Overview

- Automated Theorem Prover (ATP)
- Genetic Programming (GP)
- Genetic Reasoning
- GR-ATP-Genotype Encoding
- GR-ATP-Phenotype Encoding
- GR-ATP-Fitness Function
- Benchmarks
- Results
- Conclusion
- Questions

Automated Theorem Proving

- A subfield of automated reasoning
  - Which is itself a subfield of artificial intelligence
- Deals with the proving theorems using computer programs, heuristics, AI... (With or without human involvement)
- Involves generating a theorem as a conclusion of previously established statements, such as other theorems/accepted statements/axioms.

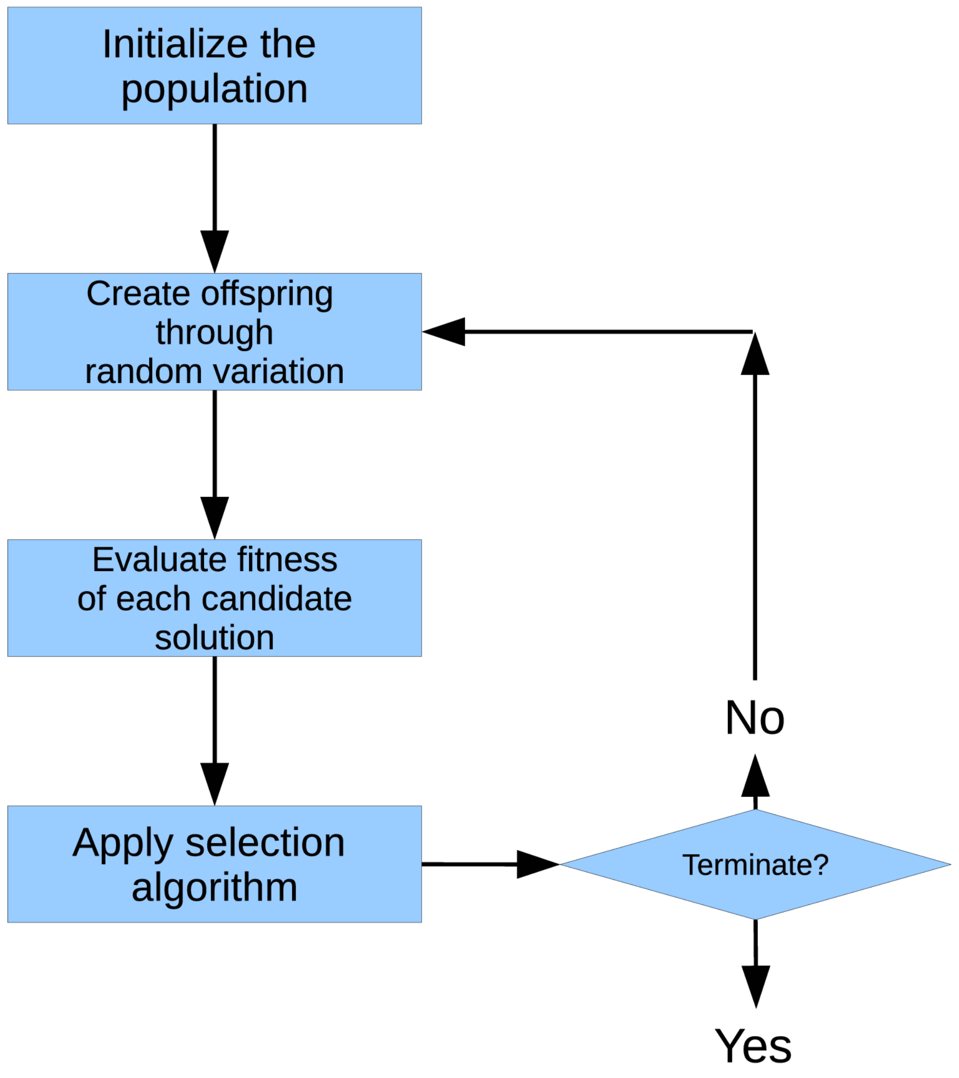
Automated Theorem Proving

- Simplest form: systematically apply rules of inference to construct all possible valid logical deductions.
  - 1950: Logic Theory Machine of Alan Newell and Herbert Simon
- In practice can find only very simple short proofs, because combinatorial explosion quickly exhaust any computer resources.
- Modern versions are just enhancements of the same old approaches: Heuristics/Exhaustive search (Large libraries of lemmas...), and various ways to prune the combinatorial explosion of possibilities of putting the parts together. Search algorithm, most rely on forms of hill-climbing algorithm, back-tracking or a best-first heuristic...
- Modern popular techniques are based on:
  - First-order resolution with unification, Lean theorem proving, Model elimination, Method of analytic tableaux, Superposition and term rewriting, Model checking, Binary decision diagrams, Higher-order unification
- Evolution is a general, and very powerful global search algorithm. Could it be the answer?

