Universidad Nacional de Córdoba

Trabajo especial de la Licenciatura en Ciencias de la Computación

Diseño de vacunas atenuadas con menor probabilidad de sufrir reversión a la virulencia

Autor:
Santiago Videla

Directora:
Dra. Laura Alonso Alemany

26 de noviembre de 2010

Resumen

Las denominadas vacunas vivas o atenuadas, han sido ampliamente utilizadas para prevenir enfermedades como la rubeola, la poliomielitis, el sarampión o la fiebre amarilla. Sin embargo, uno de los peligros potenciales del uso de este tipo de vacunas es la probabilidad de reversión a la virulencia, produciendo la enfermedad que intentan prevenir.

Se han desarrollado varios enfoques para reducir la probabilidad de reversión, tales como priorizar el uso de determinados codones o pares de codones, sin modificar la secuencia aminoacídica[6] o seleccionar polimerasas más fidedignas[18].

En este trabajo se propone un desarrollo complementario, que consiste en la implementación de un software para maximizar el número de mutaciones necesario para que la secuencia de Ácido ribonucleico (RNA) del virus vacunal, alcance secuencias semejantes a las patógenas o revertantes, manteniendo las propiedades que le otorgan la atenuación, poniendo especial hincapié en la conservación de la estructura secundaria de RNA.

Prefacio

Palabras alusivas

Índice general

Parte I

Preliminares

Capítulo 1

Introducción

> A foolish faith in authority is the worst enemy of truth.
>
> Albert Einstein

1.1. Para los ansiosos

El objetivo de este trabajo es el diseño y desarrollo de un software que sirva como soporte para el diseño de vacunas atenuadas. En este sentido, la propuesta es encontrar un conjunto de secuencias de RNA que conserven las propiedades que le otorgan la atenuación a la vacuna y que al mismo tiempo, tiendan a maximizar el numero de mutaciones necesarias para alcanzar secuencias semejantes a las patógenas o revertantes[1].

En los capítulos 2 y 3 se introducen los conceptos biológicos básicos y algunos de los problemas característicos de la bioinformática, profundizando en aquellos que están mas relacionados con este trabajo. Luego, en el capítulo 4 se describen la metodología clásica para el diseño de vacunas atenuadas, algunos antecedentes del diseño racional y finalmente, la solución propuesta. Por último, en la parte III se describen los detalles puramente técnicos y mas relevantes sobre el desarrollo del software.

1.2. Sobre las vacunas

Existen diferentes posiciones sobre la efectividad de las vacunas en la prevención de las enfermedades. Por un lado, la *versión oficial* sostiene que representan la principal herramienta para combatir enfermedades y epidemias. Pero al mismo tiempo, hay quienes aseguran que las vacunas son en muchos casos, las causantes de las enfermedades que intentan prevenir. Se cuestionan además, sus ingredientes tóxicos (aluminio, mercurio, cloroformo), en algunos casos cancerígenos o supuestamente relacionados con diferentes enfermedades como el autismo.

[1] Para el lector ajeno a la biología, qué producen la enfermedad que la vacuna debiera prevenir.

Aun así, su uso esta ampliamente aceptado en la mayoría de los países del mundo siendo, las campañas masivas de vacunación, una de las principales políticas publicas de salud.

No es el objetivo de este trabajo, profundizar en este tema ni tampoco llegar a una conclusión apresurada sobre la bondad de las vacunas, sino simplemente hacer mención a que es un debate abierto. El lector interesado, podrá consultar la bibliografía tanto a favor como en contra del uso masivo de vacunas según lo crea conveniente.

Un cacho de historia

El origen de las vacunas se remonta al año 1796 durante la epidemia del virus de la viruela en Europa. El medico rural, Edward Jenner, observo que las mujeres que ordeñaban las vacas, eventualmente contraían una especie de "viruela vacuna" por el contacto con las ubres y que luego la viruela común no les producía ningún efecto. Efectivamente, la viruela vacuna es una variante de la viruela común que no produce efectos de consideración en las personas y que dio origen a lo que hoy se conocen como vacunas vivas o atenuadas.

