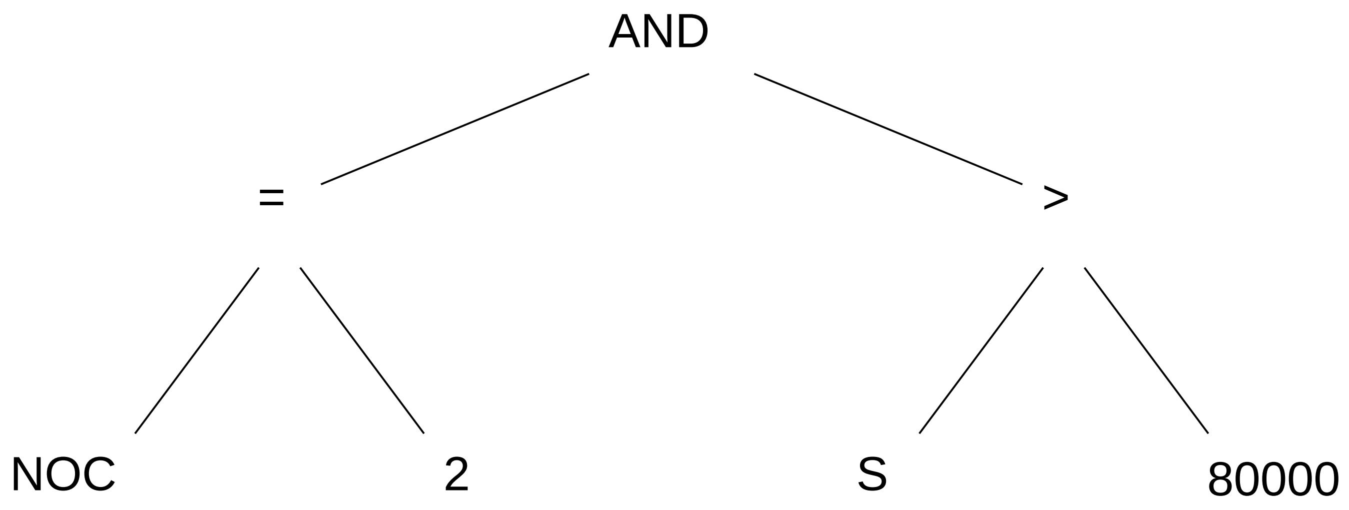
IF formula THEN good ELSE bad

- Only unknown is the right formula, hence
- Our search space (phenotypes) is the set of formulas
- Natural fitness of a formula: percentage of well classified cases of the model it stands for
- Natural representation of formulas (genotypes) is: parse trees

Introductory example: credit scoring

IF (NOC = 2) AND (S > 80000) THEN good ELSE bad

can be represented by the following tree

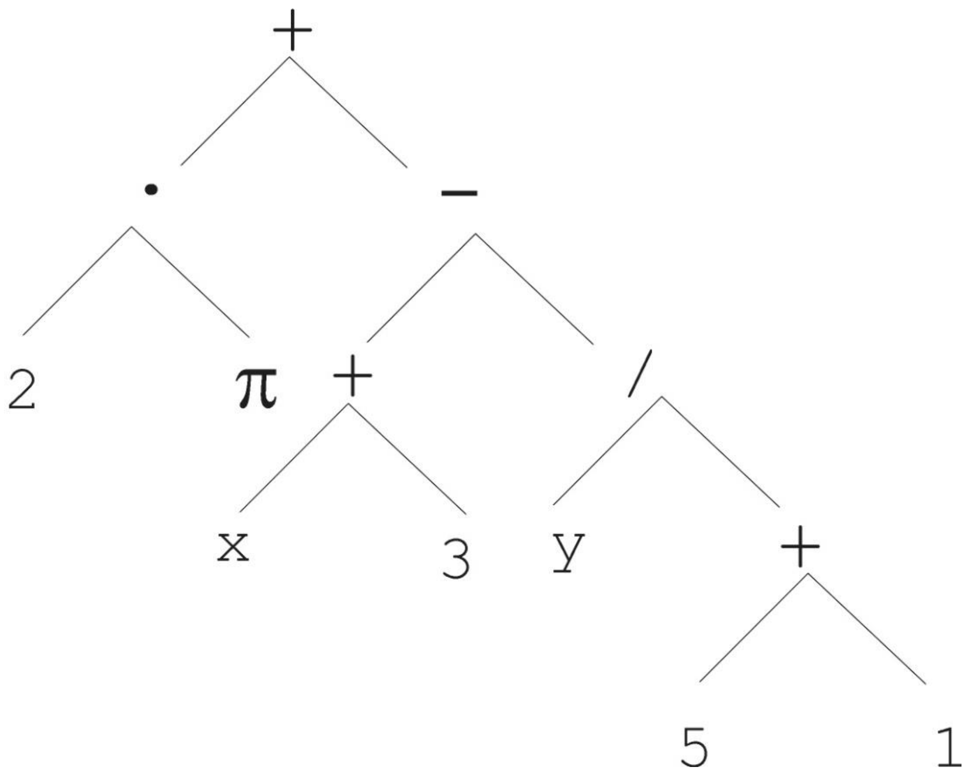


Tree based representation

- Trees are a universal form, e.g. consider
- Arithmetic formula $2 \cdot \pi + \left((x+3) - \frac{y}{5+1} \right)$
- Logical formula $(x \wedge \text{true}) \rightarrow ((x \vee y) \vee (z \leftrightarrow (x \wedge y)))$
- Program