Evolutionary Computation

- Nature inspired search algorithm
- A global search, and has lower chance of getting stuck in local optima
- A simple algorithm, easy to parallelize
    1. Generate initial population
    2. Evaluate fitness of each agent
    3. Choose fit agents
    4. Produce new generation of agents through variation (mutation and/or crossover)
    5. Goto Step-2

Evolutionary Computation

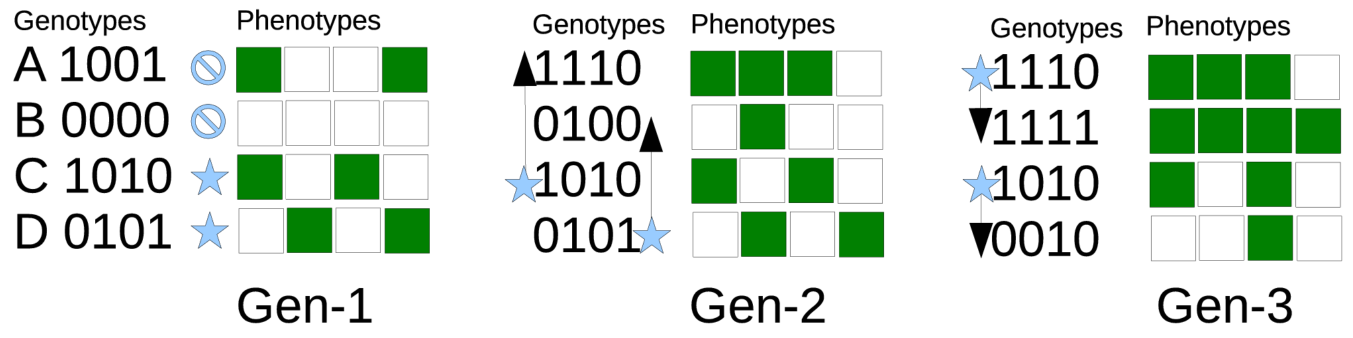

Initialize the population
Create offspring through random variation
Evaluate fitness of each candidate solution
Apply selection algorithm
Terminate?
No
Yes

Evolutionary Computation Approaches

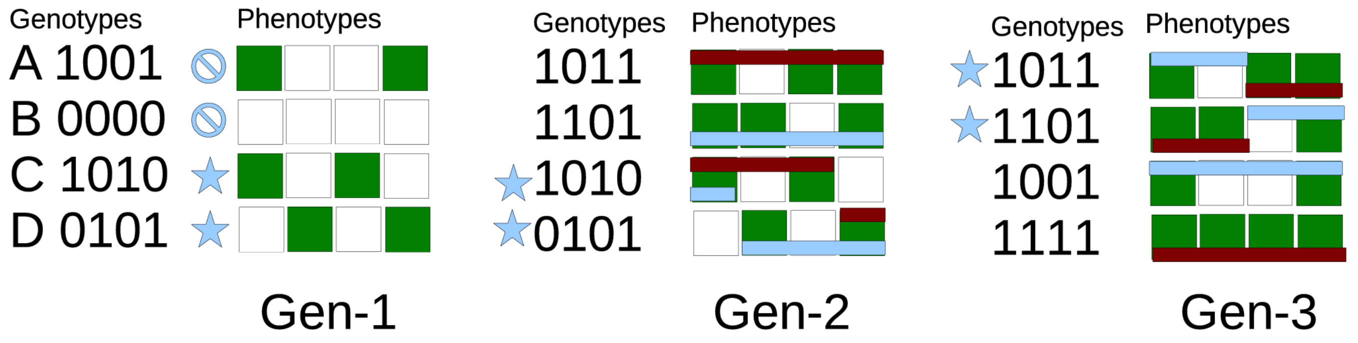
- Genetic Algorithms (John Holland, 73-75)
  - Population of fixed length genotypes, bit strings, evolved through perturbation/crossing
- Genetic Programming (John Koza, 92)
  - Variable sized chromosome based programs represented as treelike structures, with specially crafted genetic operators
- Evolutionary Strategies (Ingo Rechenberg, 73)
  - Normal distribution based, adaptive perturbations (self-adaptation)
- Evolutionary Programming (L. & D. Fogel, 63)
  - Like ES, but for evolution of state transition tables for finite-state machines (FSMs)

