# Introduction to Schema Theory

*A survey lecture of pessimistic & exact schema theory*

William C. Liles
R. Paul Wiegand

wliles@cs.gmu.edu

paul@tesseract.org

ECLab
George Mason University

# Outline of Discussion

Part I: Overview of Schema Theory ←

Part II: Pessimistic Schema Theory

Part III: Exact Schema Theory

Part IV: Conclusions

# What is a Theory?

- Set of analytical tools to help answer questions
- Particular domain in which to ask questions

# What is a Theory?

- Set of analytical tools to help answer questions
- Particular domain in which to ask questions

**What do theories *do*?**

- Predict
- Explain

# What are Schemata?

- Can view Schemata in many ways
  - Templates specifying groups (sets) of ”similar” chromosomes
  - Partitions of genome space
  - Descriptions of hyperplanes through genome space
  - Sets of search points sharing some ”syntactic feature”

# What are Schemata?

- Can view Schemata in many ways
  - Templates specifying groups (sets) of "similar" chromosomes
  - Partitions of genome space
  - Descriptions of hyperplanes through genome space
  - Sets of search points sharing some "syntactic feature"
- Example for binary representation:

"*" $\Leftrightarrow$ "Don't care"

| Schema | Members |
|---|---|
| 1**0 | 1000 |
| | 1010 |
| | 1100 |
| | 1110 |

# What is Schema Theory?

- Describes how schemata are *expected* to propagate from one generation to the next
- More specifically:
  - Divides the space into subspaces
  - Quantifies these subspaces
  - Explains how & why individuals move between subspaces

# What are Schema Theorems?

- *Specific* analytical models
- Usually reflect particular representational choices
- May provide only a lower bound schemata growth
- May provide tight bounds on schemata growth

# The Traditional Schema Theorem

- Pessimistic prediction of schema growth in a GA
- *NOT* the same as the Building Block Hypothesis

# The Traditional Schema Theorem

- Pessimistic prediction of schema growth in a GA
- *NOT* the same as the Building Block Hypothesis

**From Michalewicz, 1992:**

> SCHEMA THEOREM: *Short, low-order, above-average schemata receive exponentially increasing trials in subsequent generations of a genetic algorithm*

# The Traditional Schema Theorem

- Pessimistic prediction of schema growth in a GA
- *NOT* the same as the Building Block Hypothesis

**From Michalewicz, 1992:**

> SCHEMA THEOREM: *Short, low-order, above-average schemata receive exponentially increasing trials in subsequent generations of a genetic algorithm*

**Prediction**

# The Traditional Schema Theorem

- Pessimistic prediction of schema growth in a GA
- *NOT* the same as the Building Block Hypothesis

**From Michalewicz, 1992:**

SCHEMA THEOREM: *Short, low-order, above-average schemata receive exponentially increasing trials in subsequent generations of a genetic algorithm*

**Prediction**

BUILDING BLOCK HYPOTHESIS: *A GA seeks near optimal performance through the juxtaposition of short, low-order, high-performance schemata, called the building blocks.*

# The Traditional Schema Theorem

- Pessimistic prediction of schema growth in a GA
- *NOT* the same as the Building Block Hypothesis

**From Michalewicz, 1992:**

SCHEMA THEOREM: *Short, low-order, above-average schemata receive exponentially increasing trials in subsequent generations of a genetic algorithm*

**Prediction**

BUILDING BLOCK HYPOTHESIS: *A GA seeks near optimal performance through the juxtaposition of short, low-order, high-performance schemata, called the building blocks.*

**Explanation**

# Advantages of Schema Theory

- Appeals to our intuition of how problems may be solved (i.e., building blocks)
- Can be formulated in a concise way, such that specific instantiations for particular EAs can be ”plugged in” to main theorems
- With exact models, we can get correct predictions of expectation, as well as bounds on the probability of those expectations

# Criticism of the Schema Theory

- Results & conclusions of limited use. Why?

# Criticism of the Schema Theory

- Results & conclusions of limited use. Why?
  - Traditionally produces a lower-bound expectation for schemata growth
  - Not easy to predict global behavior without a recursive solution
  - Operates under the assumption that knowing what schemata are doing helps us understand how & why the GA is working (it doesn't always)

# Criticism of the Schema Theory

- Results & conclusions of limited use. Why?
  - Traditionally produces a lower-bound expectation for schemata growth
  - Not easy to predict global behavior without a recursive solution
  - Operates under the assumption that knowing what schemata are doing helps us understand how & why the GA is working (it doesn't always)
- Applicability is low. Why?

# Criticism of the Schema Theory

- Results & conclusions of limited use. Why?
  - Traditionally produces a lower-bound expectation for schemata growth
  - Not easy to predict global behavior without a recursive solution
  - Operates under the assumption that knowing what schemata are doing helps us understand how & why the GA is working (it doesn't always)
- Applicability is low. Why?
  - Large effort to produce very specific models
  - Most EAs modeled under ST are simple EAs, not used in practice

# Outline of Discussion

Part I: Overview of Schema Theory $\surd$

Part II: Pessimistic Schema Theory $\leftarrow$

Part III: Exact Schema Theory

Part IV: Conclusions

# Traditional Schema Theory

- Attempts to give insight
  - About *how* GAs work
  - Describes how the expected number of schemata will *at least* grow in the next generation
- Generally assumes
  - Binary, fixed-length representation
  - Bit flip mutation, 1-point crossover
  - Proportional selection

# Traditional Schema Theory

- Attempts to give insight
  - About *how* GAs work
  - Describes how the expected number of schemata will *at least* grow in the next generation
- Generally assumes
  - Binary, fixed-length representation
  - Bit flip mutation, 1-point crossover
  - Proportional selection

$$E[\,\#\text{ strs} \in \text{schema @ gen } t+1] \geq (\#\text{ strs} \in \text{schema @ } t)\cdot$$
$$(\text{rel. value of schema})\cdot$$
$$(\text{prob } \sim\text{destroyed})$$

# Notation

- Individual are fixed-length binary strings, $v \in \{0,1\}^l$
- Schema are fixed length ternary strings, $s \in \{0,1,*\}^l$
- Schema define a set of individual strings. $v \in s$ iff $\forall i \in \{0 \ldots l\}, v_i = s_i \vee s_i = *$
- Population at time $t$ is a set of binary strings, $P_t = \{v^{(1)}, v^{(2)}, \ldots, v^{(n)}\}$, $v^{(i)} \in \{0,1\}^l$
- Match is no. of strings in $P_t$ also in schema $s$. $m(s,t) = \|\{v \in P_t, v \in s\}\|$

# Notation (continued)

- Mean fitness of strings in schema $s$ also in $P_t$, $f(s,t)$
- Mean fitness of the population, $\overline{f}(t)$
- Probability of mutation, $p_m$
- Probability of crossover, $p_c$

# Notation (continued)

- Mean fitness of strings in schema $s$ also in $P_t$, $f(s,t)$
- Mean fitness of the population, $\bar{f}(t)$
- Probability of mutation, $p_m$
- Probability of crossover, $p_c$

$$E[m(s,t+1)] \geq m(s,t)\frac{f(s,t)}{f(t)}Pr[\text{surv mut}]Pr[\text{surv xover}]$$