1.3. Motivación

Siendo brutalmente honesto, lo que motivó este trabajo fue la necesidad imperiosa de terminar la Licenciatura. Aunque a medida que me fui interiorizando en el tema, la problemática que se describe a continuación lo podría haber motivado perfectamente.

Dejando de lado la discusión planteada anteriormente, el objetivo básico de una vacuna es estimular el sistema inmune sin producir la enfermedad en cuestión. En este sentido, las vacunas atenuadas presentan algunas ventajas frente a las denominadas vacunas inactivas.

- Proveen inmunidad a largo plazo.
- Bajos costos.
- Pocas dosis son suficientes para adquirir inmunidad.
- Fáciles de aplicar.

Sin embargo, este tipo de vacunas también presentan una importante desventaja, como es la probabilidad de revertir a la virulencia y que da origen a este trabajo.

En sucesivas replicaciones, el virus atenuado puede acumular mutaciones en su secuencia de RNA que le devuelvan su carácter patogénico, produciendo la enfermedad que se deseaba prevenir. A diferencia de los virus Ácido desoxirribonucleico (DNA), los virus RNA poseen una alta frecuencia de mutaciones estimada en 0.1 a 10 mutaciones por genoma replicado[18].

Un caso paradigmático es el de la vacuna Sabin contra la poliomielitis, también conocida como Oral Polio Vaccine (OPV). Esta vacuna, desarrollada por Albert Sabin en 1957, es una vacuna atenuada administrada por vía oral para prevenir la poliomielitis.

A raíz de una campaña impulsada por la World Health Organization (WHO) en 1988 y utilizando la OPV, hacia finales de 2002, se había logrado interrumpir la transimisión endémica del poliovirus en 209, de los 216 países del mundo[2].

Sin embargo, la alta inestabilidad genética de esta vacuna dio lugar a un nuevo grupo de virus conocidos como poliovirus circulantes derivados de la vacuna (cVDPV) que presentan propiedades similares a las del poliovirus salvaje, incluyendo neurovirulencia[2] y responsables de una nueva enfermedad denominada parálisis poliomielítica asociada a la vacuna oral (VAPP).

Diferentes estudios muestran que VAPP ocurre a una tasa de aproximadamente 1 caso cada 750,000 a 1 millon de niños que reciben la primer dosis de OPV[2]. Mas aún, en Estados Unidos entre 1980 y 1999, el 95 % de los casos registrados de poliomielitis paralitica, fueron VAPP[12].

En Argentina, el ultimo caso registrado de VAPP se produjo en el año 2009, en la provincia de San Luis[7]. Esto derivo en un Alerta Epidemiológico acompañado de fuertes campañas de vacunación con la vacuna OPV.

1.4. Antecedentes

Ante estos datos, se reconoce que uno de los obstáculos para erradicar la poliomielitis, es la OPV en si misma[5]. Luego, surge la necesidad de buscar nuevas formas de diseñar vacunas atenuadas que estén libres de riesgos, o al menos tengan menor probabilidad de revertir a la virulencia.

Es a partir de esto y de los avances en la virología molecular, que empieza a tomar fuerza la idea de **racionalizar el diseño de vacunas atenuadas**, de forma tal de poder controlar y cuantificar la atenuación de las vacunas[13].

Algunos de los métodos propuestos y que analizaremos con mayor detalle mas adelante, son la deoptimización de codones[6] y la fidelidad en la replicación del virus atenuado[18].

1.5. Propuesta

En este trabajo, realizado con la colaboración de la Fundación para el Desarrollo de la Programación en Ácidos Nucleicos (FuDePAN)[3], se propone la implementación de un software, que hemos dado en llamar Combinatory Vaccine Optimizer (vac-o), para el diseño de secuencias de RNA que optimicen las vacunas atenuadas como un problema de **"optimización combinatoria basado en restricciones"**.