Genetic Algorithm

Simple Mutations

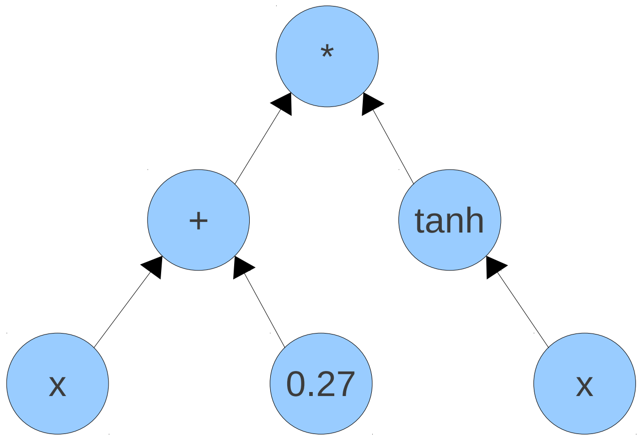


Crossover



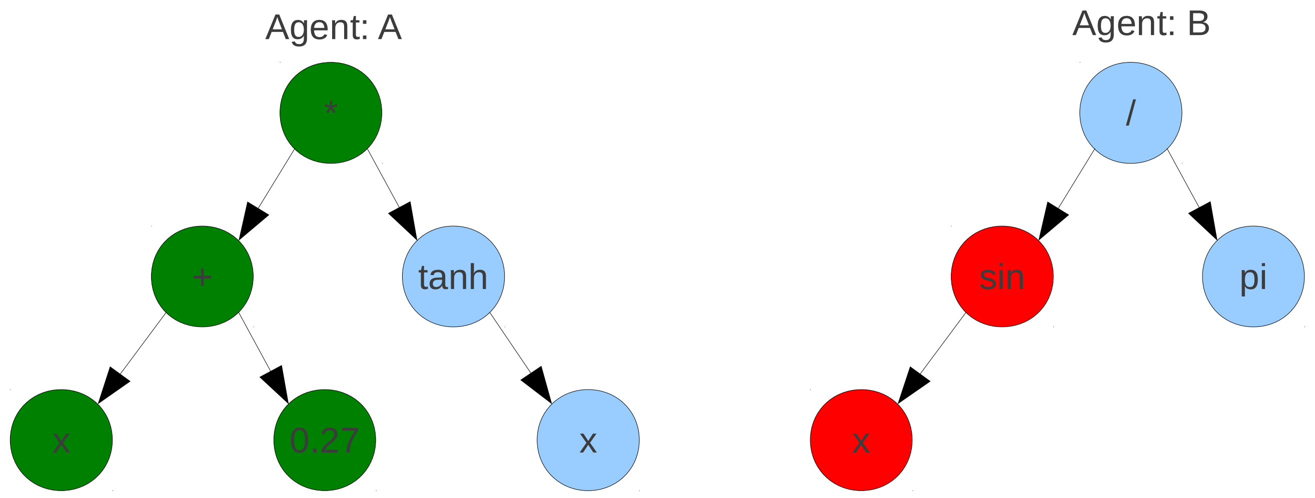
Genetic Programming

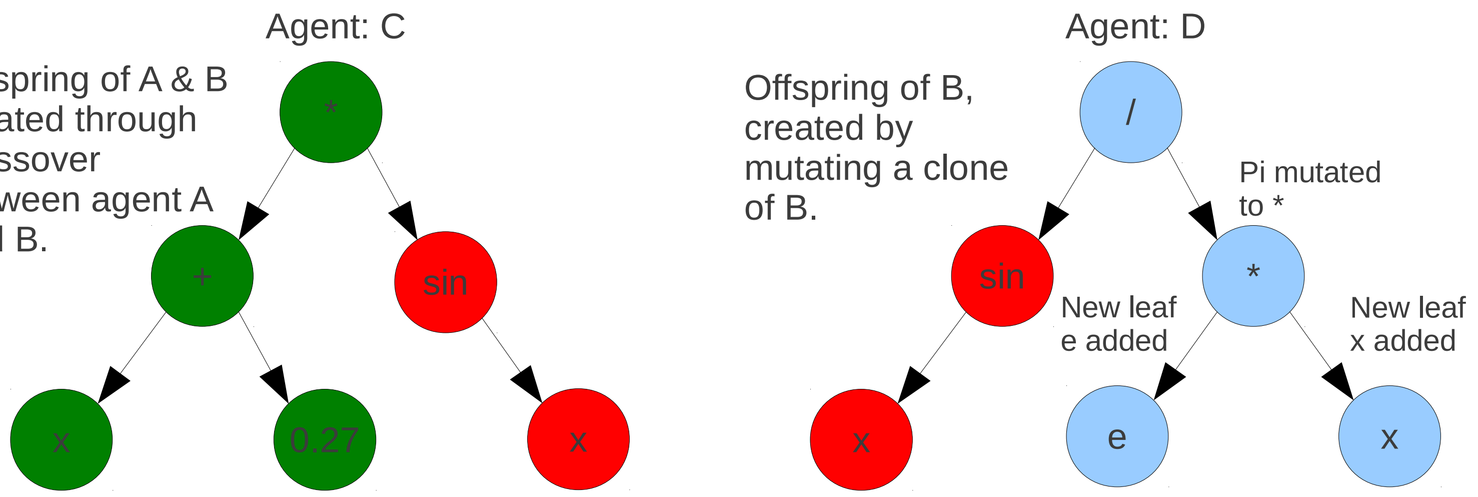
Tree encoded genotype:



Phenotype: (x+0.27)*tanh(x)

Genetic Programming





Offspring of A & B created through crossover between agent A and B.

Offspring of B, created by mutating a clone of B.

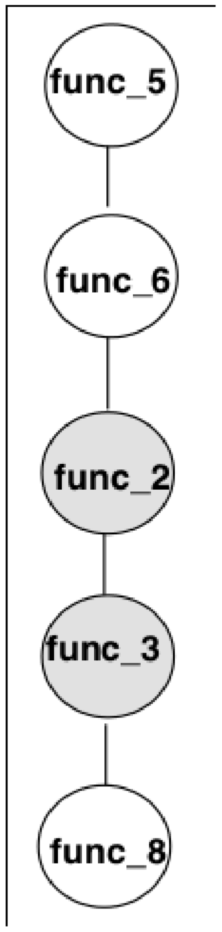
Genetic Reasoning Based ATP

- Genetic programming applied to ATP, what do we need?
- Standard of all evolutionary computation based systems:
  - agent/solution/genotype representation
  - problem/theorem representation
  - mutation/crossover methods that can be applied to agent's representation
  - fitness function

•Agent/Solution/Genotype representation

- Linear genotype composed of rules of inference.
  - The solution is a list of replacement rules (production rules). System should be able to handle statements about arithmetics in a logic as powerful as first order logic. Showing them to either be true or false.
  - In paper: quantifier-free first order logic with equality. The rules of inference are from propositional logic and equality.
- An available pool of functions, where each function represents a rule of inference.
  - Exp: X + 0 => X
  - Universal quantification is indicated by leaving variables free. Existential quantification is represented by a Skolem function, as common in several approaches to ATP.
  - Natural numbers presented in the form of the zero (0) symbol and the successor function.