# Measures of Schema

- **Order** of schema is the # of fixed positions, $o(s) = \|\{\forall i \in \{1 \ldots l\}, s_i \neq *\}\|$
- **Defining length** is the longest distance between two fixed positions,

$$\delta(s) = \max\left(|i - j|,\ \forall i, j \in \{1 \ldots l\}, s_i \neq * \wedge s_j \neq *\right)$$

# Measures of Schema

- **Order** of schema is the # of fixed positions, $o(s) = \|\{\forall i \in \{1 \ldots l\}, s_i \neq *\}\|$
- **Defining length** is the longest distance between two fixed positions,

$$\delta(s) = \max\left(|i - j|, \forall i, j \in \{1 \ldots l\}, s_i \neq * \wedge s_j \neq *\right)$$

**Example:**

$s =$ `* 0 * 1 1 * * 1 *`

$o(s) = 4$

$\delta(s) = 6$

# Surviving Mutation

- Probability no mutation at a given position occurs is $1 - p_m$
- All mutations are independent
- We only care about the fixed positions
- So probability schema survives disruption is $(1 - p_m)^{o(s)}$

# Surviving Crossover

- A crossover event which does not divide the defined positions in the schema is harmless
- The probability the crossover breaks the schema is $\frac{\delta(s)}{l-1}$
- Thus the probability to survive crossover is $1 - p_c \frac{\delta(s)}{l-1}$

# The Traditional Schema Theorem

$$E[m(s,t+1)] \geq m(s,t)\frac{f(s,t)}{f(t)}(1-p_m)^{o(s)}\left(1-p_c\frac{\delta(s)}{l-1}\right)$$

# The Traditional Schema Theorem

$$E[m(s,t+1)] \geq m(s,t)\frac{f(s,t)}{f(t)}(1-p_m)^{o(s)}\left(1-p_c\frac{\delta(s)}{l-1}\right)$$

But we can do better, since this ignores the possibility that you select a breeding partner from the same schema...

$$m(s,t)\frac{f(s,t)}{f(t)}(1-p_m)^{o(s)}\left(\left(1-p_c\frac{\delta(s)}{l-1}\right)\left(1-\frac{m(s,t)f(s,t)}{n\cdot f(t)}\right)\right)$$

# The Traditional Schema Theorem

$$E[m(s,t+1)] \geq m(s,t)\frac{f(s,t)}{f(t)}(1-p_m)^{o(s)}\left(1-p_c\frac{\delta(s)}{l-1}\right)$$

But we can do better, since this ignores the possibility that you select a breeding partner from the same schema...

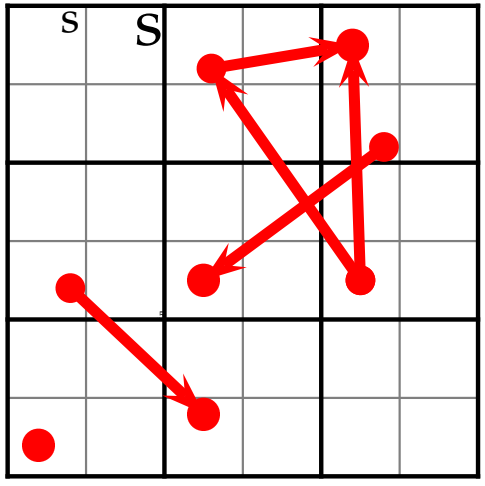
$$m(s,t)\frac{f(s,t)}{f(t)}(1-p_m)^{o(s)}\left(\left(1-p_c\frac{\delta(s)}{l-1}\right)\left(1-\frac{m(s,t)f(s,t)}{n\cdot f(t)}\right)\right)$$

More generally, we can replace the selection method...

$$n\cdot p(s,t)(1-p_m)^{o(s)}\left(\left(1-p_c\frac{\delta(s)}{l-1}\right)(1-p(s,t))\right)$$

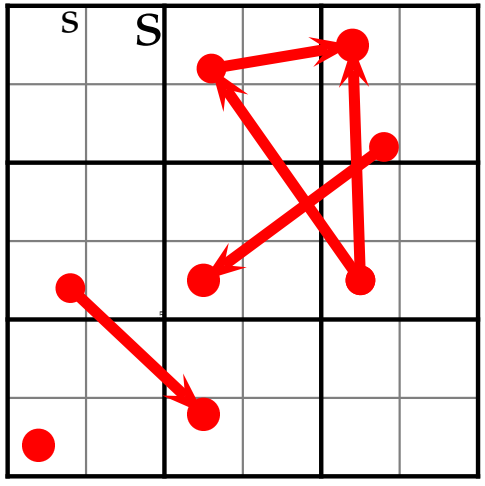
Where $p(s,t)$ is the probability of selecting schema $s$ at generation $t$. For proportional selection $p(s,t)=\frac{m(s,t)}{n}\cdot\frac{f(s,t)}{f(t)}$.

# What does this tell us?




- Helps explain how $m(s,t)$ is affected by GA operators
- Helps predict how $m(s,t)$ varies from one generation to the next
- Provides only a *lower bound*

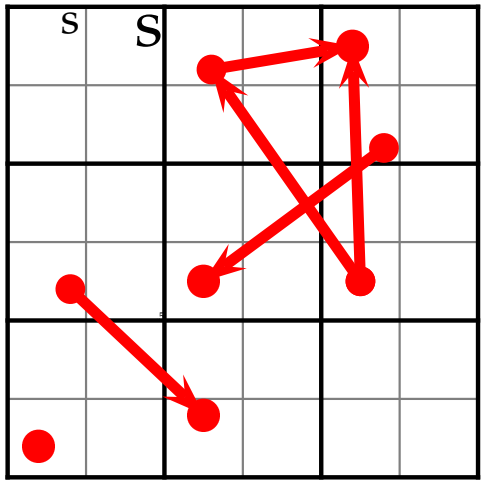
# What does this tell us?




- Helps explain how $m(s,t)$ is affected by GA operators
- Helps predict how $m(s,t)$ varies from one generation to the next
- Provides only a *lower bound*

- Want appropriate schema defn *(granularity)*
- Want appropriate measures *(meaning)*

# What does this tell us?




- Helps explain how $m(s,t)$ is affected by GA operators
- Helps predict how $m(s,t)$ varies from one generation to the next
- Provides only a *lower bound*

- Want appropriate schema defn *(granularity)*
- Want appropriate measures *(meaning)*

  - No. indiv in schema
  - Avg fitness of schema
  - Avg fitness of pop
  - Disruption of ops
  - etc.

# Genetic Programming Schemata

- We can express schemata as a list of ordered pairs

  $0 * 1 * * 1\ 1 \Leftrightarrow [(0,1),(1,3),(11,6)]$

- Supposing $v \in \{a,b,c,d\}^l$,

  $a * b\ c * a\ d \Leftrightarrow [(a,1),(bc,3),(ad,6)]$

- Traditional ST specifies the complete string, so position is important.
- But we could match partial strings...
- And we could match strings independent of position...