Este tipo de problemas consiste en asignar valores a un conjunto finito de variables que satisfagan determinadas condiciones o restricciones. Estas variables conforman las "componentes de la solución", y las combinaciones de los distintos valores que puede tomar cada componente forman las potenciales soluciones del problema. Luego, usando una función de evaluación sobre las soluciones, se debe encontrar una solución, o varias, que maximicen o minimicen dicha función.[11].

En nuestro caso, las restricciones serán propiedades sobre partes de la secuencia de RNA, como la conservación de la secuencia aminoacídica y la estructura

[2]Tendencia o capacidad de un microorganismo de afectar el sistema nervioso.

[3]`http://www.fudepan.org.ar`

secundaria. Las soluciones serán secuencias completas de RNA y la función de evaluación sobre estas soluciones, que deseamos maximizar, estará dada por el número de mutaciones necesario para alcanzar una secuencia revertante.

En este sentido, la principal innovación que presenta este trabajo, es la utilización de diferentes algoritmos para la predicción de estructura secundaria (directa e inversa), con el fin de determinar secuencias de RNA que conserven parte de la estructura secundaria del virus atenuado y en consecuencia, mantengan la atenuación y las propiedades como vacuna.

Para guiar el desarrollo del trabajo se tomo como caso testigo la vacuna OPV. Esto nos permitió establecer una serie de requerimientos básicos, que esperamos sean de utilidad para otras vacunas.

Desde el punto de vista de la implementación, se puso especial atención en utilizar diferentes principios y patrones de Object Oriented Programming (OOP) con el fin de lograr un software que sea altamente modular y que permita ser extendido en el futuro con nuevas funcionalidades.

El código fuente fue liberado bajo licencia GNU General Public License v3 (GPLv3) y puede ser accedido a través del repositorio Subversion (SVN)[4]

[4] `http://vac-o.googlecode.com`

Parte II

La Biología y las Ciencias de la Computación

Capítulo 2

Biología

> Essentially, all models are wrong, but some are useful.
>
> George E. P. Box

Siendo este un trabajo desde las Ciencias de la Computación, seria absurdo intentar abordar rigurosamente todos los conceptos biológicos involucrados en, por ejemplo, la replicación de un virus dentro de una célula. Al mismo tiempo, para poder comprender la complejidad del problema que este trabajo se propone resolver y las decisiones que se tomaron a la hora de implementar el software, es necesario contar con ciertas definiciones y conceptos biológicos elementales que veremos a continuación.

2.1. Lo esencial

Mas allá de la complejidad que existe dentro de un célula, sus componentes y funcionamiento, se destacan tres macro moléculas fundamentales compuestas por moléculas pequeñas:

- DNA y RNA - compuestas por nucleótidos.
- Proteína - compuesta por aminoácidos.

El dogma central

El dogma central de la biología molecular establece que la información fluye del DNA al RNA y luego a las proteínas.

$$\textbf{DNA} \longrightarrow \textbf{RNA} \longrightarrow \textbf{Proteínas}$$

Por supuesto, este es un modelo extremadamente simplificado al que le faltan componentes fundamentales para que todo funcione como corresponde. Podemos empezar diciendo que las flechas del modelo, implican transformaciones moleculares y por lo tanto, existen entidades encargadas de llevar adelante estas transformaciones.

La entidad que transforma el DNA en RNA se denomina RNA polimerasa, y el proceso de transformación se conoce como *transcripción*. Por otro lado, la entidad que transforma el RNA en proteínas, se denomina ribosoma y el proceso de esta tranformación es conocido como *traducción*.