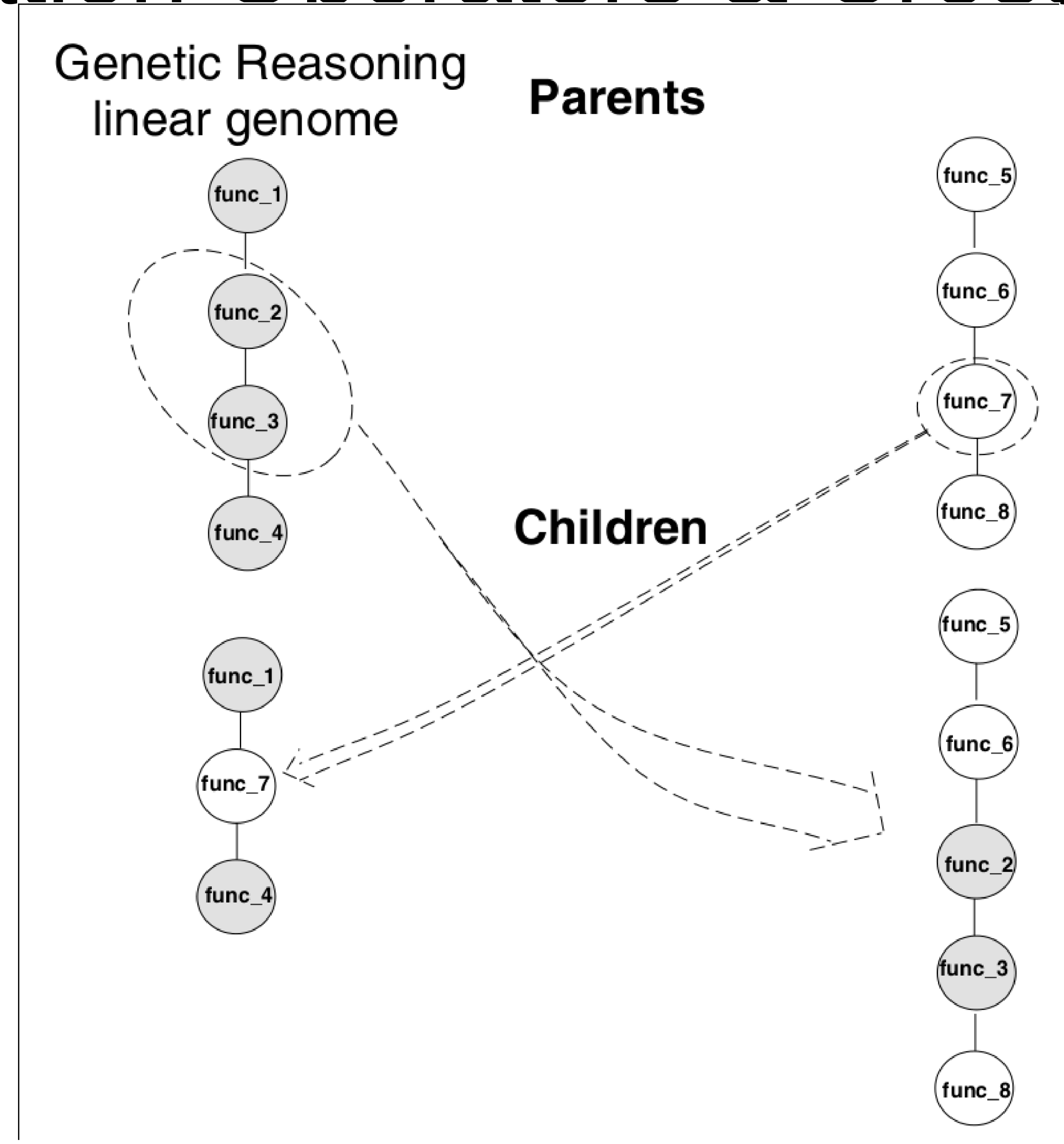
Genotype



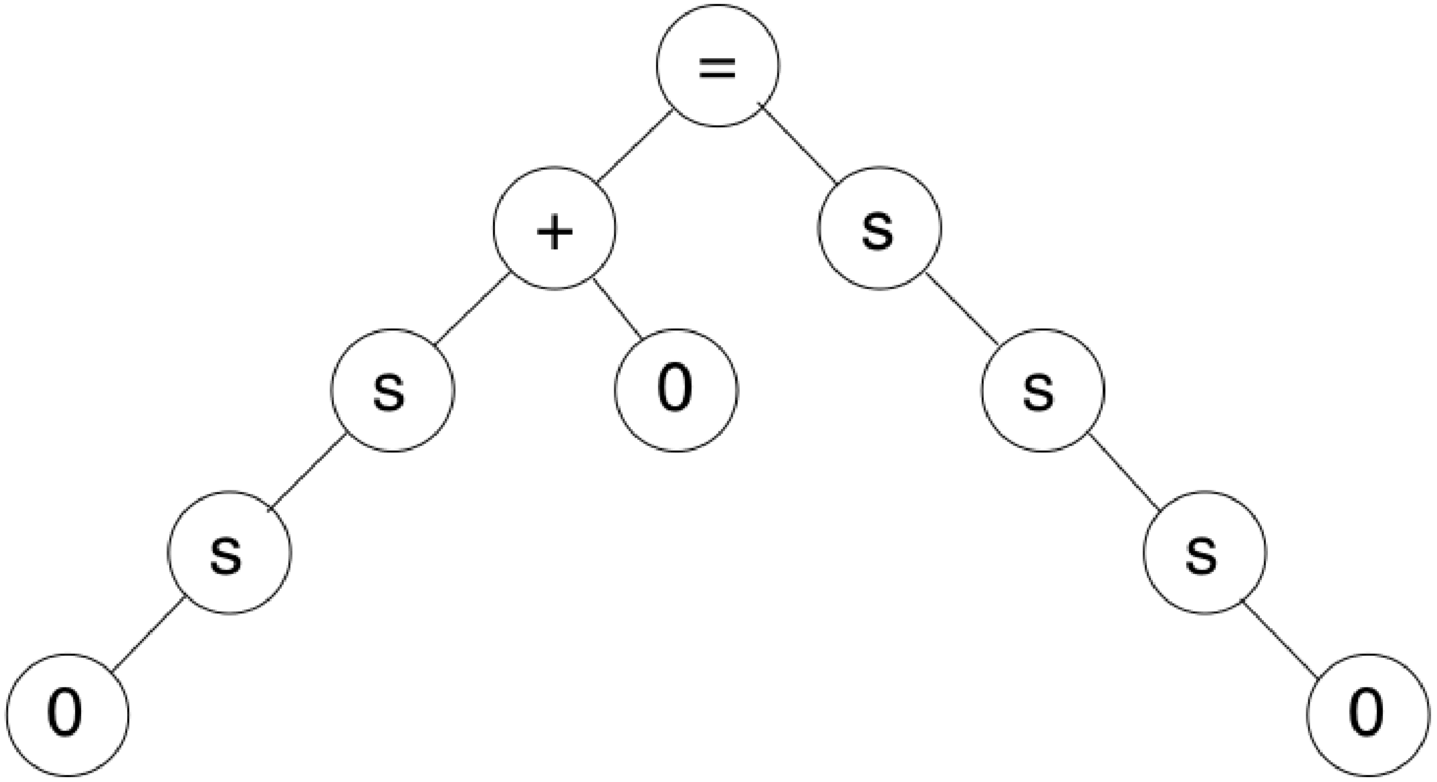
Mutation Operators & Crossover

- Steady state replacement with a tournament selection of four individuals.
- Mutations of the children occur with a specified probability and intensity. A number of positions are selected and replaced randomly according to the intensity specified by the researcher.