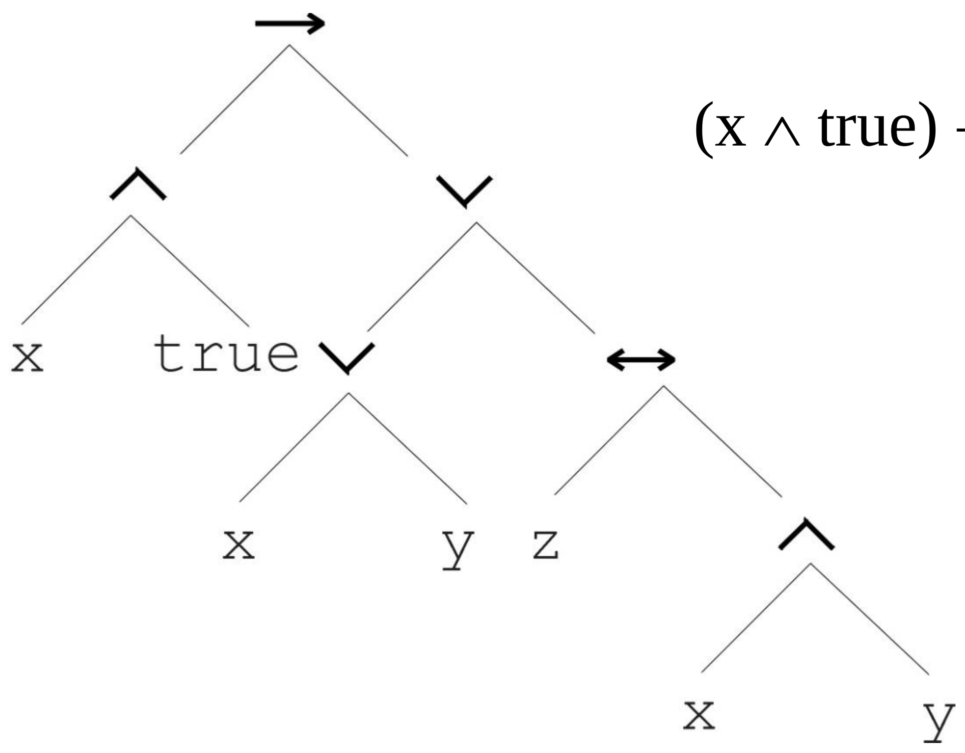
```
i =1;
while (i < 20)
{
        i = i +1
}
```

Tree based representation



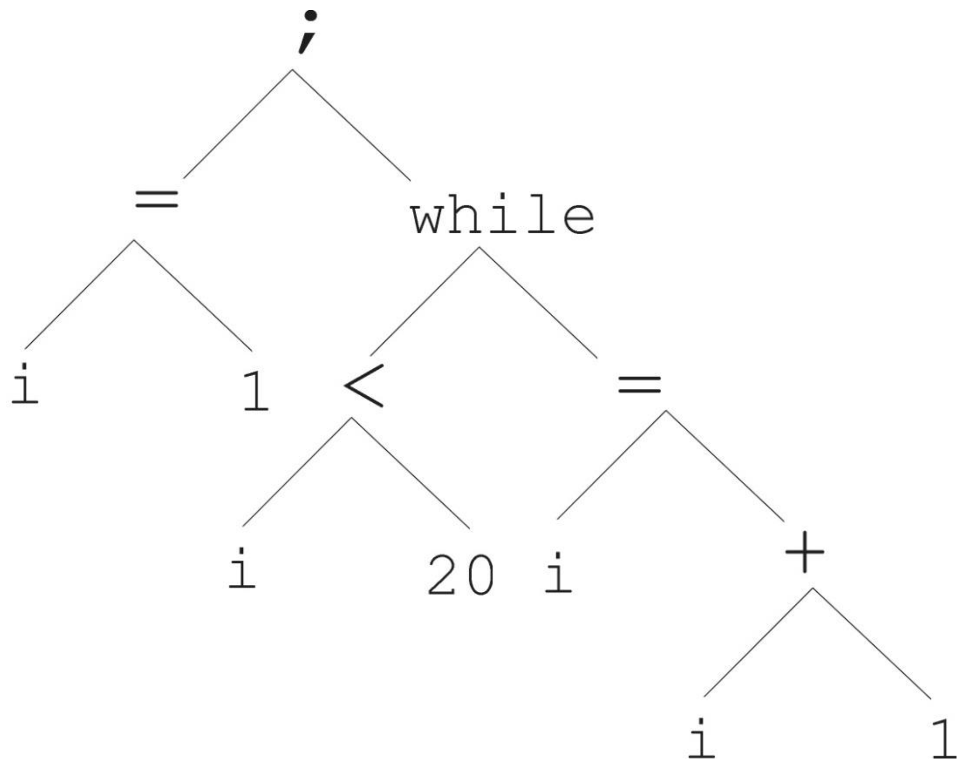
$$2\cdot\pi+\left((x+3)-\frac{y}{5+1}\right)$$

Tree based representation



$(x \wedge true) \rightarrow ((x \vee y) \vee (z \leftrightarrow (x \wedge y)))$

Tree based representation



```
i =1;
while (i < 20)
{
        i = i +1
}
```

Tree based representation

- In GA, ES, EP chromosomes are linear structures (bit strings, integer string, real-valued vectors, permutations)
- Tree shaped chromosomes are non-linear structures
- In GA, ES, EP the size of the chromosomes is fixed
- Trees in GP may vary in depth and width