Ácidos nucleicos

Ambos DNA y RNA son, como mencionamos anteriormente, macro moléculas compuestas por moléculas pequeñas. Estas moléculas pequeñas que los componen se denominan nucleótidos. Para definir en detalle a los nucleótidos, deberíamos profundizar en conceptos químicos que no son relevantes para este trabajo. Sin embargo, podemos decir que en el DNA aparecen 4 nucleótidos, Adenina (A), Guanina (G), Timina (T) y Citosina (C). Mientras que en el RNA la T es reemplazada por Uracilo (U). Estos nombres, se corresponden con la base nitrogenada que conforma cada nucleótido y que es lo que diferencia uno del otro. Luego, tanto el DNA como el RNA suelen describirse por su secuencia de nucleótidos.

El DNA se presenta como una doble cadena de nucleótidos, en la que las 2 hebras están unidas a través de las bases complementarias (reglas de Watson-Crick[1]), A con T y C con G.

En contraste con el DNA, el RNA se presenta como una cadena simple de nucleótidos. No obstante esto, el RNA también puede plegarse siguiendo las reglas de Watson-Crick (A-T, C-G) e inclusive usando pares menos estables como A-G. Esto da lugar a lo que se conoce como la estructura secundaria y que veremos con mayor detenimiento mas adelante.

El concepto de "estabilidad" que se acaba de mencionar esta relacionado con la cantidad de energía libre que se genera como consecuencia de la unión de dos bases de nucleótidos. Generalmente, predominan las uniones entre bases con menor energía libre (mas estables), aunque esto no siempre es así.

Proteínas

Al igual que lo ácidos nucleicos, las proteínas son macro moléculas compuestas por moléculas pequeñas, pero en este caso las moléculas que las componen, se denominan aminoácidos. También al igual que con los nucleótidos, dejaremos de lado la descripción química de los aminoácidos y nos centraremos en los aspectos que sean mas relevantes para este trabajo.

Existen 20 aminoácidos diferentes que suelen representarse por su nombre abreviado (usando 3 letras o 1 letra). Las proteínas son esencialmente secuencias de al menos 50 aminoácidos y son responsables de diferentes tareas fundamentales para el correcto funcionamiento de la célula. Sin ir mas lejos, el proceso de traducción, es decir la síntesis de proteínas, lo realizan los ribosomas, que están compuestos en parte por proteínas[2].

Del DNA a las proteínas

Como ya vimos mas arriba, el dogma central nos dice que la información en la célula fluye desde el DNA hasta convertirse en proteínas a través de transfor-

[1] Fracis H. Crick y James D. Watson recibieron en 1962 el Premio Nobel en Fisiología o Medicina.

[2] El lector se preguntara, ¿qué fue primero, el ribosoma o la proteína?.

maciones moleculares denominadas *transcripción* y *traducción*. También vimos, en muy pocas palabras, que tanto el DNA como el RNA están compuestos por nucleótidos mientras que las proteínas están compuestas por aminoácidos. No profundizaremos en los procesos de transcripción y traducción, pero si nos interesa ver como se traducen los nucleótidos a los aminoácidos.

Para esto necesitamos introducir el concepto de código genético. El código genético es, precisamente, el conjunto de normas o reglas con las cuales el material genético (secuencia de nucleótidos) se traduce en proteínas (secuencia de aminoácidos). El código define una relación entre secuencias de 3 nucleótidos, llamadas codones, y los aminoácidos. A cada codon le corresponde a lo sumo un aminoácido, pero como veremos en seguida, cada aminoácido puede estar relacionado con mas de un codon.

Como vimos anteriormente, en el RNA aparecen 4 nucleótidos. Luego, la cantidad de codones posibles, esta dada por $4^3 = 64$ codones posibles. Por otro lado, dijimos que existen 20 aminoácidos, lo que rápidamente nos hace notar que la relación definida por el código genético, no puede ser inyectiva. En criollo, la misma secuencia de aminoácidos puede ser codificada por distintas secuencias de nucleótidos.