- Population size: Varies (chosen by researcher, depends on problem difficulty).
- Initial genotype: Minimal, randomly chosen initial list of replacement rules, ranging in size from 1 to 1/3 of the number of nodes in the problem.

Mutation Operators & Crossover

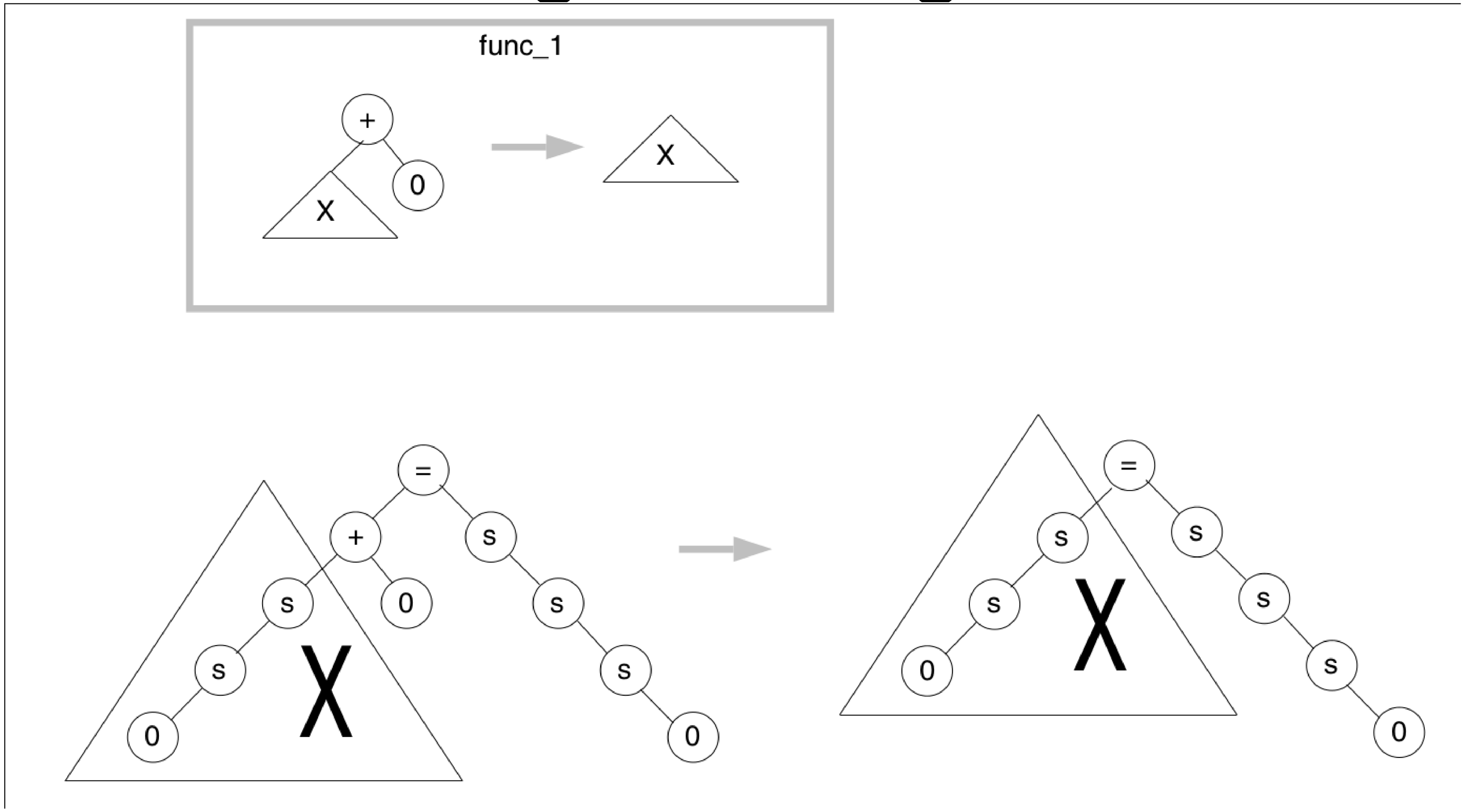


Problem/Theorem Representation



Representation of the statement 3 = 2 + 0.