# GP Schemata (continued)

- No. of individuals matching $s$ in $P_t$ is $m(s,t)$
- No. of substrings matching $s$ in $P_t$ is $\iota(s,t)$ *(instantiations)*

**Example Population:**

a b a a c

a c c a b

c c c c b

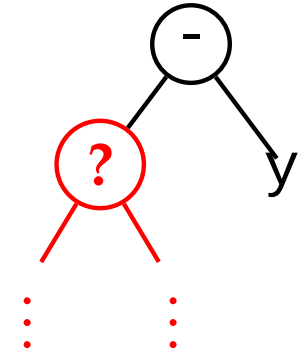
a b c a b

$$m([ab],t) = 3, \quad \iota([ab],t) = 4$$

$$m([a,b],t) = 4, \quad \iota([a,b],t) = 12$$

# Rosca's Rooted Tree Schemata

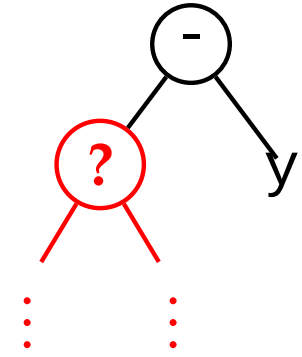
- Syntactically:
  - Every schema is a contiguous tree fragment
  - Every schema includes the root node of the tree
  - Use '#' character rather than '*' for *don't care* symbol
- Semantically:
  - A schema defines a set of programs
  - $s =$ ( - # y ) defines the set of programs:



# Rosca's Rooted Tree Schemata

- Syntactically:
  - Every schema is a contiguous tree fragment
  - Every schema includes the root node of the tree
  - Use '#' character rather than '*' for *don't care* symbol
- Semantically:
  - A schema defines a set of programs
  - $s =$ ( - # y ) defines the set of programs:

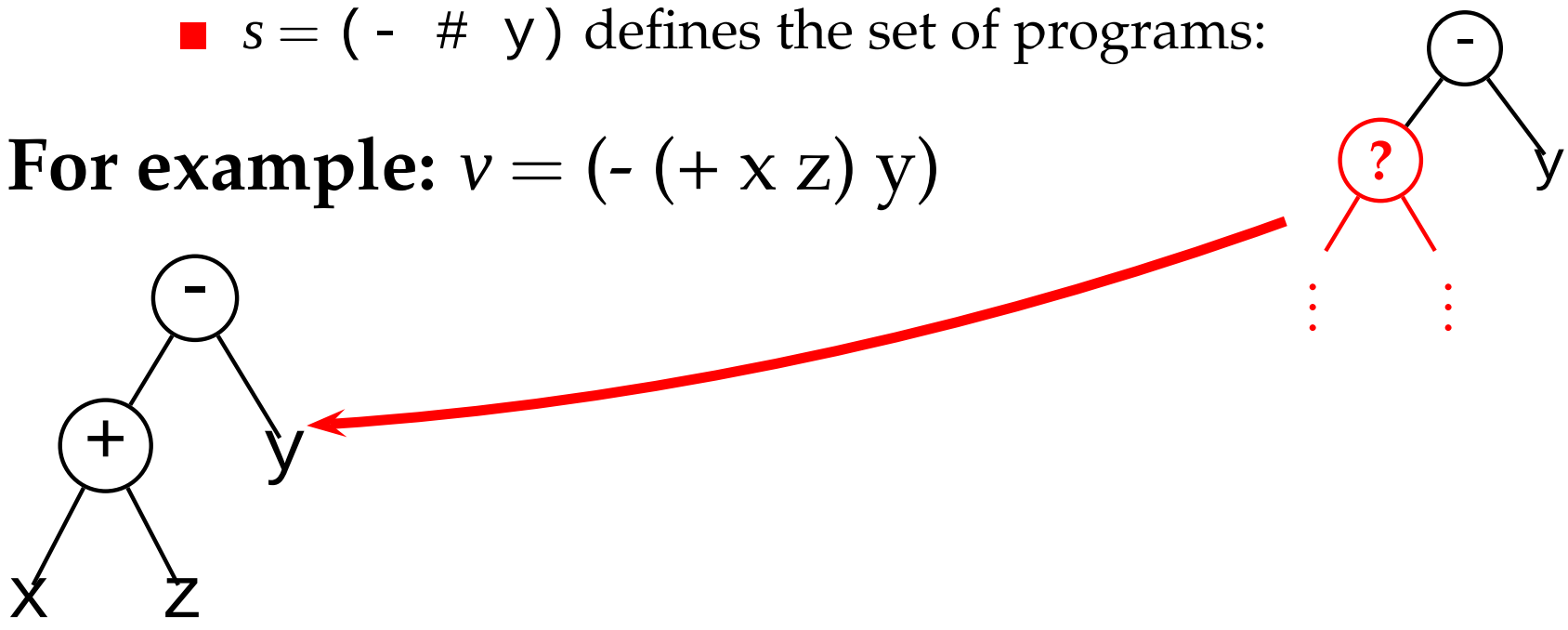
**For example:** $v =$ (- (+ x z) y)



# Rosca's Rooted Tree Schemata

- Syntactically:
  - Every schema is a contiguous tree fragment
  - Every schema includes the root node of the tree
  - Use '#' character rather than '*' for *don't care* symbol
- Semantically:
  - A schema defines a set of programs
  - $s =$ ( - # y ) defines the set of programs:

**For example:** $v =$ (- (+ x z) y)



# Rosca's Schemata Notation/Measures

- Size of program $v$ matching schema $s$ is $N(v)$
- The fitness of program $v$ is $f(v)$
- The order of schema is number of defining symbols it contains, $O(v)$
- The population at time $t$ is the multiset $P(t)$
- All programs in $P(t)$ which are also members of the schema $s$ is denoted $s \cap P(t)$
- The probability (per child) that mutation is applied is $p'_m$

# Rosca's Schema Theorem

$$E[m(s,t+1)] \geq$$
$$m(s,t)\frac{f(s,t)}{f(t)}\left[1-(p'_m+p_c)\sum_{v\in s\cap P(t)}\frac{O(s)}{N(v)}\frac{f(v)}{\sum_{v\in s\cap P(t)}f(v)}\right]$$

- Divide space into subspaces containing programs of different sizes and shapes

# Rosca's Schema Theorem

$$E[m(s,t+1)] \geq$$

$$m(s,t)\frac{f(s,t)}{f(t)}\left[1-(p'_m+p_c)\sum_{v\in s\cap P(t)}\overbrace{\frac{O(s)}{N(v)}}^{\text{Frag. of inst.}}\frac{f(v)}{\sum_{v\in s\cap P(t)}f(v)}\right]$$

- Divide space into subspaces containing programs of different sizes and shapes
- Estimate fragility of schema instances

# Rosca's Schema Theorem

$$E[m(s,t+1)] \geq$$

$$m(s,t)\frac{f(s,t)}{f(t)}\left[1-(p'_m+p_c)\underbrace{\sum_{v\in s\cap P(t)}\overbrace{O(s)}^{\text{Frag. of inst.}}\frac{f(v)}{N(v)}\frac{f(v)}{\sum_{v\in s\cap P(t)}f(v)}}_{\text{Prob disrupting s}}\right]$$