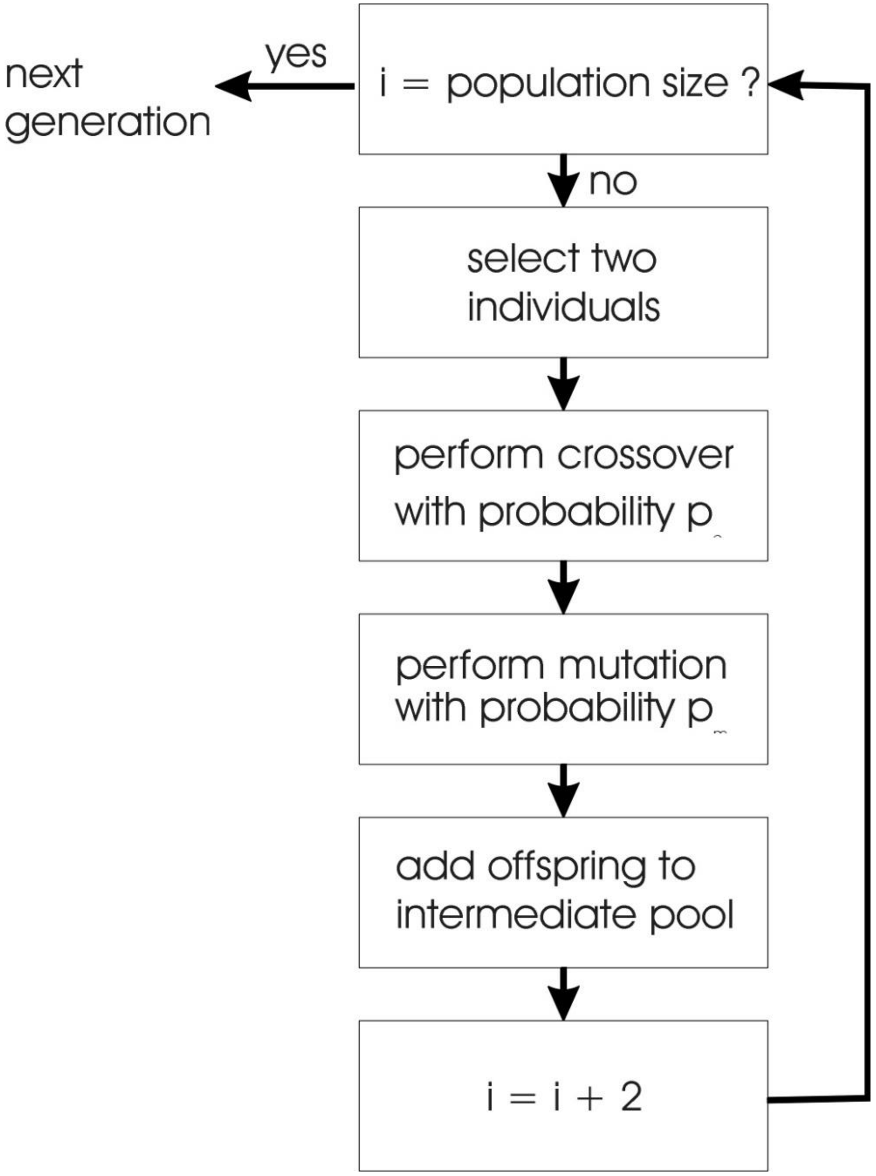
Tree based representation

- Symbolic expressions can be defined by
  - Terminal set T
  - Function set F (with the arities of function symbols)
- Adopting the following general recursive definition:
  1. Every $t \in T$ is a correct expression
  2. $f(e_1, \ldots, e_n)$ is a correct expression if $f \in F$, arity(f)=n and $e_1, \ldots, e_n$ are correct expressions
  3. There are no other forms of correct expressions
- In general, expressions in GP are not typed (closure property: any $f \in F$ can take any $g \in F$ as argument)

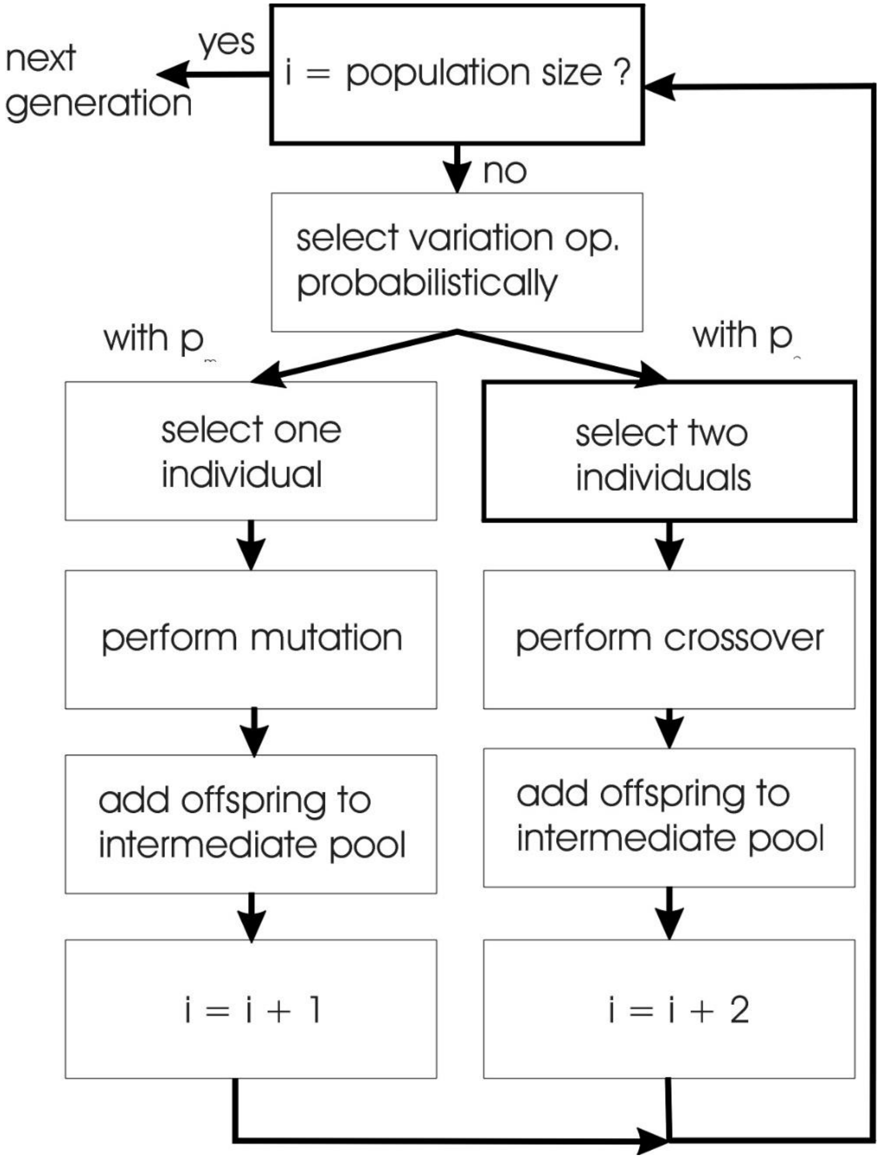
Offspring creation scheme

Compare

- GA scheme using crossover AND mutation sequentially (be it probabilistically)
- GP scheme using crossover OR mutation (chosen probabilistically)


i = population size ?
yes
next generation
no
select two individuals
perform crossover with probability p
perform mutation with probability p
add offspring to intermediate pool
i = i + 2

