Deliverable D1.4

Funding Scheme: THEME [ICT-2007.8.0] [FET Open]

Paving the Way for Future Emerging DNA-based Technologies: Computer-Aided Design and Manufacturing of DNA libraries

Grant Agreement number: 265505
Project acronym: CADMAD

Deliverable number: D1.4

Deliverable name: GUI with full project management facilities and open APIs

<table>
<thead>
<tr>
<th>Contractual Date(^1) of Delivery to the CEC: M36</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual Date of Delivery to the CEC:</td>
</tr>
<tr>
<td>Author(s)(^2): Prof. N. Krasnogor</td>
</tr>
<tr>
<td>Participant(s)(^3): UNEW</td>
</tr>
<tr>
<td>Work Package: WP1</td>
</tr>
<tr>
<td>Security(^4): Pub</td>
</tr>
<tr>
<td>Nature(^5): R</td>
</tr>
<tr>
<td>Version(^6): 0.1</td>
</tr>
<tr>
<td>Total number of pages: 8</td>
</tr>
</tbody>
</table>

\(^1\) As specified in Annex I
\(^2\) i.e. name of the person(s) responsible for the preparation of the document
\(^3\) Short name of partner(s) responsible for the deliverable
\(^4\) The Technical Annex of the project provides a list of deliverables to be submitted, with the following classification level:
  - Pub - Public document; No restrictions on access; may be given freely to any interested party or published openly on the web, provided the author and source are mentioned and the content is not altered.
  - Rest - Restricted circulation list (including Commission Project Officer). This circulation list will be designated in agreement with the source project. May not be given to persons or bodies not listed.
  - Int - Internal circulation within project (and Commission Project Officer). The deliverable cannot be disclosed to any third party outside the project.
\(^5\) R (Report): the deliverables consists in a document reporting the results of interest.
\(^6\) Two digits separated by a dot:
  - The first digit is 0 for draft, 1 for project approved document, 2 or more for further revisions (e.g. in case of non acceptance by the Commission) requiring explicit approval by the project itself;
  - The second digit is a number indicating minor changes to the document not requiring an explicit approval by the project.
Deliverable D1.4

Abstract

This deliverable calls for a summary report on the APIs that the end-users, i.e., biologists, can utilise for programming a combinatorial DNA library by means of the DNALD language as well as for the reverse parsing planning algorithm. These have been kept simple and concise by design as we assume very little programming background from the end users.

Keywords:
DNALD, reverse parsing planning algorithm

Introduction

a. Aim / Objectives

(please note that these Aim/Objectives are identical to those in D1.5)

DNA programming is the DNA-counterpart of computer programming, that is, the basic computer programming cycle is to modify an existing program, test the modified program, and iterate until the desired behavior is obtained. Similarly, the DNA programming loop is to modify a DNA molecule, test its resulting behavior, and iterate until the goal (which is either understanding the behaviour or improving it) is achieved. One key difference between the two is that unlike computer programming, our understanding of DNA as programming language is very far from being perfect, and therefore trial and error are the norm rather than the exception in DNA-based research and development. Therefore, DNA programming is more efficient if multiple variants of a DNA program, also called a DNA library, are created and tested in parallel, rather than creating and testing just one program at a time. Hence the basic DNA programming cycle takes the best DNA programs from the previous cycle, uses them as a basis for creating a new set of DNA programs, tests them, and iterates until the goal is achieved, ideally, maximising reuse of DNA strands and library designs. The goal of WP1 is thus to create a fully functional combinatorial DNA library editor with possibility of strands reuse. The objectives are:

O1.1. Developing a textual DNA programming language (DNALD)
O1.2. Developing an open source graphical user interface (GUI)

In this deliverable Dr. Jonathan Blake worked until the 1s/November/2013, while Dr. Pawel Widera worked since then for the duration of the reporting period. Prof. Krasnogor worked throughout the reporting period. Please note that UNEW has not received payment from Nottingham university since September 2013.