- Divide space into subspaces containing programs of different sizes and shapes
- Estimate fragility of schema instances
- Use wt'd sum & prob of seln. to est. prob. of disrupting schema

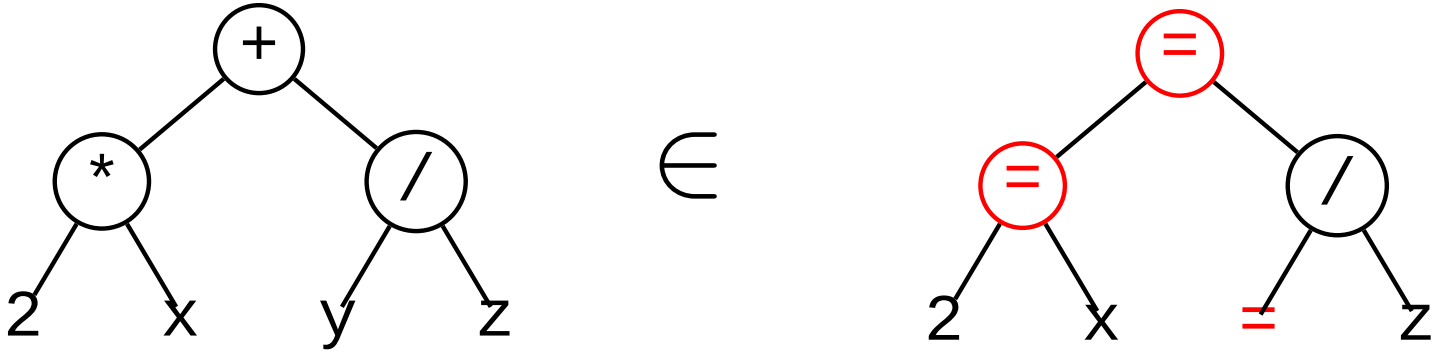
# Poli-Langdon Schemata

- Fixed size and shape
- Set of functions is denoted, $\mathcal{F}$
- Set of terminals is denoted, $\mathcal{T}$
- The "=" symbol is "don't care" for a *single* function or terminal
- Schema $s$ is a rooted tree composed of nodes from $\mathcal{F} \cup \mathcal{T} \cup \{=\}$

# Poli-Langdon Schemata

- Fixed size and shape
- Set of functions is denoted, $\mathcal{F}$
- Set of terminals is denoted, $\mathcal{T}$
- The "=" symbol is "don't care" for a *single* function or terminal
- Schema $s$ is a rooted tree composed of nodes from $\mathcal{F} \cup \mathcal{T} \cup \{=\}$

**For example:** $\mathcal{F} \in \{+,-,*,/\}, \mathcal{T} \in \{1,\ldots,9,x,y,z\}$



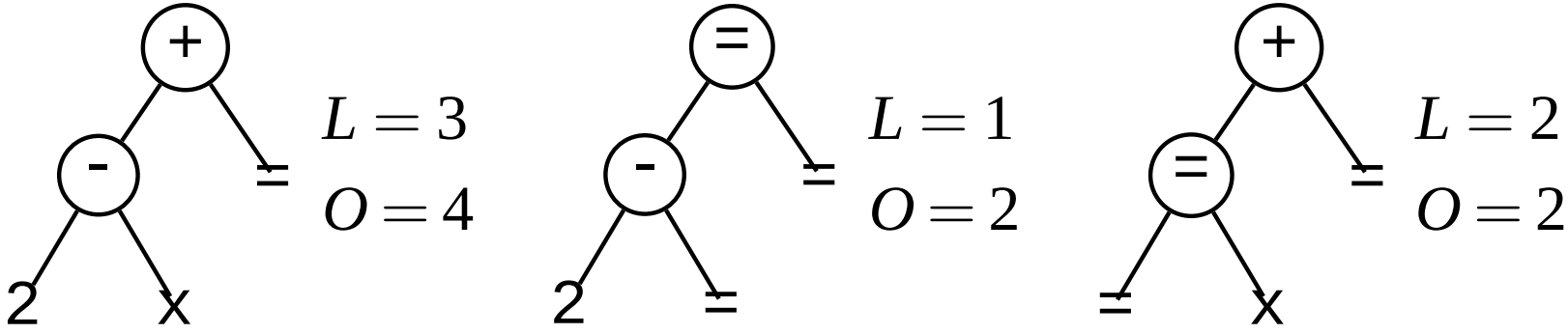
# Poli-Langdon ST Notation/Measures

- The order is the number of non-= symbols of the schema, $O(s)$
- The length is the total number of nodes in the schema, $N(s)$
- The defining length is the number of links in the minimum tree fragment including all non-= symbols in a schema, $L(s)$

# Poli-Langdon ST Notation/Measures

- The order is the number of non-= symbols of the schema, $O(s)$
- The length is the total number of nodes in the schema, $N(s)$
- The defining length is the number of links in the minimum tree fragment including all non-= symbols in a schema, $L(s)$

**For example:**



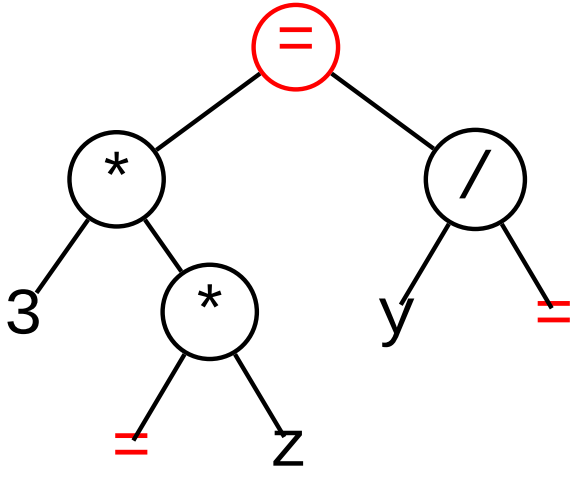
# Poli-Langdon ST Notation/Measures

- The zeroth order schema with the same structure as schema $s$ is G(s)
- The conditional prob. that $s$ is disrupted by x-over from parent not in $G(s)$ is $p_{diff}$
- The prob. of selecting the $s$ from population at time $t$ is $p(s,t)$

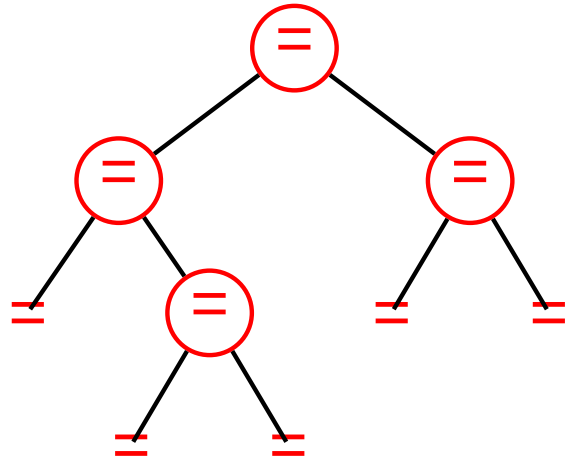
# Poli-Langdon ST Notation/Measures

- The zeroth order schema with the same structure as schema $s$ is G(s)
- The conditional prob. that $s$ is disrupted by x-over from parent not in $G(s)$ is $p_{diff}$
- The prob. of selecting the $s$ from population at time $t$ is $p(s,t)$