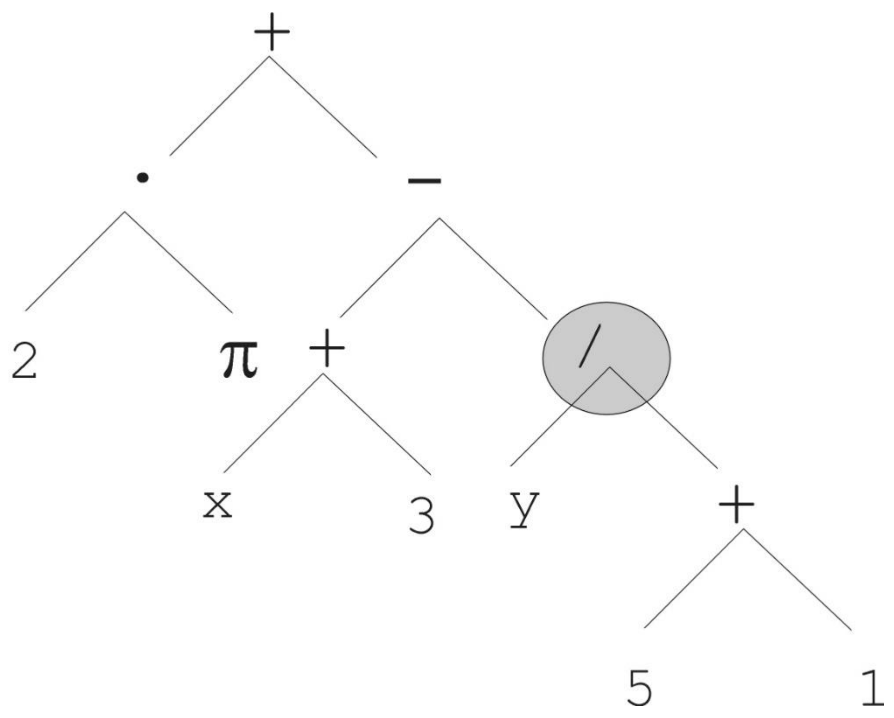
GA flowchart

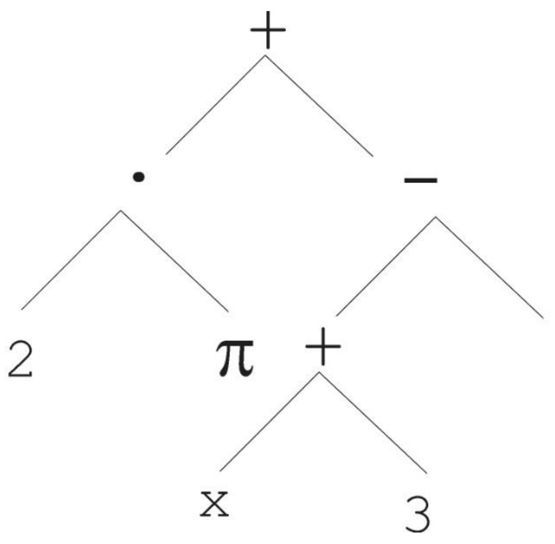


GP flowchart

Mutation

- Most common mutation: replace randomly chosen subtree by randomly generated tree



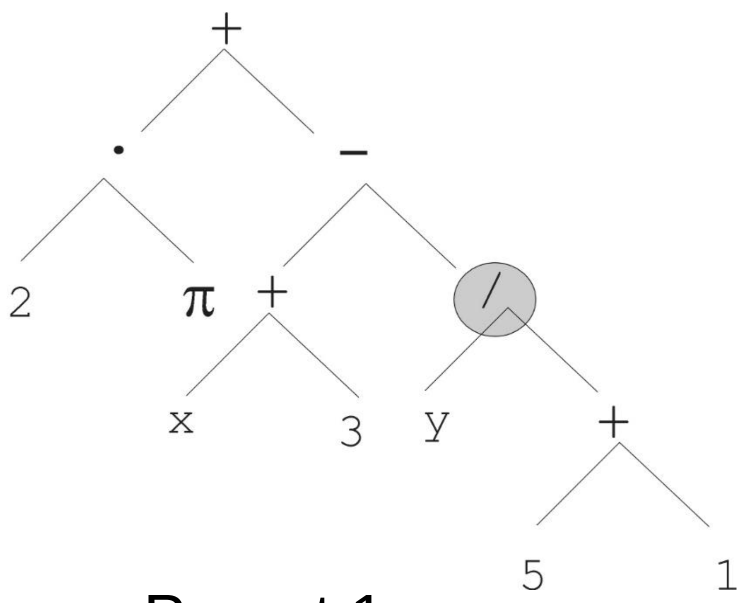


Mutation cont’d

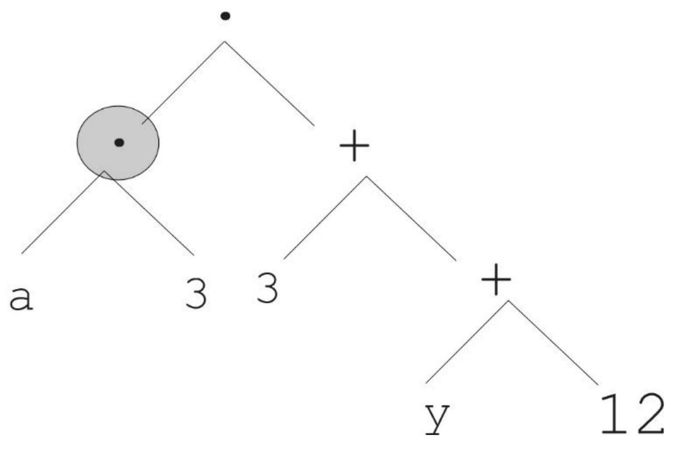
- Mutation has two parameters:
  - Probability p_m to choose mutation vs. recombination
  - Probability to chose an internal point as the root of the subtree to be replaced
- Remarkably p_m is advised to be 0 (Koza’92) or very small, like 0.05 (Banzhaf et al. ’98)
- The size of the child can exceed the size of the parent

Recombination

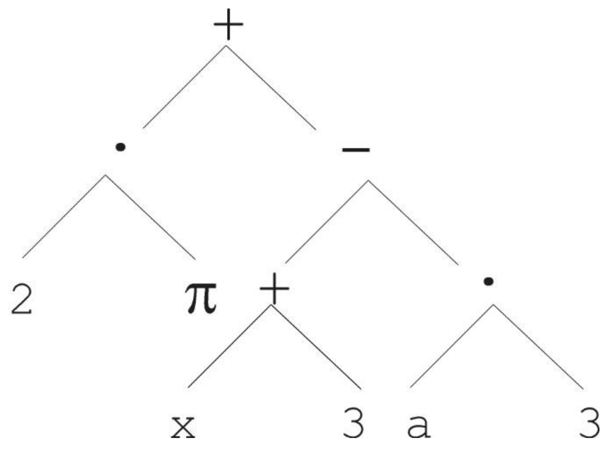
- Most common recombination: exchange two randomly chosen subtrees among the parents
- Recombination has two parameters:
  - Probability p_c to choose recombination vs. mutation
  - Probability to chose an internal point within each parent as crossover point
- The size of offspring can exceed that of the parents