La tabla de código genético indica qué codones se traducen en qué aminoácido. Hay aminoácidos que son representados por tan solo un codon, mientras otros, son representados hasta por 6 codones. Mas adelante, veremos que una de las estrategias para racionalizar el diseño de vacunas atenuadas, se basa en determinar que codones es conveniente usar para codificar una secuencia de aminoácidos determinada, y como esto afecta la atenuación del virus.

Además, debemos mencionar la existencia de 4 codones especiales. Estos codones, 1 denominado de *START* y los otros 3 denominados de *STOP*, sirven como indicadores para el proceso de transformación desde el RNA hasta las proteínas (traducción).

Esto ultimo es fundamental para delimitar 2 tipos de regiones sobre una secuencia de RNA:

- Regiones traducidas: se las denomina Open Reading Frame (ORF) y son las partes de la secuencia que efectivamente se traducen en proteínas. Puede haber uno o varios ORF, pero cada uno debe empezar con un codon de *START* y terminar con uno de *STOP*.

- Regiones no traducidas: Se las denomina Untranslated Region (UTR) y se encuentran a izquierda y derecha de un ORF. Se suele hablar de un 5'-UTR y un 3'-UTR, para hacer referencia al extremo izquierdo y derecho respectivamente[3].

Por ultimo, haremos referencia al Internal Ribosomal Entry Site (IRES). Un IRES es una secuencia de nucleótidos que permite el inicio del proceso de traducción y suele encontrarse en el 5'-UTR de los virus RNA. Esta secuencia y la estructura secundaria que formen, determina en gran medida la síntesis de proteínas del virus.

[3]La notación de 5' y 3' viene de la notación utilizada en química para numerar los carbonos de un compuesto orgánico.

2.2. La estructura secundaria de RNA

Como ya mencionamos, el plegamiento de una secuencia de RNA entre sus bases complementarias, determina lo que se denomina *estructura secundaria de RNA*. Conocer la estructura secundaria es fundamental para comprender el funcionamiento de los distintos tipos de RNA y de la célula en general. Existen diferentes tipos de RNA pero se distinguen 3 tipos principales:

- messenger RNA (mRNA).
- ribosomal RNA (rRNA).
- transfer RNA (tRNA).

Cada uno de estos tipos de RNA, desarrollan diferentes funciones en el modelo del dogma central con el fin de lograr las transformaciones moleculares de transcripción y traducción. Por lo tanto, los plegamientos o estructuras secundarias que forman, determinan en gran medida la actividad celular.

La existencia del termino *estructura secundaria*, nos hace suponer que existe también una *estructura primaria*. De hecho, ya vimos la estructura primaria de RNA cuando dijimos que el RNA, se suele representar como una secuencia de nucleótidos. Efectivamente, a esta secuencia se la denomina estructura primaria. También existe la estructura terciaria de RNA, pero la dejaremos de lado por no ser relevante en este trabajo.

Definición 1. Una estructura primaria de RNA S, es una secuencia de RNA de longitud n, $A = a_1a_2a_3 \dots a_n$ con $a_i \in \{A, U, G, C\}$

Definición 2. Dada una estructura primaria o secuencia de RNA de longitud n, la estructura secundaria, es un conjunto S de pares (i, j) con $1 \leq i < j \leq n$ tal que, $\forall (i, j), (i', j') \in S$ se satisfacen

- $j - i > 3$
- $i = i' \Leftrightarrow j = j'$
- $i < i' \Rightarrow i < i' < j' < j \vee i < j < i' < j'$

Nótese que cada base de la secuencia puede estar a lo sumo "apareada" o "unida" con una sola base, o eventualmente con ninguna. Además, esta definición supone la ausencia de *pseudo-knots*. Un *pseudo-knot* son 2 pares de bases superpuestos, es decir, (i, j) y (i', j') con $i < i' < j < j'$. Esta suposición se debe principalmente a que de otra manera, los algoritmos que veremos mas adelante para la predicción de estructura secundaria, pasan de tener complejidad polinomial a ser NP-completos[14]. Sin embargo, esta suposición esta justificada en que la ocurrencia de *pseudo-knots* es poco frecuente en la naturaleza y en que además pueden ser considerados en análisis posteriores[20].

