A Genetic Programming Approach to Support the Design of Service Compositions

Lerina Aversano, Massimiliano Di Penta, and Kunal Taneja
RCOST - Research Centre on Software Technology
University of Sannio
Palazzo ex Poste, Via Traiano 1, 82100 Benevento, Italy
aversano, dipenta, tankunal@unisannio.it
Outline

- Motivation
- Approach
  - The objective
  - Finding a solution with Genetic Programming
    - Representation
    - Operators
    - Fitness function
- Illustrative example
- Conclusion and work in progress
Motivation

- Users dealing with the development of service oriented system are interested to find operations that accomplish a functional requirement. However… sometimes the operation suited for our needs is not available.

- A key promises of building applications with web services is that one should be able to compose a given set of service operations to fulfill the application’s needs.

  it is becoming particularly important to have methods and tools that aid the design of service compositions that can accomplish a functional requirement.
The objective

The objective of this work is to determine service compositions that realizes a given functional goal.

A single service returning these information does not exist, but an aggregation of services can meet the requirement.
Using Genetic Programming

Determining the best service compositions can be viewed as an optimization problem:

maximize a fitness function defined considering the correspondence of the inputs outputs of the services in the composition with respect to the functional request.

... different techniques can be adopted...

we propose the GP search for a solution of the problem
GA and GP

GA originated with an idea of applying the biological principle of evolution in artificial systems.

GA is an iterative procedure that searches for the best solution of a given problem among a population, represented by a finite string of symbols, named the genome.

At each evolutionary step, individuals are evaluated using a fitness function and selected for reproduction using a selection operator.

- The evolution is made by means of two operators: the crossover operator and the mutation operator.

In the case of GP, the genome represents a program while the crossover and mutation operators represent ways to recombine two programs in new one.

- Successfully applications: grammar inference, electronic circuit generation [Koza ‘92, Koza ‘94]
encoding the problem: the genome

the genome is represented as a tree where non-terminal nodes represent composite service workflow constructs and terminal (leaf) nodes represent single services invoked inside the workflow.

Non-terminal nodes can be:
- Sequence nodes
- Switch nodes
- Flow nodes
- Loop nodes
crossover operator

The crossover operator randomly selects two nodes (i.e., subtrees) from two individuals and swaps them, producing two new trees (individuals)
**mutation operator**

The mutation operator randomly picks a node and changes it in:

- a non-terminal node, randomly selecting its type (sequence, switch, flow, loop) and adding to it some randomly selected (children),
- terminal nodes (i.e., single service invocations) it changes the node in a terminal node, randomly selecting the invoked service.
We adopted the following criteria:

1. a correct solution, i.e., a workflow that produces the desired output from the available inputs, scores 1
   - The matching of a single service is based on the service interface
2. Partial solutions are evaluated as follows: the fitness of an individual is calculated starting from the leaves, if a particular leaf produces outputs desired by the parent, the score is incremented by 0.01

\[
F_{goal}(g) = \omega_1 \cdot \text{num}_{sc}(g) + \omega_2 \cdot \text{num}_{snc}(g) + \omega_3 \cdot \sum_{i=1}^{n} \left( \text{num}_{nodes}(g) + \text{num}_{loops}(g) \right)
\]
GA Settings

- crossover and mutation are applied with probability 0.7 and 0.01 respectively.
- the selection operator is the roulette-wheel selection: individuals with the highest fitness function have the highest probability to reproduce.
- the initial population consists of randomly generated trees of depth 2 with a max of 5 nodes as leaves
- the defined weights are $w_1 = 1.0$, $w_2 = 0.75$ and $w_3 = 0.01$.
- a set of service interfaces defined by the authors, that use the entities defined in the "context" domain
An example of the results

Each request is specified by the input and the output of the service composition that an user aim to design.

`ServiceRequest1` specifies a request for compositions that receive Presentation and Person returns Application and PDA;
Work in progress

- Incorporating QoS constraints

- Handling constrained inputs and outputs.

- Assess the usefulness of the approach
  - With controlled experiments
  - Comparing with other, replicable approaches (e.g. Woolel)

- Apply the approach to more complex case studies involving sets of real services;