Putting it all together



Application of func_1 to the statement 3 = 2 + 0.
Where func_1(x) = x + 0 => x

Fitness Function

- In many ways this is the most difficult part in these types of problems.
- We need a smooth path towards greater fitness (a way to discern that one solution is better than another, that it's getting closer to the full proof)
- GR based ATP Fitness: # of nodes in tree after all replacement rules have been applied.
  - Solution: 1 node (true/axiom or false/contradiction)

Current state of GR based ATP

- As noted, at this time the system can only be applied to: quantifier-free first order logic with equality problems. The rules of inference are from propositional logic and equality.
- Functions defining all boolean arithmetic operations are built-in ($\wedge, \vee, \neg, \rightarrow$) are available.
- The natural numbers and arithmetics are defined by the Peano axioms and the symmetry relation.
- It is possible to add functions defining abstract data types and lemmas to support a specific application.

Benchmarks

- 3 problems dealing with algebraic simplification, standard axiomatization of the real numbers.
  - “*The axiomatization is not complete. For example, inequality and the axiom of choice are not represented. Nevertheless, it is sufficient for our purposes in this paper. The production rules and their representation are listed in table*”
- Two identities from basic algebra:
  - $A * 0 = 0$: (*A0)
  - $N * N = 1$: (*NN)
  - “*To prove these identities, one has to expand the initial expressions significantly before it is possible to apply a greedy approach, and only go ”downhill”.*”
- Difficult because it demands several operations without any fitness feedback: (+A(+(+B(N*A))C))

Available Replacement Rules

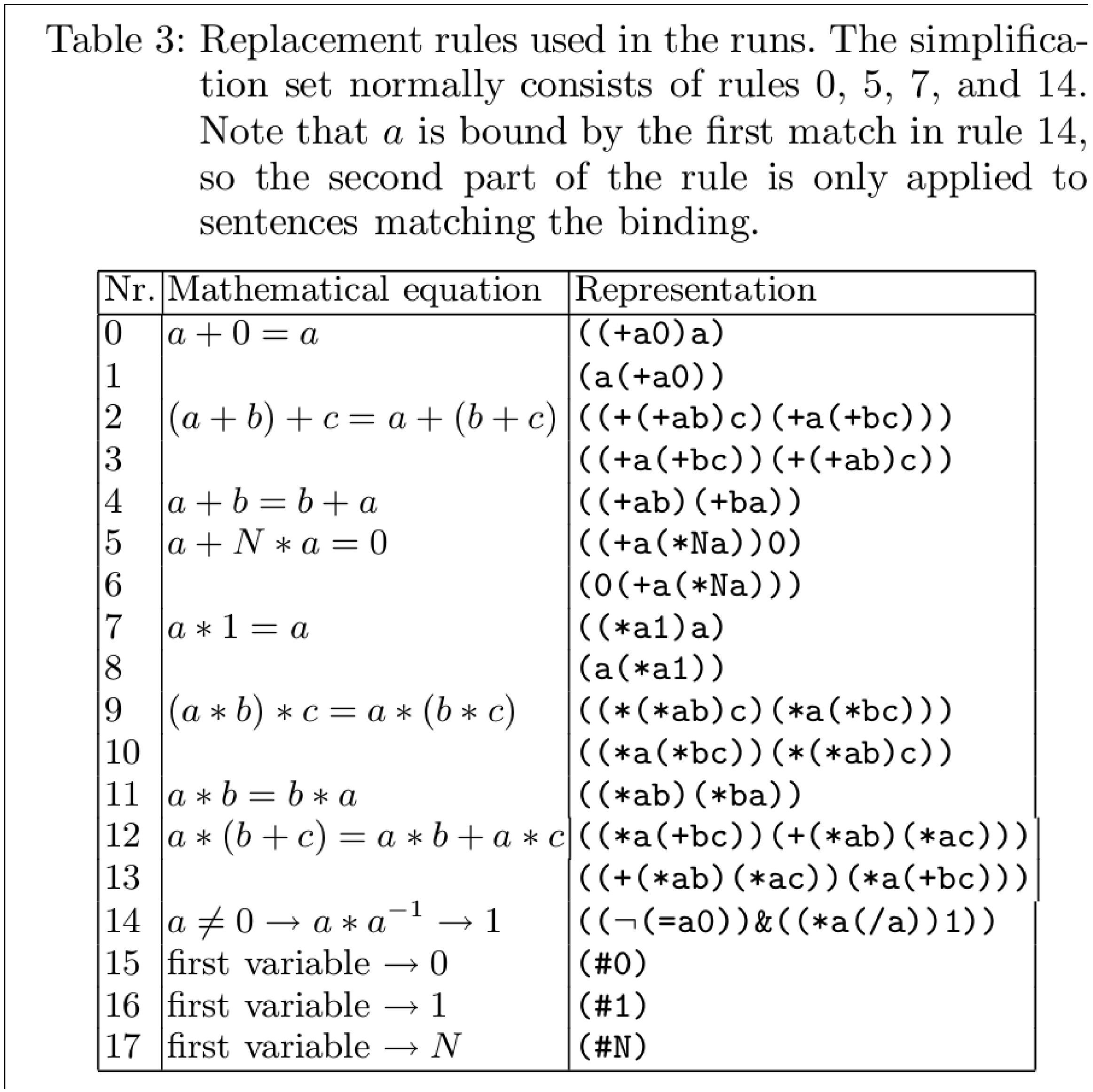


Table 3: Replacement rules used in the runs. The simplification set normally consists of rules 0, 5, 7, and 14. Note that a is bound by the first match in rule 14, so the second part of the rule is only applied to sentences matching the binding.