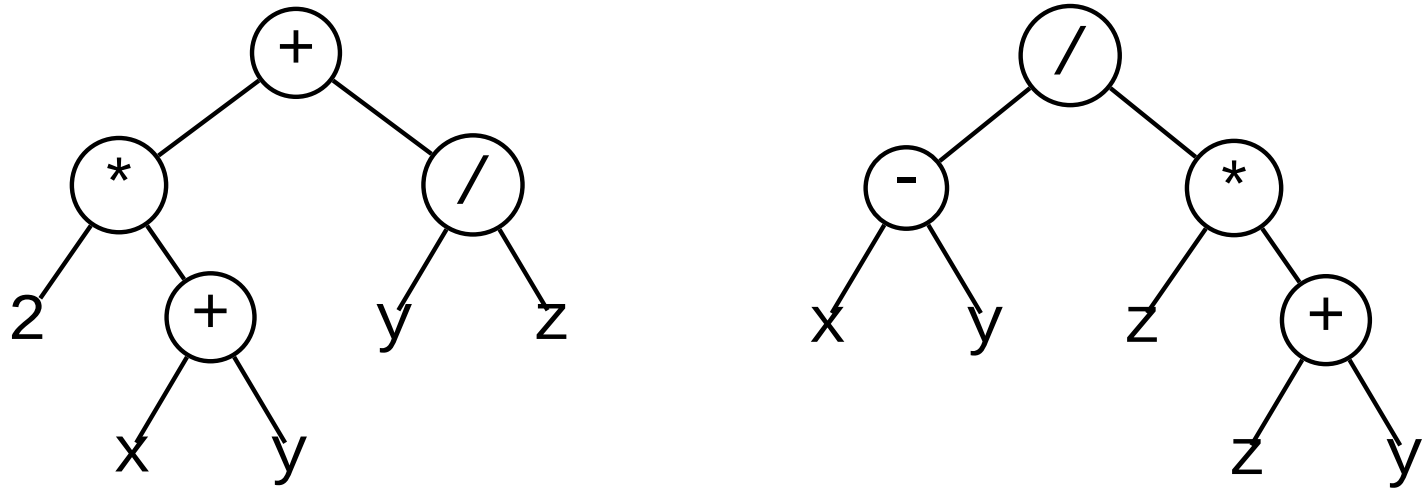
**Schema $s$:**




**Schema $G(s)$:**

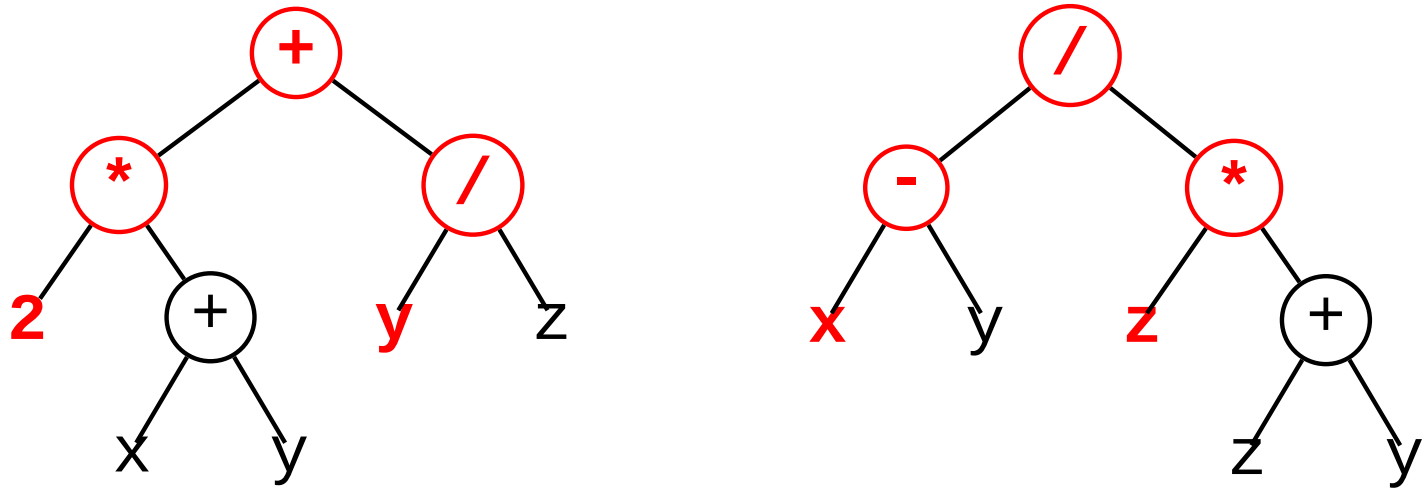


# GP One-Point Crossover



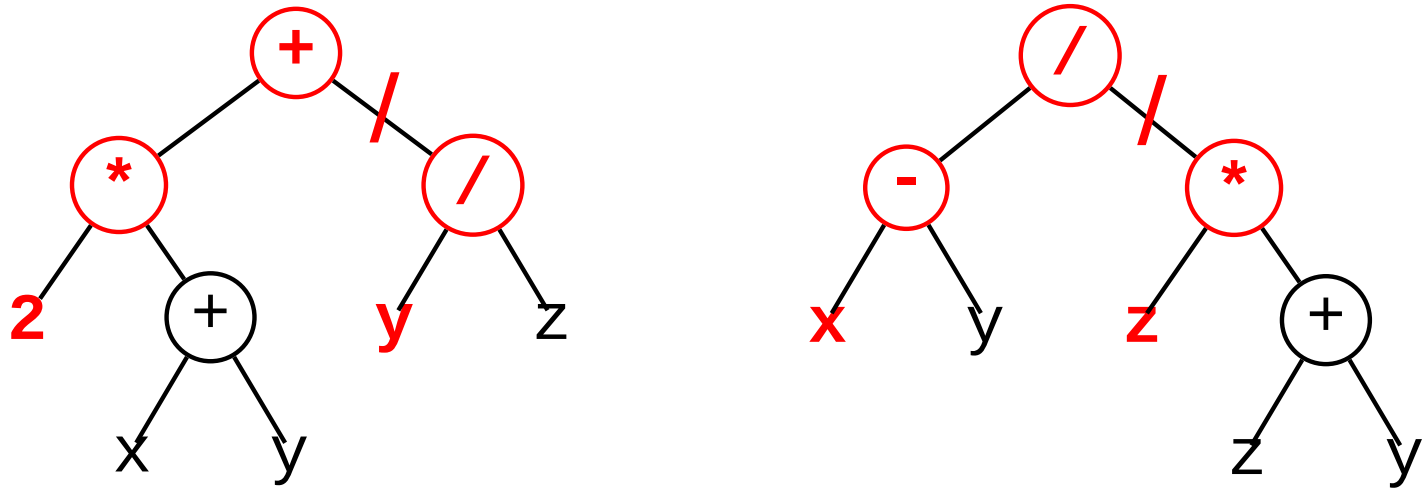

- Align the two parents

# GP One-Point Crossover



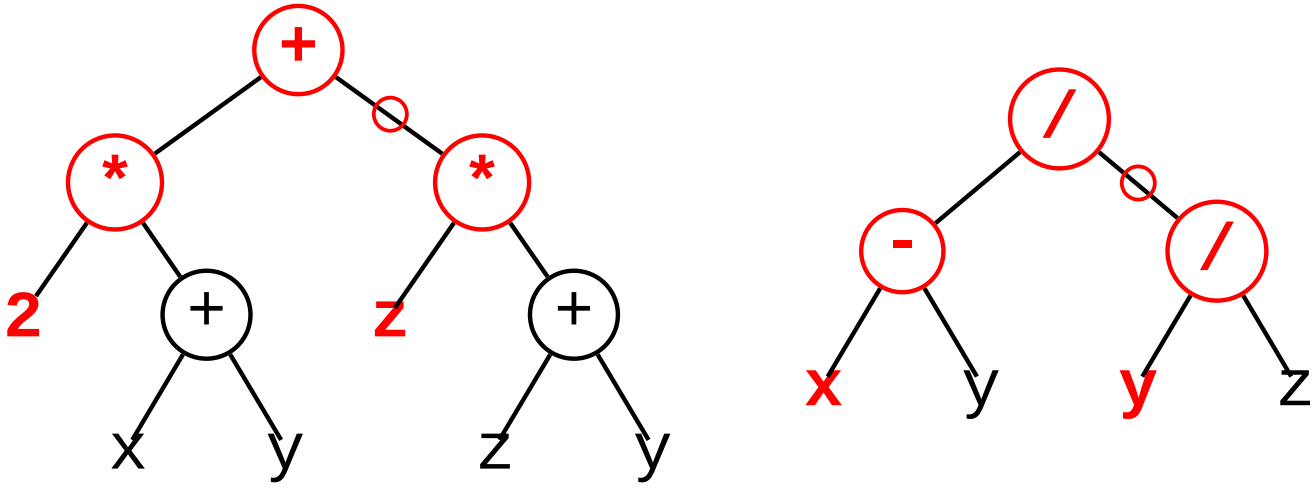

- Align the two parents
- Identify the region which is struct. common

# GP One-Point Crossover




- Align the two parents
- Identify the region which is struct. common
- Choose a x-over pt from within that region

# GP One-Point Crossover




- Align the two parents
- Identify the region which is struct. common
- Choose a x-over pt from within that region
- Perform subtree crossover at that point

# Poli-Langdon Theorem (pess, 1-pt x-over)

$$E[m(s,t+1)] \geq n \cdot \underbrace{p(s,t)}_{\text{selection}} \underbrace{(1-p_m)^{O(s)}}_{\text{mutation}}$$

$$\left\{1 - p_c \left[p_{diff}(t)(1 - p(G(s),t) + \frac{L(s)}{N(s)}(p(G(s),t) - p(s,t))\right]\right\}$$

- Pessimistic ST for 1-pt crossover using any selection

# Poli-Langdon Theorem (pess, 1-pt x-over)

$$E[m(s,t+1)] \geq n \cdot \underbrace{p(s,t)}_{\text{selection}} \underbrace{(1-p_m)^{O(s)}}_{\text{mutation}}$$

$$\left\{ 1 - p_c \left[ \underbrace{p_{diff}(t)(1-p(G(s),t)}_{\text{xover w/ parent} \notin G(s)} + \frac{L(s)}{N(s)}(p(G(s),t) - p(s,t)) \right] \right\}$$

- Pessimistic ST for 1-pt crossover using any selection
- Prob, of crossing over with a parent not in $G(s)$

# Poli-Langdon Theorem (pess, 1-pt x-over)

$$E[m(s,t+1)] \geq n \cdot \underbrace{p(s,t)}_{\text{selection}} \underbrace{(1-p_m)^{O(s)}}_{\text{mutation}}$$

$$\left\{ 1 - p_c \left[ \underbrace{p_{diff}(t)(1-p(G(s),t))}_{\text{xover w/ parent} \notin G(s)} + \underbrace{\frac{L(s)}{N(s)}(p(G(s),t)-p(s,t))}_{\text{xover w/ parent} \in G(s)} \right] \right\}$$

- Pessimistic ST for 1-pt crossover using any selection
- Prob, of crossing over with a parent not in $G(s)$
- Prob, of crossing over with a parent in $G(s)$

# Outline of Discussion

Part I: Overview of Schema Theory $\surd$

Part II: Pessimistic Schema Theory $\surd$

Part III: Exact Schema Theory $\leftarrow$