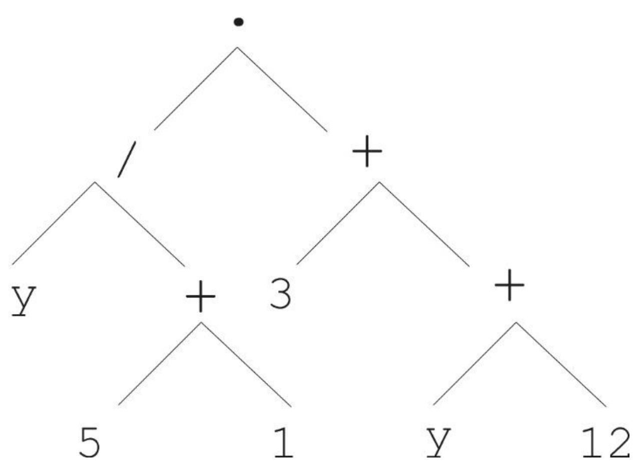
Parent 1



Parent 2



Child 1



Child 2

Selection

- Parent selection typically fitness proportionate
- Over-selection in very large populations
  - rank population by fitness and divide it into two groups:
  - group 1: best x% of population, group 2 other (100-x)%
  - 80% of selection operations chooses from group 1, 20% from group 2
  - for pop. size = 1000, 2000, 4000, 8000 x = 32%, 16%, 8%, 4%
  - motivation: to increase efficiency, %'s come from rule of thumb
- Survivor selection:
  - Typical: generational scheme (thus none)
  - Recently steady-state is becoming popular for its elitism

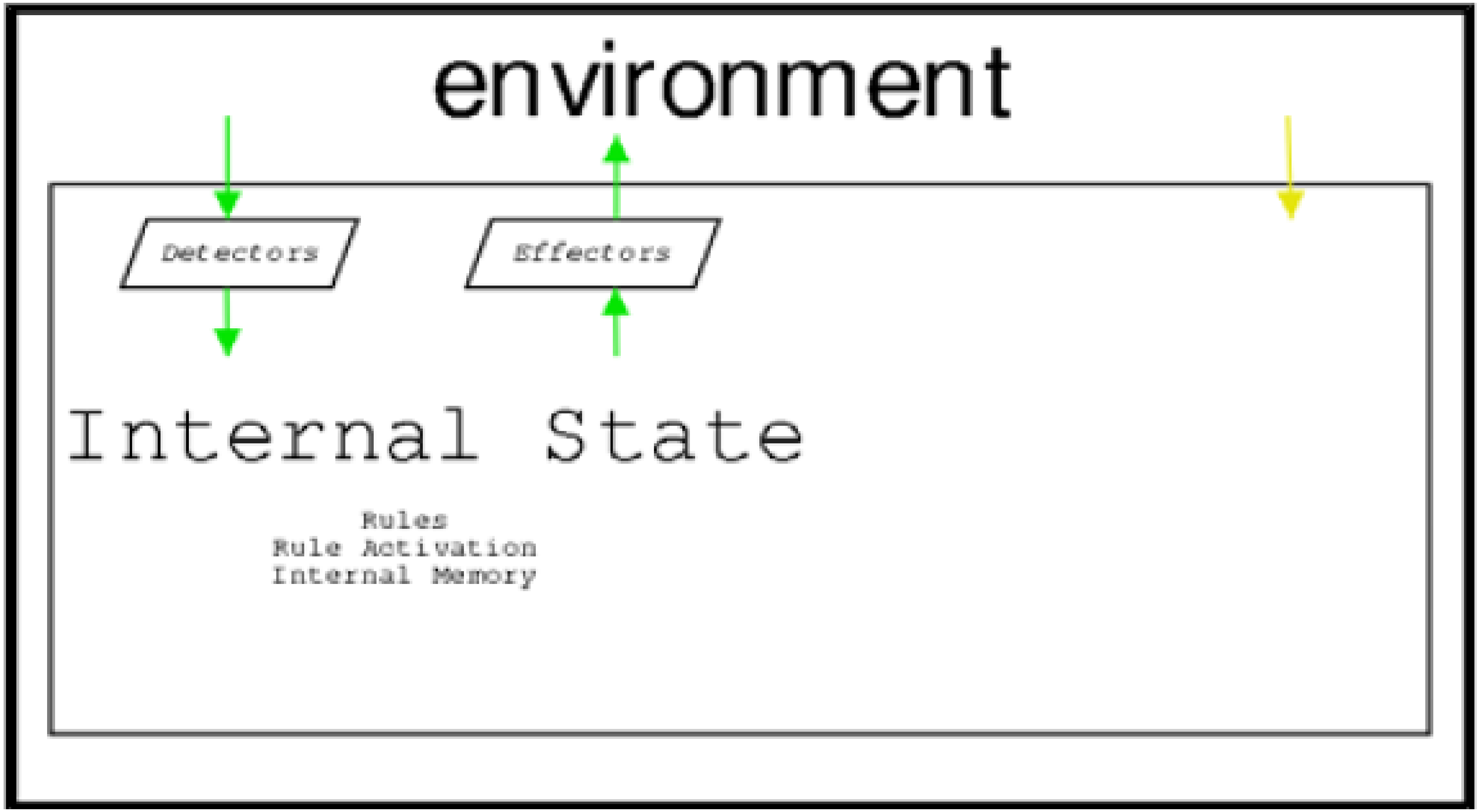
Initialisation

- Maximum initial depth of trees D_{max} is set
- Full method (each branch has depth = D_{max}):
  - nodes at depth $d < D_{max}$ randomly chosen from function set F
  - nodes at depth $d = D_{max}$ randomly chosen from terminal set T
- Grow method (each branch has depth $\leq D_{max}$):
  - nodes at depth $d < D_{max}$ randomly chosen from $F \cup T$
  - nodes at depth $d = D_{max}$ randomly chosen from T
- Common GP initialisation: ramped half-and-half, where grow & full method each deliver half of initial population

Bloat

- Bloat = “survival of the fattest”, i.e., the tree sizes in the population are increasing over time
- Ongoing research and debate about the reasons
- Needs countermeasures, e.g.
  - Prohibiting variation operators that would deliver “too big” children
  - Parsimony pressure: penalty for being oversized