This deliverable tackles O1.2, in particular Tasks 1.3 & Tasks 1.4, and successfully reaches milestone MS2 thus it should be considered successfully completed.

b. State of the Art

Keywords that would serve as search label for information retrieval
Certain tools for editing and manipulating DNA strings were developed long ago. In recent years a new system called GENOCAD\(^1\) was developed, offering a GUI based on context-free grammars, to design DNA molecules based on known building blocks. High-end commercial packages, e.g. DNA2.0\(^2\), Accelrys\(^3\), DNASTAR\(^4\), etc provide advanced features for designing cloning sequences and plasmids. These features include sophisticated GUI, project management, codon-usage optimizations, restriction site handling, etc. However, similarly to GENOCAD, they are intrinsically geared towards building high-level structured entities (e.g. plasmids) rather than multiple, combinatorially dependent DNA sequences, which are more likely to advance R&D and are the focus of this WP. In addition, these tools are limited in their expressive power – with most of them it is impossible to formally define degrees of freedom (for example using amino acids notations together with a custom defined codon table) which can very well ease the actual construction of the desired constructs. Existing tools are also not built for large libraries specifications, where some shared fragments can be defined and reused in several molecules, allowing fast generation of the specification.

c. Innovation

This deliverable is built on solid formal language theory, state-of-the-art software engineering and algorithms. We created a software suite that integrates our DNA library specification language (DNALD) into an integrated software development environment based on ECLIPSE. The specification of DNA libraries in the DNALD IDE is supported by a concise and powerful language (delivered in reporting period 2) with a state-of-the-art graphical user interface that is friendly, fast, robust and portable and that has been tested in Windows, Mac and Linux environments. Furthermore, our software enables extensive manipulation of a DNA molecule's representations while maximizing the use of existing DNA and minimizing the need for synthesizing new DNA, and permits the user to estimate a reverse parsing-planning route to manufacturing the libraries by means of a new and, currently, the fastest algorithm for DNA library design.

2. Implementation

This deliverable requires a summary report on the DNALD language API and the API for utilising the reverse parser and planning algorithm. Thus report is related to the functional software described in full in D1.5. and thus we have kept it concise; it should be read in conjunction with D1.5. We devide the implementation section into two areas, namely, the API for the DNALD language and the API for the reverse parser and planning algorithm.

API to the DNALD Language

The DNALD language is a rich and expressive programming language for combinatorial DNA library specification. Its programmatic API is built around essentially 8 family sets of operations, namely, (1) sequences definitions, (2) concatenations of molecular sequences, (3) repetition of sequences, (4) subsequence referencing, (5) sequences replacement, (6) special operators, (7) set operations on sequences and (8) backtranslations. We exemplify these DNAL APIs these below via simple examples.
1) Sequence definition

A name can be assigned to a sequence using ":=" operator. Sequence must be enclosed in single or double quotation marks. Non-letter characters in a sequence are ignored (e.g. numbers, space, line end). The case of the letters is not important.

\[ \text{alpha} := \text{atcgtac} \]
\[ \text{beta} := \text{"CTAGCTATAC"} \]

2) Concatenation

Sequences can be joined together using space character.

\[ \text{gamma} := \text{alpha beta} \quad \text{(result is "atcgtacCTAGCTATAC")} \]

3) Repetition

Sequences can be repeated a given number of times.

\[ \text{repetition} := \text{alpha} \ast 2 \quad \text{(result is "atcgtacatcgtac")} \]

4) Sub-sequences

A fragment of the sequence can be extracted using the square bracket operator. The operator argument is character index, or range of indices or "end" keyword denoting the last character in the sequence.

\[ \text{nucleotide} := \text{alpha}[2] \quad \text{(result is "t")} \]
\[ \text{subsequence1} := \text{beta}[2:4] \quad \text{(result is "TAG")} \]
\[ \text{subsequence2} := \text{beta}[7:end] \quad \text{(result is "ATAC")} \]

5) Replacement

The basic sequence modifications can be applied using extended square bracket syntax.