Part IV: Conclusions

# Exact Schema Theory

- Two added elements:
  - Tight bounds on $E[m(s,t+1)]$ by forming an equality, rather than an inequality
  - Now able to estimate variance, and thus determine the certainty of the estimate

# Exact Schema Theory

- Two added elements:
  - Tight bounds on $E[m(s,t+1)]$ by forming an equality, rather than an inequality
  - Now able to estimate variance, and thus determine the certainty of the estimate

*Account for creation & survival of schema*

*Assume parents are selected independently*

# Exact Schema Theory

- Two added elements:
  - Tight bounds on $E[m(s,t+1)]$ by forming an equality, rather than an inequality
  - Now able to estimate variance, and thus determine the certainty of the estimate

*Account for creation & survival of schema*

*Assume parents are selected independently*

**"Exact" in what sense?**

Because it is possible to predict with a known certainty whether $m(s,t+1)$ will be above a certain threshold, the expectation operator can be removed from the theorem.

# Transmission Probability

- Want to know $\alpha(s,t)$:
  - The probability that at generation $t$, individuals produced via genetic operators (including selection & cloning) will be in schema $s$.
  - $\alpha(s,t) = Pr[s \text{ survives}] + Pr[s \text{ constructed}]$
- $\alpha(s,t)$ can be quite difficult to elcit

# Transmission Probability

- Want to know $\alpha(s,t)$:
  - The probability that at generation $t$, individuals produced via genetic operators (including selection & cloning) will be in schema $s$.
  - $\alpha(s,t) = Pr[s \text{ survives}] + Pr[s \text{ constructed}]$
- $\alpha(s,t)$ can be quite difficult to elcit
- But assuming we knew it, if parents are selected independently...

# Transmission Probability

- Want to know $\alpha(s,t)$:
  - The probability that at generation $t$, individuals produced via genetic operators (including selection & cloning) will be in schema $s$.
  - $\alpha(s,t) = Pr[s \text{ survives}] + Pr[s \text{ constructed}]$
- $\alpha(s,t)$ can be quite difficult to elcit
- But assuming we knew it, if parents are selected independently...