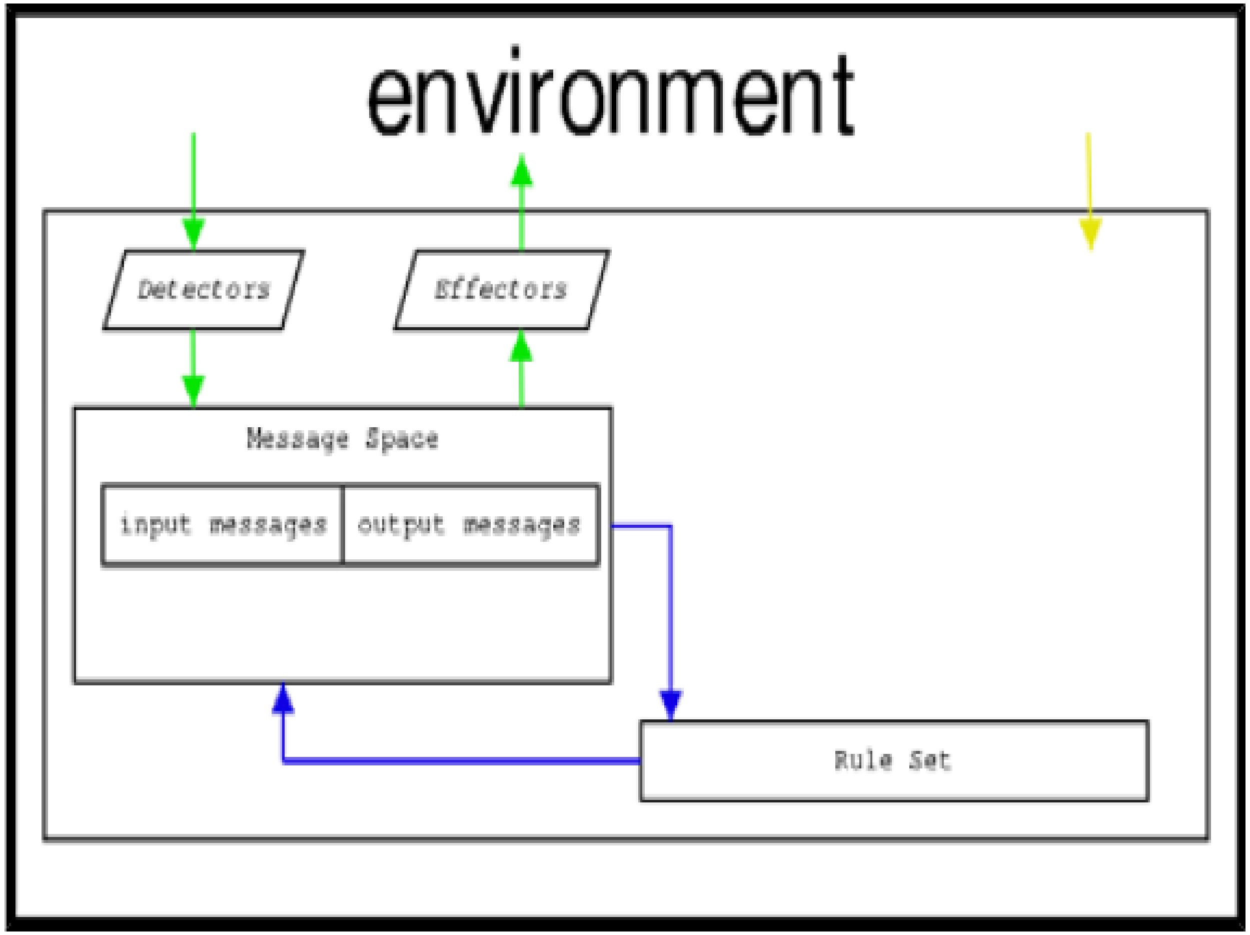
Classifier Systems



CS are Rule Based

Basic Cycle:

- Detectors post messages
- Rules are matched
- Messages are wiped
- Matching rules post messages
- Actuators act on messages



CS Rules

CS Rules

- Production Rules
- Left-hand side and Right-hand side
- if <condition> then <action>
- Computationally Complete
- Convenient

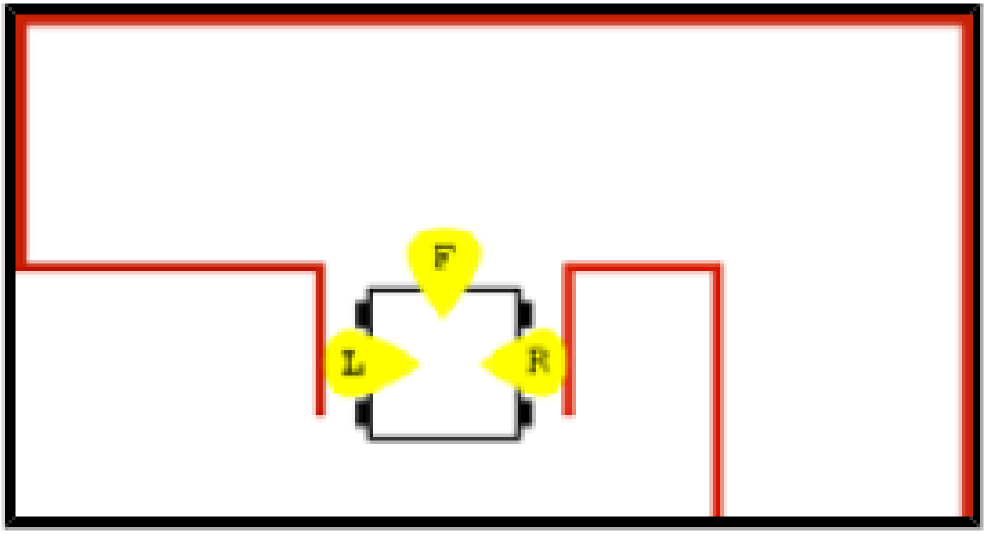
Written as Strings:

- Messages: binary strings
- Right-hand sides: binary strings
- Left-hand sides: ternary strings

Don't carein left-hand side

- Partial message matching

CS Rules - Example



Rule	Sensors			Actuators		
	wall left	wall front	wall right	turn left	go forward	turn righ
1	#	0	#	0	1	0
2	0	1	#	1	0	0
3	1	1	#	0	0	1

