A GP Approach to QoS-Aware Web Service Composition including Conditional Constraints

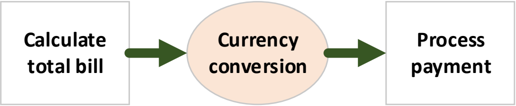
Alexandre Sawczuk da Silva, Hui Ma, Mengjie Zhang

Evolutionary Computation Research Group
School of Engineering and Computer Science, Victoria University of Wellington

IEEE Congress on Evolutionary Computation, 25-28 May 2015

Introduction

Service-Oriented Architecture (SOA): Organise processes and data in reusable modules for integration into new applications.

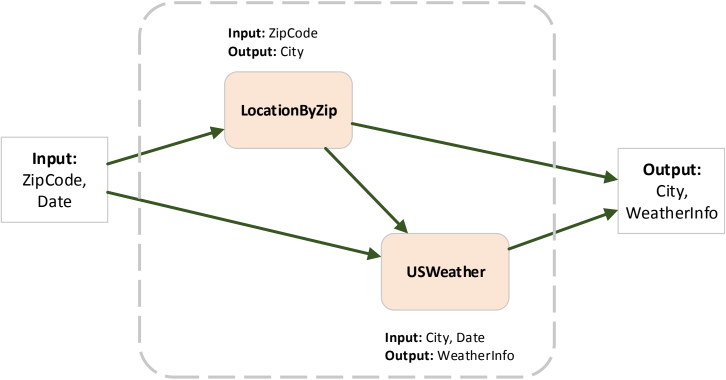


Web service

A functionality module that provides operations accessible over the network via a standard communication protocol.

Web Service Composition

The combination of Web services to achieve a more complex task.
Fully automated scenario:



New weather by zip code service

Composition Dimensions

1. **Functional correctness:** Service inputs and outputs must be properly linked (e.g. *FourDigitNumber* → *ZipCode*, but not *FourDigitNumber* → *City*).

2. **Conditional constraints:** Condition leading to multiple possible execution paths (e.g. if *City* is a *NewZealandCity*, produce *WindForecast* instead of *GeneralForecast*).

3. **Quality of Service (QoS):** The overall quality of the composition (e.g. lowest execution time, lowest cost).

Existing Approaches

AI Planning

Build a solution service by service.
Dimensions: *Functional correctness, conditional constraints.*

Evolutionary Computation (EC)

Improve population of solutions over multiple generations.
Dimensions: *Functional correctness, QoS.*

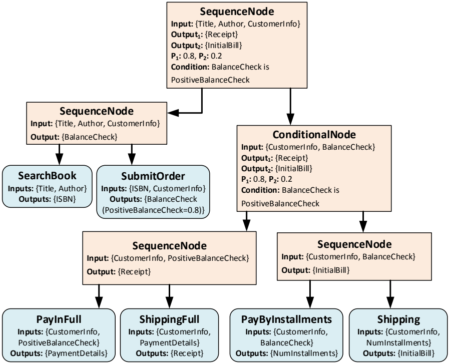
Hybrid Approaches

Combine AI planning and EC ideas.
Dimensions: *Functional correctness, QoS.*

Goal

To propose a Genetic Programming (GP) composition approach that simultaneously considers all dimensions.

1. Trees preserve functional correctness
2. Conditions encoded in trees
3. Optimisation performed on QoS

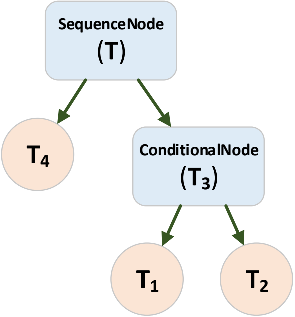


Candidate Representation

Population Initialisation

An algorithm is used to create a candidate in graph format, and then translate it into a tree representation.

```
Input  : I, O1, O2, C, P
Output: candidate tree T
if O2 ≠ ∅ then
    G1 ← createGraph(I ∪ C.if, O1);
    G2 ← createGraph(I∪C.else, O2);
    T1 ← toTree(G1.input);
    T2 ← toTree(G2.input);
    T3 ← new ConditionalNode(C);
    T3.leftChild ← T1;
    T3.rightChild ← T2;
    if C ⊑ I then
        T3.prob ← P;
        return T3;
    else
        G4 ← createGraph(I, C.else);
        T4 ← toTree(G4.input);
        T3.prob ← T4.final.P;
        T ← new SequenceNode();
        T.leftChild ← T4;
        T.rightChild ← T3;
        return T;
    end
else
    G ← createGraph(I, O1);
    T ← toTree(G.input);
    return T;
end
```



Mutation and Crossover

Fitness Function

Verbatim

Example (Theorem Slide Code)

```
\begin{frame}
\frametitle{Theorem}
\begin{theorem}[Mass--energy equivalence]
$E = mc^2$
\end{theorem}
\end{frame}
```

Figure

Uncomment the code on this slide to include your own image from the same directory as the template .TeX file.

Citation

An example of the `\cite` command to cite within the presentation:

This statement requires citation [Smith, 2012].

References

John Smith (2012)
Title of the publication
Journal Name 12(3), 45 – 678.

Thank you!

Questions?