La estructura secundaria, el IRES y la atenuación de un virus

La importancia de la estructura secundaria en este trabajo radica, fundamentalmente, en su participación en la síntesis de proteínas del virus y en consecuencia, en su atenuación. Para los virus RNA, se puede decir que la estructura secundaria del IRES, determina la "afinidad" o "atracción" con los ribosomas.

Luego, esto incide fuertemente en el proceso de traducción de las proteínas que causan la enfermedad, o que están involucradas en el proceso de replicación del virus.

Una estructura secundaria con menor afinidad (que la estructura secundaria del virus), le permitiría al sistema inmune generar los anticuerpos necesarios para protegerse del virus antes de que se produzca la enfermedad.

2.3. Los virus RNA

Un virus RNA es esencialmente, aquel que tiene el RNA como su material genético. Como ya se menciono en la sección 1.3, una de las características de este tipo de virus, es su alta frecuencia de mutaciones. Esto se debe, principalmente, a la alta tasa de error en su RNA polimerasa[4] (involucrada en el proceso de replicación del virus), estimada entre 1×10^{-3} a 1×10^{-5} errores por nucleótido por ciclo de replicación[18]. Debido a que los virus de RNA suelen tener menos de 10,000 nucleótidos, esto se traduce en 0.1 a 10 mutaciones por genoma replicado.

Si retomamos lo dicho en la sección 2.2 sobre la "afinidad" entre la estructura secundaria y los ribosomas, y cómo esto impacta en la atenuación del virus, se puede ver que esta alta frecuencia de mutaciones, conduce a la posibilidad de que en sucesivas replicaciones, el virus sufra mutaciones que le devuelvan su estructura secundaria original (o similar) y en consecuencia, pierda su atenuación.

Poliovirus

El poliovirus es un virus RNA de aproximadamente 7,500 nucleótidos de longitud, miembro del género *Enterovirus* de la familia *Picornaviridae* y causante de la poliomielitis. El poliovirus puede atacar el sistema nervioso y destruir las células nerviosas encargadas del control de los músculos. Como consecuencia, los músculos afectados dejan de cumplir su función y se puede llegar a una parálisis irreversible. En casos severos, la enfermedad puede conducir a la muerte.

La vacuna OPV

Ya presentamos en la sección 1.3 a la vacuna OPV contra la poliomielitis y sus principales complicaciones. Con los conceptos vistos hasta el momento, estamos en condiciones de profundizar sobre el porque de estas complicaciones.

Se han identificado 3 serotipos[5] de poliovirus: Poliovirus type 1 (PV1), Poliovirus type 2 (PV2) y Poliovirus type 3 (PV3). Los 3 serotipos son extremadamente virulentos y producen los mismos síntomas de la enfermedad. Para cada uno de estos serotipos, se desarrolló su correspondiente virus atenuado Sabin 1, Sabin 2 y Sabin 3 respectivamente que luego fueron combinados en la vacuna OPV.

Cada uno de estos virus atenuados presentan diferentes niveles de riesgo o probabilidad de revertir a la virulencia. El 90 % de los casos registrados de VAPP son causados por Sabin 2 o Sabin 3, mientras que tan solo el 10 % restante, es causado por Sabin 1[16]. Esto se atribuye principalmente, a la cantidad de bases de nucleótidos en que difieren, cada serotipo con respecto a su correspondiente

[4] A diferencia de la DNA polimerasa que posee la capacidad de detectar y corregir errores.