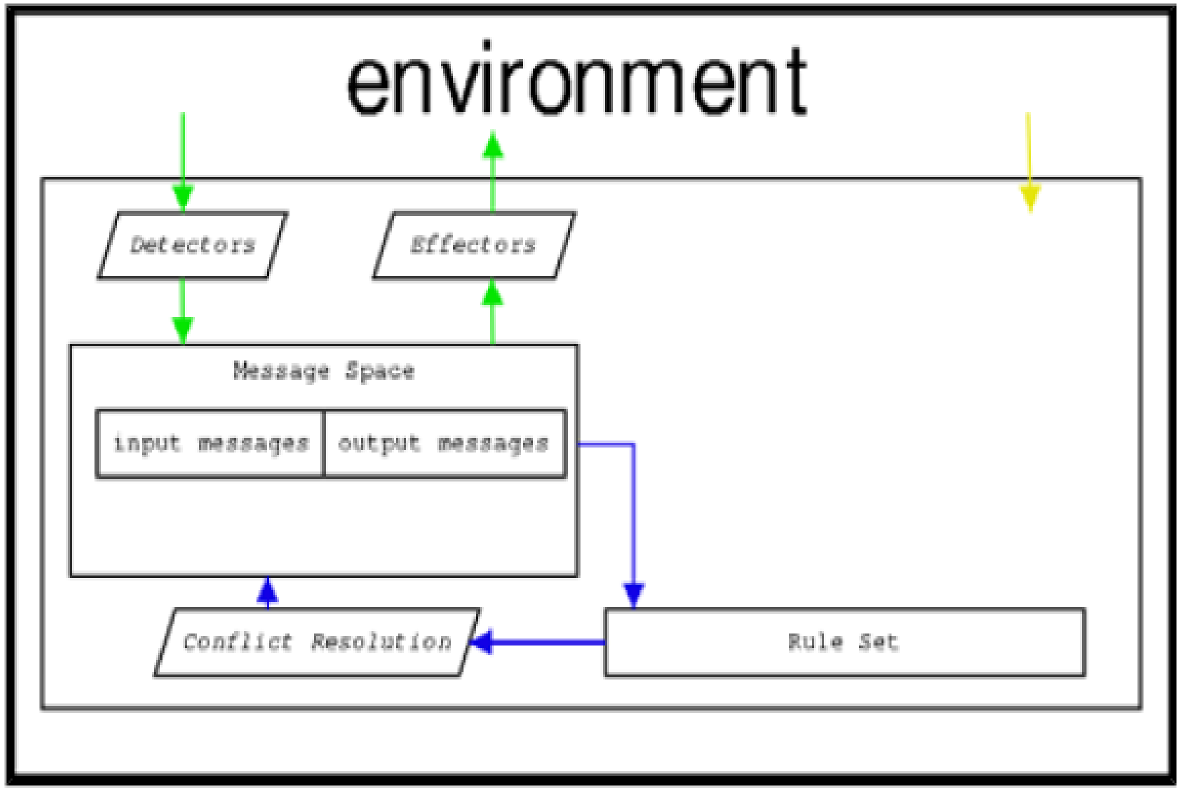
no confliciting rules...!

Rule Conflict

Rule	Sensors			Actuators		
	wall left	wall front	wall right	turn left	go forward	turn right
1	#	#	#	0	1	0
2	#	1	#	1	0	0
3	1	1	#	0	0	1

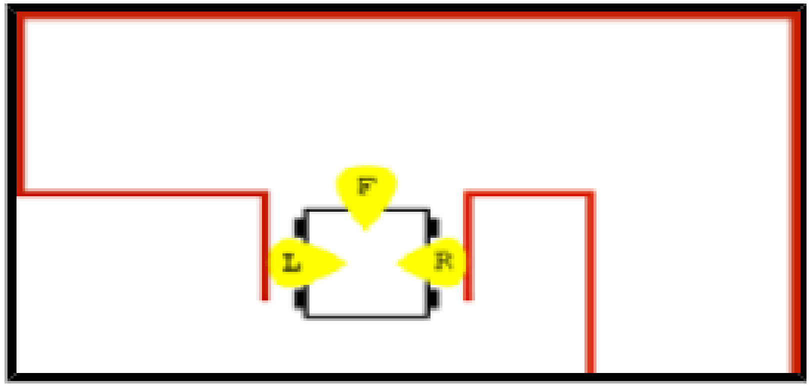
Conflicting Rules

- More than one rule matches, and
- different rules produce conflicting output
- Or: too many rules match, and
- message board overflows



