Biopp

Hugo Arregui[1], Daniel Gutson[1]
1 FuDePAN, Córdoba, Argentina

Background

Biological alphabets manipulation is a recurring need in bioinformatics to work with RNA, DNA, proteins, pseudonucleotids, etc. This need is currently satisfied by software distributions capable of achieving this task in a more or less complete manner. All these distributions present some fundamental problems related to the inherent complexity of the alphabets multiplicity, which translates into complexity in types and interfaces that is challenging for tools users. In addition to that, there is a clear overhead required to check and detect incompatibilities which imposes a severe toll on computational resources as it normally takes place in program execution time.

Results

Combining Aspect Oriented Programming (AOP) and Generative Programming (GP, see reference 1) concepts, it was developed a library capable of managing combinational complexity and multiplicity in a clear and extremely flexible manner. For example, modeling alphabets as aspects, they can be easily combined in a production-chain fashion according to the GP paradigm, the error messages can be defined within the problem's context, avoiding the presentation of obscure or irrelevant information to the programmer. Additionally, by means of a C++ language functionality called template metaprogramming, it is possible to process type checking, compatibility checking and other overhead tasks in compilation time, freeing up computer resources during execution time to be used exclusively for its specific tasks, achieving in this manner significant reduction in processing time.

Conclusion

This library offers significant advantages over the existing tools in the following areas:
1) Reduction of effort and development time: simplified integration and interaction between modules by means of AOP and GP techniques, providing, at the same time, early detection of errors through static analysis of code.
2) Reduction of computational resources and processing times required for program execution: this is achieved by moving the overhead-related tasks (type checkings, etc) from execution time to compilation time by means of template metaprogramming in C++.

References

1) Generative Programming - Methods, Tools, and Applications by Krzysztof Czarnecki and Ulrich W. Eisenecker Addison-Wesley, June 2000