\[ \text{mutation} := \text{alpha}[3:4="tt"] \quad \text{(result is "attttac")} \]
\[ \text{insertion} := \text{alpha}[3="aaac"] \quad \text{(result is "ataaacgtac")} \]
\[ \text{deletion} := \text{alpha}[3:5=""] \quad \text{(result is "atac")} \]

6) Special operators

Special functions can be used to obtain a reversed complementary sequence.

\[ \text{reversed} = \text{reverse(alpha)} \quad \text{(result is "catgcta")} \]
\[ \text{complementary} = \text{complement(alpha)} \quad \text{(result is "tagcatg")} \]
rev_compl = reverse(complement(alpha))  (result is "gtacgat")

7) Set operations

Sequences can be added to a set using "+" operator. Then other operators can be applied to the entire set, instead of a single sequence.

union1 := alpha + beta  (result is {"atcgtac", "CTAGCTATAC"})
union2 := alpha + gamma  (result is {"atcgtac", "atcgtacCTAGCTATAC"})

intersection := union1 & union2  (result is {"atcgtac"})
difference := union1 - union2  (result is {"CTAGCTATAC"})
xor := union1 ^ union2  (result is {"CTAGCTATAC", "atcgtacCTAGCTATAC"})

8) Backtranslations

This is used for dealing with exponential numbers of possibilities in the way a protein might be coded. It allows the programmer of a DNA library to “ambiguate” codon tables with or without usage data.

An example of how a programme using this function API might look like is shown in Fig. 1. The API allows the programmer of DNA libraries to utilise weights associated to codons that can be used to calculate usage probabilities. The software utilizes as a default codon table is E.coli K12 but the programmer can specify the same or another codon table.
Figure 1: A programme utilising backtranslation API functions

**API to the Reverse Parsing Planner**

To enable scripting, the planner web application provide a simple REST API:

- resource URL: [http://www.dnald.org/planner/api/run](http://www.dnald.org/planner/api/run)
- request type: POST
- parameter: "input" (required)
- string with library targets (one target per line)
- response format: HTML

The output contains the algorithm execution statistics, link to a graphical representation of the plan and a list of steps in each stage of the plan.

An example of how to call this API is provided below:
#!/usr/bin/env python

DNALD planner REST API usage example.

import urllib
import urllib2
import html5lib

APP_URL = "http://www.dnald.org/planner/"

# define target library
targets = ['g m o c", "o p m o c", "q b m o c", "a d m o c", "z y m o c"]

# send request and read the results
data = "\n".join(targets)
request = urllib2.Request(APP_URL + "api/run", urllib.urlencode(data))
html = urllib2.urlopen(request).read()

# parse the output HTML
document = html5lib.parse(html, "etree", "utf-8", False)

# download the plan graph image
img_url = document.find(".//img").get("src")
urllib.urlretrieve(APP_URL + img_url, "plan.png")

# print the plan
print "\n".join(e.text for e in document.iter() if e.tag in ["h4", "pre"])

3. Results

The software developed for WP1 is complex and as our end-users are biologists, we have tried to hide this complexity behind a sophisticated integrated development environment (see D1.5 report) for DNALD and a simple webserver for the reverse parser planning. Thus the APIs described here for DNALD and the webserver has been kept concise as to remain within the philosophy that the end-users should find it easy to use. An experience programmer, e.g. a computer scientist, willing to extend our code can of course do that by downloaded the sources for both DNALD IDE and the reverse planning algorithm. These sources are available from:

DNALD IDE: http://www.dnald.org/download/

Reverse Parser Planning Algorithm: www.dnald.org/planner/ACS_sb-2013-00161v_SI.zip

4. Conclusions
We provide a concise summary of the DNALD language API and the reverse parsing planning algorithm API. This deliverable’s report accompanies the report for D1.5 and should be read in conjunction also with the papers for the reverse planning algorithm that have been published in [1,2].

5. References


6. Abbreviations

List all abbreviations used in the document arranged alphabetically.

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNALD</td>
<td>DNA Library Design</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
</tbody>
</table>