[5] A los fines prácticos y relevantes en este trabajo, formas en que se presenta un determinado virus.

virus atenuado. Mientras que la atenuación en Sabin 2 y Sabin 3, esta dada por el impacto de 2 o a lo sumo 3 mutaciones, la atenuación en Sabin 1 es mas compleja y esto justificaría su menor probabilidad de revertir a la virulencia[16].

Mas allá de estas diferencias, los tres virus atenuados, Sabin 1, Sabin 2 y Sabin 3, presentan mutaciones en la 5'-UTR que contribuyen a la atenuación. Además, de los aproximadamente 740 nucleótidos de la 5'-UTR, los primeros 620 se conservan en todos los poliovirus, mientras que las 100 bases que preceden el ORF son las mas divergentes. Todo esto sugiere que la 5'-UTR tiene un rol muy importante en el ciclo de vida de los poliovirus[16].

Capítulo 3

Bioinformática y Biología computacional

> I can't be as confident about computer science as I can about biology. Biology easily has 500 years of exciting problems to work on. It's at that level.
>
> Donald Knuth

Por tratarse de una disciplina relativamente nueva, nos permitimos una breve introducción, definición y repaso de los principales problemas abordados por la misma profundizando en aquellos problemas que están directamente relacionados con este trabajo. En particular, la predicción de estructura secundaria.

3.1. Bio*

La biología depende en gran parte de la química para poder avanzar y esto dio lugar a lo que se conoce como *bioquímica*. Análogamente, la necesidad de explicar fenómenos biológicos a nivel atómico, dio lugar a la *biofísica*. La *biomatemática* por su parte, se enfoca en el modelado de procesos biológicos utilizando técnicas matemáticas que permitan simular y predecir su comportamiento. La enorme cantidad de datos recopilados por los biólogos y la necesidad de herramientas para interpretarlos, dio origen a lo que hoy conocemos como *bioinformática*.

La bioinformática fue precedida por lo que se llamó, *biología computacional*. Aunque no hay una definición precisa para ninguno de los dos términos, la biología computacional se caracterizo por enfocarse en los aspectos teóricos/formales de las Ciencias de la Computación, mientras que la bioinformática supo estar mas relacionada con el procesamiento de grandes volúmenes de datos, usualmente, almacenados en la Internet. Actualmente, se suele hacer referencia a ambos términos de manera indiferente.

Problemas clásicos

A continuación presentamos algunos de los problemas o temas de investigación abordados por la bioinformática.

- **Análisis de secuencias de DNA o RNA**. TBD
- **Diseño de secuencias de DNA o RNA**. TBD
- **Predicción de interacción entre proteínas**. TBD

3.2. Predicción de estructura secundaria

Uno de los problemas que omitimos mencionar anteriormente y que describiremos con mayor detalle, es el que se conoce como "predicción de estructura secundaria".

Este problema consiste en, dada una estructura primaria o secuencia de RNA, determinar la estructura secundaria correspondiente. Además, diferentes secuencias de RNA pueden tener la misma estructura secundaria, por lo que también nos interesa determinar para una estructura secundaria dada, las secuencias de RNA que conservan esa estructura.

En ingles, se suele denominar a estos dos problemas "folding" e "inverse folding" respectivamente. A continuación describimos brevemente las diferentes aproximaciones a cada uno de ellos.

Por lo mencionado en la secciones 2.2 y 2.3, predecir la estructura secundaria de una secuencia de RNA y conocer las posibles secuencias que conservan una estructura secundaria determinada, es de gran importancia en este trabajo.

3.2.1. Predicción directa (folding)

Existen esencialmente 2 tipos de algoritmos para determinar o predecir la estructura secundaria de una secuencia de RNA.

- **Predicción por minimal free energy (mfe)**. Propuesto e implementado por Michael Zuker en 1981[21], utiliza programación dinámica para encontrar la estructura secundaria que minimiza la energía libre.
- **Predicción comparativa**. Utiliza diferentes métodos para comparar secuencias y estructuras con el fin de obtener una estructura por "consenso"[9].