Conflict Resolution Mechanism required

Conflicting Rules



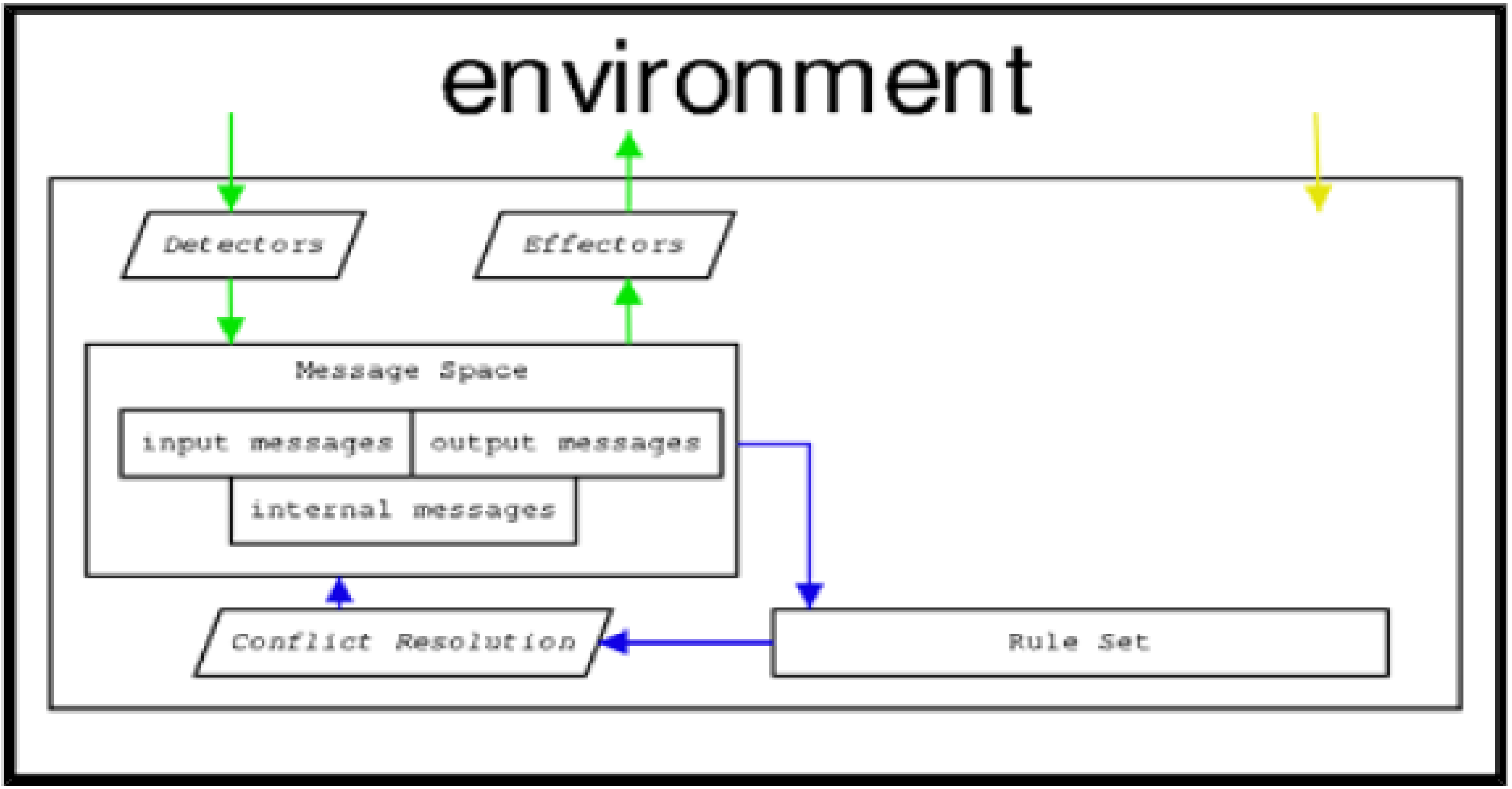
Solution: Strength Value

- Strongest rules win
- Noisy auction

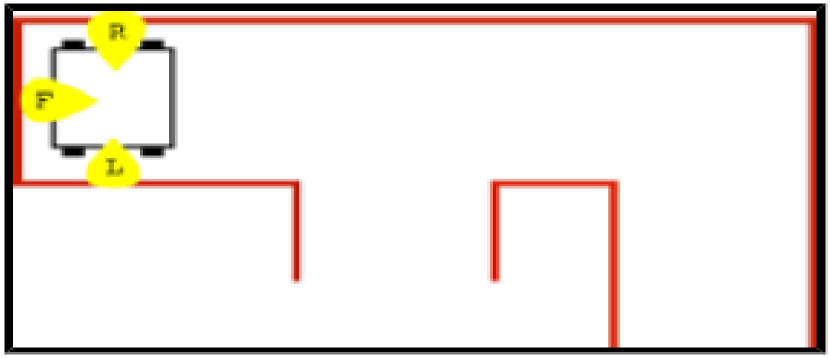
Rule		Sensors			Actuators		
Number	Strength	wall left	wall front	wall right	turn left	go forward	turn right
1	10	#	#	#	0	1	0
2	10	#	1	#	1	0	0
3	10	1	1	#	0	0	1

Question: in the example above, what values for strengths give the correct behaviour?

Internal Messages



Internal Message Example



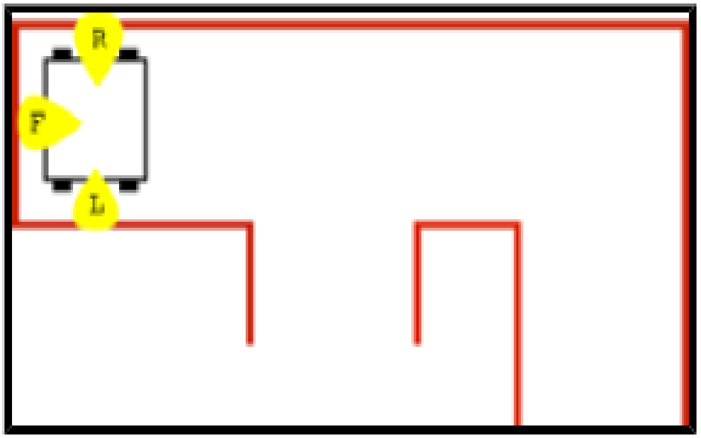
Assumption: robot cannot turn in dead end

- Allow reverse…

Rule		Sensors			Actuators			
Number	Strength	wall left	wall front	wall right	turn left	go forward	turn right	go back
1	10	1	1	1	0	0	0	1

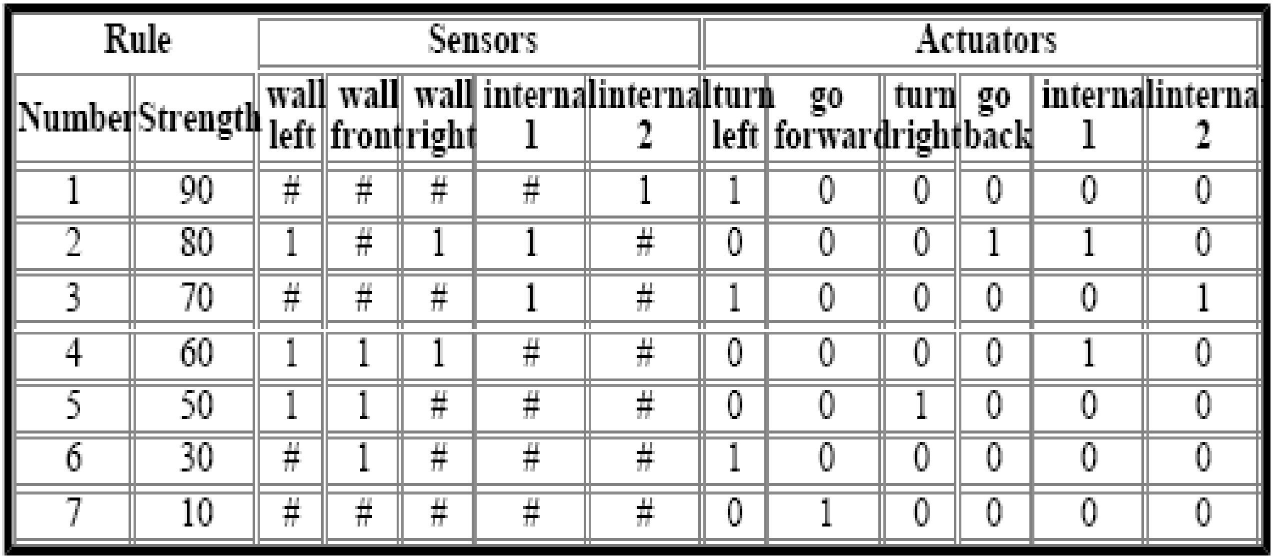
Question: what is the problem with this solution?

Internal Message Example (cont)



Internal State Memory

- LHS match environmental and/or internal messges
- RHS set environmental and/or internal messages
- Allows complex rule chains
- Allows complex action sequences



Rule		Sensors					Actuators					
Number	Strength	wall left	wall front	wall right	internal 1	internal 2	turn left	go forward	turn right	go back	internal 1	internal 2
1	90	#	#	#	#	1	1	0	0	0	0	0
2	80	1	#	1	1	#	0	0	0	1	1	0
3	70	#	#	#	1	#	1	0	0	0	0	1
4	60	1	1	1	#	#	0	0	0	0	1	0
5	50	1	1	#	#	#	0	0	1	0	0	0
6	30	#	1	#	#	#	1	0	0	0	0	0
7	10	#	#	#	#	#	0	1	0	0	0	0