**Then $m(s,t)$ is binomially distributed!**

$$\therefore \quad Pr[m(s,t+1)] = \binom{n}{k} \alpha(s,t)^k \left(1-\alpha(s,t)\right)^{n-k}$$

# Distribution of $m(s,t+1)$

- Now we can compute expectation and variance exactly:
  - $E[m(s,t+1)] = n \cdot \alpha(s,t)$
  - $Var[m(s,t+1)] = n \cdot \alpha(s,t)\ (1-\alpha(s,t))$
- Now we can compute a probabilistic ST:
  - Using Chebyshev's inequality & some algebra
  - $Pr\left[m(s,t+1) > n\alpha(s,t) - k\sqrt{n\alpha(s,t)(1-\alpha(s,t))}\right] \geq 1 - \frac{1}{k^2}$

    for any given constant $k$
  - Now have a relationship between the bounding condition and accuracy of the prediction

# Stephens & Waelbroeck's ST

- Exact ST for GAs with fixed-length binary representations
- One-point crossover (no mutation)

$$\alpha(s,t) = (1-p_c)p(s,t) + \frac{p_c}{l-1}\sum_{i=1}^{l-1} p\left(L(s,i),t\right)\ p\left(R(s,i),t\right)$$

where:

$L(s,i)$ replaces right $(i+1)\ldots l$ positions with "*"

$R(s,i)$ replaces left $1\ldots i$ positions with "*"

# Fixed Size/Shape GP ST

- All programs have same size and shape
- One-point GP crossover

$$\alpha(s,t) = (1-p_c)p(s,t) + \frac{p_c}{N(s)} \sum_{i=1}^{N(s)-1} p\left(l(s,i),t\right)\ p\left(u(s,i),t\right)$$

where:

$N(s)$ is number of nodes in schema $s$

$l(s,i)$ repl all nodes above x-over pt $i$ with "=" nodes

$u(s,i)$ repl all nodes below x-over pt $i$ with "=" nodes

# Hyperschemata

- Set of functions is denoted, $\mathcal{F}$
- Set of terminals is denoted, $\mathcal{T}$
- The "=" symbol is "don't care" for a *single* function or terminal
- The "#" symbol is "don't care" for any *valid subtree*
- Schema $s$ is a rooted tree composed of:
  - Internal nodes from $\mathcal{F} \cup \{=\}$
  - Leaf nodes from $\mathcal{T} \cup \{=, \#\}$

# Hyperschema Example

**For example:** $\mathcal{F} \in \{+,-,*,/\}, \mathcal{T} \in \{1,\ldots,9,x,y,z\}$

```
(+ (* 2 (+ x y)) (/ y z))
```

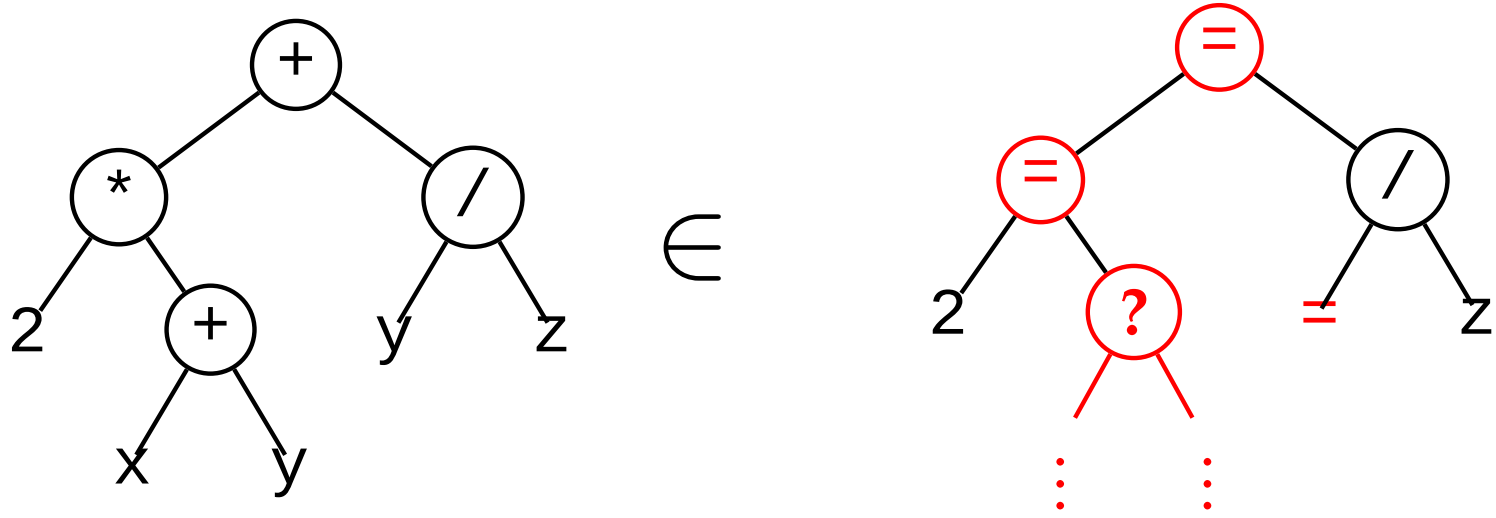
$\in$ `(= (= 2 #) (/ = z))`

# Hyperschema Example

**For example:** $\mathcal{F} \in \{+,-,*,/\}, \mathcal{T} \in \{1,\ldots,9,x,y,z\}$

```
(+ (* 2 (+ x y)) (/ y z))
```

$\in$ `(= (= 2 #) (/ = z))`



# Hyperschemata Notation/Measures

Let's now denote a member of $P_t$ as $v_i$

- No. nodes in tree fragment rep. common region b/w programs $v_1$ and $v_2$ is $NC(v_1, v_2)$
- Set of indices of crossover points in common region is $C(v_1, v_2)$
- Durak delta function, $\delta(x) = 1$ if x is true, and 0 otherwise
- $L(s, i)$ is hyperschema obtained by replacing all nodes on path between x-over pt. $i$ and root node with "=" nodes and all subtrees connected to those nodes with "#"
- $U(s, i)$ is hyperschema obtained by replacing all nodes below x-over pt. $i$ with "#" node

# Microscopic Exact GP ST

- Valid for populations of programs of any size and shape
- Generalization of Rosca's & P/L schemata (hence "hyper")
- One-point crossover, no mutation
- "Microscopic" since it must consider in detail each member of the population

$$\alpha(s,t) = (1-p_c)\,p(s,t) + p_c \sum_{v_1} \sum_{v_2} \frac{p(v_1,t)p(v_2,t)}{NC(v_1,v_2)} \sum_{i \in C(v_1,v_2)} \delta(v_1 \in L(s,i))\delta(v_2 \in U(s,i))$$

Where first two sums are over all individuals in the population.