Nr.	Mathematical equation	Representation
0	$a + 0 = a$	`((+a0)a)`
1		`(a(+a0))`
2	$(a + b) + c = a + (b + c)$	`((+(+ab)c)(+a(+bc)))`
3		`((+a(+bc))(+(+ab)c))`
4	$a + b = b + a$	`((+ab)(+ba))`
5	$a + N * a = 0$	`((+a(*Na))0)`
6		`(0(+a(*Na)))`
7	$a * 1 = a$	`((*a1)a)`
8		`(a(*a1))`
9	$(a * b) * c = a * (b * c)$	`((*(*ab)c)(*a(*bc)))`
10		`((*a(*bc))(*(*ab)c))`
11	$a * b = b * a$	`((*ab)(*ba))`
12	$a * (b + c) = a * b + a * c$	`((*a(+bc))(+(*ab)(*ac)))`
13		`((+(*ab)(*ac))(*a(+bc)))`
14	$a \neq 0 \rightarrow a * a^{-1} \rightarrow 1$	`((¬(=a0))&((*a(/a))1))`
15	first variable $\rightarrow 0$	`(#0)`
16	first variable $\rightarrow 1$	`(#1)`
17	first variable $\rightarrow N$	`(#N)`

Results

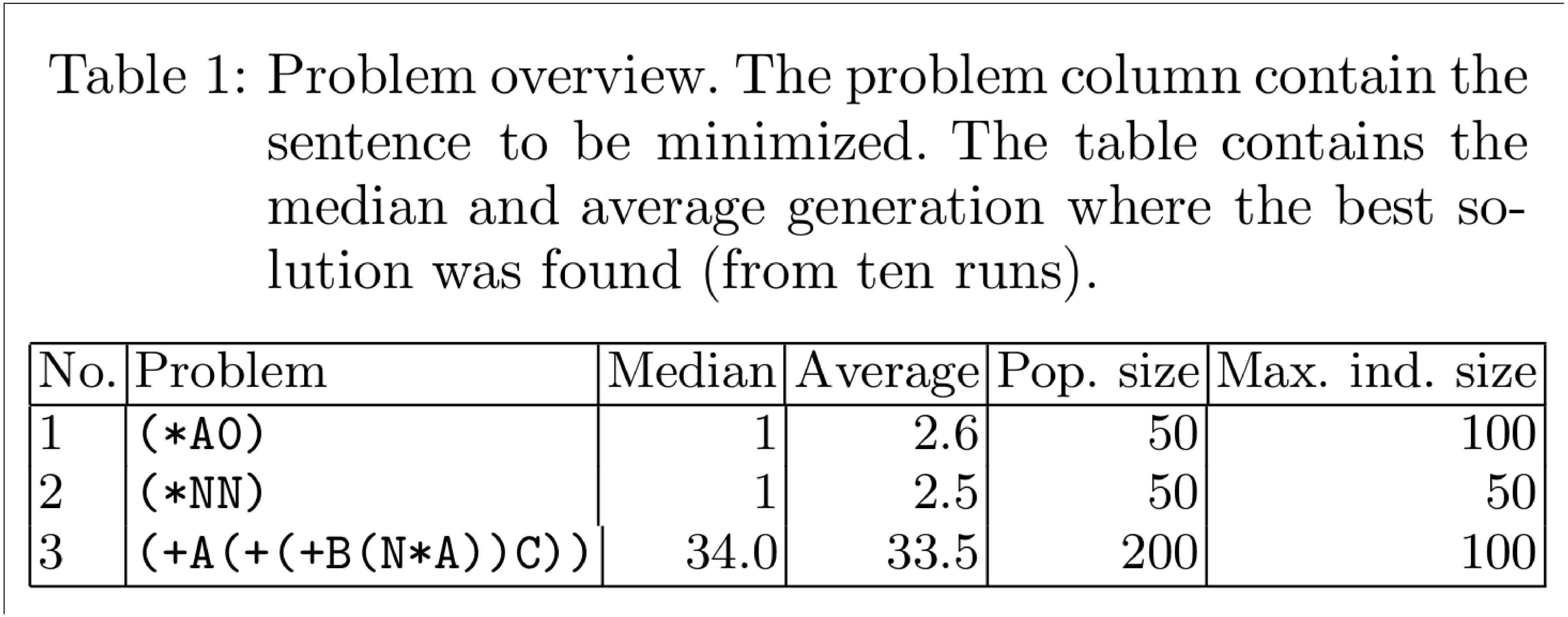


Table 1: Problem overview. The problem column contain the sentence to be minimized. The table contains the median and average generation where the best solution was found (from ten runs).

No.	Problem	Median	Average	Pop. size	Max. ind. size
1	`(*A0)`	1	2.6	50	100
2	`(*NN)`	1	2.5	50	50
3	`(+A(+(+B(N*A))C))`	34.0	33.5	200	100

Example Proof

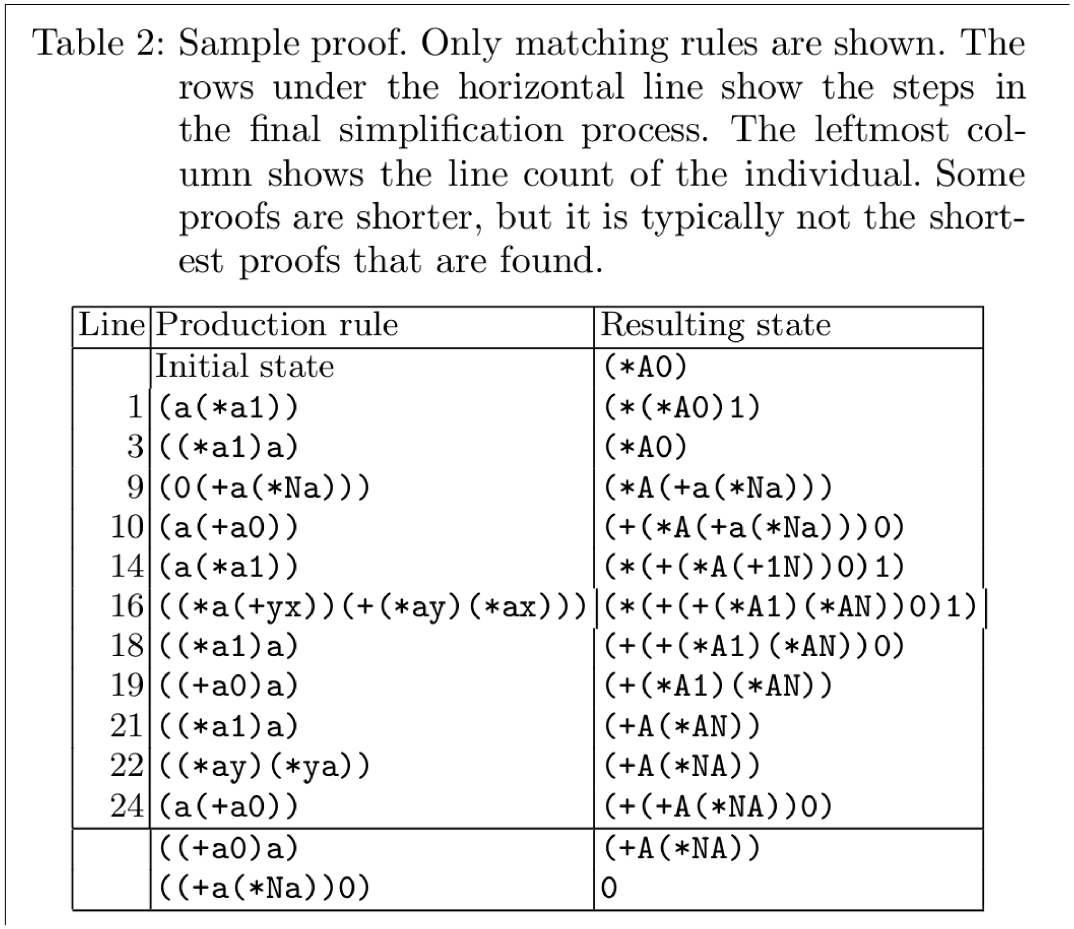


Table 2: Sample proof. Only matching rules are shown. The rows under the horizontal line show the steps in the final simplification process. The leftmost column shows the line count of the individual. Some proofs are shorter, but it is typically not the shortest proofs that are found.

Line	Production rule	Resulting state
	Initial state	`(*A0)`
1	`(a(*a1))`	`(*(*A0)1)`
3	`((*a1)a)`	`(*A0)`
9	`(0(+a(*Na)))`	`(*A(+a(*Na)))`
10	`(a(+a0))`	`(+(*A(+a(*Na)))0)`
14	`(a(*a1))`	`(*(+(*A(+1N))0)1)`
16	`((*a(+yx))(+(*ay)(*ax)))`	`(*(+(+(*A1)(*AN))0)1)`
18	`((*a1)a)`	`(+(+(*A1)(*AN))0)`
19	`((+a0)a)`	`(+(*A1)(*AN))`
21	`((*a1)a)`	`(+A(*AN))`
22	`((*ay)(*ya))`	`(+A(*NA))`
24	`(a(+a0))`	`(+(+A(*NA))0)`
	`((+a0)a)`	`(+A(*NA))`
	`((+a(*Na))0)`	`0`

Conclusion

- Not a very clear paper
- They did not continue this work, unfortunately, and there are only 3 papers that deal directly with this particular approach.
- I still found the approach rather elegant and viable, even if it was not presented in the best of ways.
- I'm currently working on extending it.

References

- [1] Nordin, Peter, and Wolfgang Banzhaf. Genetic reasoning evolving proofs with genetic search. Univ., Systems Analysis Research Group, 1996.
- [2] Nordin, Peter, Anders Eriksson, and Mats Nordahl. "Genetic reasoning: evolutionary induction of mathematical proofs." Genetic Programming. Springer Berlin Heidelberg, 1999. 221-231.
- [3] Dumitrescu, D. & Oltean, M. “An evolutionary Algorithm for theorem proving.” Studia Univ. “Babes-Bolayi”, Informatica, Volume XLIV , Number 2, 1999

Questions

- A better fitness function?
- Hybridization with another ATP?
- Why so few papers with this approach, are there particular innate flaws in applying EC to ATP?