# Advantages of Macroscopic ST

- Way of "course-graining" the right-hand side
  - Do not have to consider each individual in population
  - Mainly we are interested in *average properties*
- Becomes a proper generalization of all ST so-far described
  - Holland's ST
  - Stephens & Waelbroeck's ST
  - Poli-Langdon's fixed size & shape ST
  - Poli-Langdon's microscopic exact ST

# Macroscopic Exact GP ST

- $G_i$ represents all possible schemata of order 0 of the $i^{th}$ fixed size & shape.
- Enumerating all fixed size & shape order 0 schemata, $G_1, G_2, \ldots$, we cover the entire search space
- One-point crossover, no mutation

$$\alpha(s,t) = (1-p_c)\,p(s,t) +$$
$$p_c \sum_j \sum_k \frac{1}{NC(G_j,G_k)} \sum_{i \in C(G_j,G_k)} p\left(L(s,i) \cap G_j, t\right) p\left(R(s,i) \cap G_k, t\right)$$

# Other Macroscopic GP ST

- Homologous crossover
- Node-invariant subtree-crossing crossover
- ∴ Standard crossover
- Linear, length-only ST:
  - Homologous crossover
  - Headless chicken crossover
  - Subtree mutation crossover

# GP and the Building Block Hypothesis

- O'Reilly (1995) suggests crossover too destructive, so no BBH for GP
- Langdon & Poli (for 1-pt crossover) suggest otherwise:
  - $L(s.i) \cap G_j$ and $U(s.i) \cap G_k$ behave like GA BBs
  - GP really builds solutions via construction of lower order BBs
  - But GP BBs do not have to be above average fitness, short, or even of particularly low order

# Outline of Discussion

| | | |
|---|---|---|
| Part I: | Overview of Schema Theory | $\surd$ |
| Part II: | Pessimistic Schema Theory | $\surd$ |
| Part III: | Exact Schema Theory | $\surd$ |
| Part IV: | Conclusions | $\leftarrow$ |

# Traditional versus Exact Schema Theory

| **Traditional** | **Exact** |
|---|---|
| GA/Binary representation | Main focus has been in GP |
| Fixed length rep. | Variable length rep. |
| ”Pessimistic” inequality provides lower bound on expectation | Uses equality, so correctly predicts the expectation |
| | Can compute probability of expectation |

# Riccardo Poli's View of GP ST

- Compatibility with existing theory
  - GP Schema Theory is a superset of GA Schema Theory

    GAs are a subset of GP
  - Overlaps with dynamical systems & Markov modeling

# Riccardo Poli's View of GP ST

- Compatibility with existing theory
  - GP Schema Theory is a superset of GA Schema Theory
    GAs are a subset of GP
  - Overlaps with dynamical systems & Markov modeling
- Framework for est. specialized theorems
  - There is no "correct" schema theorem, it depends on the context of the question
    Must find the right level of granularity
  - Different operators lead to different ST

# Riccardo Poli's View of GP ST

- Compatibility with existing theory
  - GP Schema Theory is a superset of GA Schema Theory
    GAs are a subset of GP
  - Overlaps with dynamical systems & Markov modeling
- Framework for est. specialized theorems
  - There is no "correct" schema theorem, it depends on the context of the question
    Must find the right level of granularity
  - Different operators lead to different ST
- Valid and useful tool for researchers
  - The meaning of GP building blocks is clearer
  - Exact GP ST can help (and has helped) guide designers to create better representations, operators, and algorithms
  - Can tell (and has told) us useful things about how GP works...

# What *has* ST told us?

- Prediction:
  - Local estimations for schema propagation
  - Local estimations of average fitness improvement
  - Rank operators for given fitness function (derive Price's theorem)
- Explanation:
  - Understand *why* certain things happen (formally):
    - When algorithms behave differently for different initial populations
    - Rate changes of different selection operators
    - Why (and how) GP adds/removes individuals from a schema
  - Understand operator bias: ★
    - When mutation and recombination have creation/survival advantages over one another in GAs. (Spears, 2000)
    - Linear, 1-pt crossover is unbiased w.r.t. program length. (McPhee, 2001)
    - Standard crossover is biased toward longer programs, but in which primitives are uniformly distributed in quantity and position (generalized Geiringer's theorm). (Poli, 2002)
    - Combining operators *may* arrest growth from standard crossover (model validation studies). McPhee, 2002)

# What ST *hasn't* told us?

- How to build the perfect EA to solve problems of some particular problem class
- What kind of long term behaviors can we expect from our EAs
- Whether or not EAs (nearly) converge to (near) optimal solutions and under what contexts
- How long we can expect to wait for an EA to converge
- What kind of car Sean should buy, without verbal feedback

# What ST *hasn't* told us?

- How to build the perfect EA to solve problems of some particular problem class (Omnipotence)
- What kind of long term behaviors can we expect from our EAs
- Whether or not EAs (nearly) converge to (near) optimal solutions and under what contexts
- How long we can expect to wait for an EA to converge
- What kind of car Sean should buy, without verbal feedback

# What ST *hasn't* told us?

- How to build the perfect EA to solve problems of some particular problem class
  (Omnipotence)
- What kind of long term behaviors can we expect from our EAs
  (Dynamical systems, Global analysis)
- Whether or not EAs (nearly) converge to (near) optimal solutions and under what contexts
- How long we can expect to wait for an EA to converge
- What kind of car Sean should buy, without verbal feedback

# What ST *hasn't* told us?

- How to build the perfect EA to solve problems of some particular problem class
  (Omnipotence)
- What kind of long term behaviors can we expect from our EAs
  (Dynamical systems, Global analysis)
- Whether or not EAs (nearly) converge to (near) optimal solutions and under what contexts
  (Dynamical systems?)
- How long we can expect to wait for an EA to converge
- What kind of car Sean should buy, without verbal feedback

# What ST *hasn't* told us?

- How to build the perfect EA to solve problems of some particular problem class
  (Omnipotence)
- What kind of long term behaviors can we expect from our EAs
  (Dynamical systems, Global analysis)
- Whether or not EAs (nearly) converge to (near) optimal solutions and under what contexts
  (Dynamical systems?)
- How long we can expect to wait for an EA to converge
  (Global analysis)
- What kind of car Sean should buy, without verbal feedback

# What ST *hasn't* told us?

- How to build the perfect EA to solve problems of some particular problem class
  (Omnipotence)
- What kind of long term behaviors can we expect from our EAs
  (Dynamical systems, Global analysis)
- Whether or not EAs (nearly) converge to (near) optimal solutions and under what contexts
  (Dynamical systems?)
- How long we can expect to wait for an EA to converge
  (Global analysis)
- What kind of car Sean should buy, without verbal feedback

  (Omnipotence, Telepathy)

# References

Goldberg, D. Genetic Algorithms in Search, Optimization and Machine Learning. 1989

Langdon, W. and Poli, R. Foundations of Genetic Programming. 2002

Langdon, W. and Poli, R. Tutorial on Foundations of Genetic Programming. In GECCO 2002 Tutorials. 2002

Michalewicz, Z. Genetic Algorithms + Data Structures = Evolution Programs, 3rd Revised and Extended Edition. 1996

McPhee, N. et al. A schema theory analysis of the evolution of size in genetic programming with linear representations. In EUROGP 2001 Proceedings. 2001

Poli, R. et al. On the Search Biases of Homologous Crossover in Linear Genetic Programming and Variable-length Genetic Algorithms. In GECCO 2002 Proceedings. 2002

Poli, R. and McPhee, N. Markov chain models for GP and variable-length GAs with homologous crossover. In GECCO 2001 Proceedings. 2001

Spears, W. Evolutionary Algorithms: The Role of Mutation and Recombination. 2000

Stephens, C. and Waelbroeck. Schemata Evolution and Building Blocks. In Evolutionary Computation 7(2). 1999