Si bien la "predicción comparativa" presenta un incremento en la fidelidad de los resultados obtenidos con respecto a la "predicción por mfe" [9], este tipo de algoritmos requieren la existencia de un conjunto de secuencias relacionadas entre si (homologas) y esto no siempre es posible. En particular para este trabajo nos interesa poder realizar predicciones de la estructura secundaria a partir de una sola secuencia, por lo que la "predicción comparativa" fue descartada.

Entre las implementaciones de la "predicción por mfe", se destacan **RNAfold**[10] y **Mfold**[21]. Ambas implementan el algoritmo propuesto por Michael Zuker con complejidad $\mathcal{O}(N^3)$ donde N es la longitud de la secuencia, aunque **RNAfold** se presenta como una versión mejorada y mas eficiente en la práctica. Como ya mencionamos en la sección 2.2, para lograr esta complejidad polinomial, es

necesario suponer la ausencia de *pseudo-knots*. De lo contrario, esta demostrado que el problema es NP-completo[14].

A continuación, se puede ver un ejemplo de una predicción realizada con **RNAfold**.

```
sancho@mulata:~$ RNAfold

Input string (upper or lower case); @ to quit
....,....1....,....2....,....3....,....4....,....5....,....6....,....7....,....8
AAAGGCAACGGCCAU
length = 15
AAAGGCAACGGCCAU
...(((....)))..
 minimum free energy =  -4.40 kcal/mol
```

El resultado obtenido es precisamente, la energía libre y la estructura secundaria representada con paréntesis y puntos. Donde los pares de paréntesis indican las bases "apareadas" o "unidas" y los puntos, las bases libres.

3.2.2. Predicción inversa (inverse folding)

Las principales implementaciones que abordan este problema, lo plantean como un Constraint Satisfaction Problem (CSP) y utilizan variantes de búsqueda local estocástica para resolverlo. La función objetivo y que se desea minimizar, es la distancia estructural[1] entre la estructura mfe de la secuencia solución[2] y la estructura secundaria dada.

Si bien la complejidad de este problema no esta determinada, a diferencia de otros CSP, la evaluación de las posibles soluciones es muy costosa ya que implica la "predicción mfe" sobre cada secuencia candidata ($\mathcal{O}(N^3)$). Luego, se deben utilizar diferentes técnicas que tiendan a minimizar el numero de predicciones realizadas sobre la secuencia completa.

En general, las diferentes implementaciones tienen como parámetros de entrada, la estructura secundaria y opcionalmente, una secuencia de RNA incompleta (algunas bases indefinidas, generalmente representadas con la letra N). Se distinguen dos pasos o etapas principales, que marcan las diferencias entre una y otra implementación:

1. **Inicialización:** determinar una secuencia inicial completando la secuencia dada como parámetro. En el caso de no recibir ninguna secuencia como parámetro, se asume una secuencias con todas las bases indefinidas.

2. **Búsqueda local:** mejorar iterativamente la secuencia inicial hasta alcanzar una secuencia solución. Es decir, realizar cambios en la secuencia que tiendan a minimizar la distancia estructural con la estructura buscada.

Entre estas implementaciones, se destacan **RNAinverse**[10], **INFO-RNA**[4] y **RNA-SSD**[1]. Las últimas dos, se presentan como mejoras a la primera proponiendo diferentes formas de generar la secuencia inicial e implementando algoritmos de búsqueda local estocástica mas complejos.

[1] Existen diferentes métodos y algoritmos para calcular la distancia entre estructuras cuya descripción exceden este trabajo.

[2] En este contexto, una secuencia solución es aquella cuya predicción de estructura secundaria da como resultado la estructura buscada.

Notar que las tres implementaciones dependen de la generación de números *pseudo aleatorios* y ya que la "semilla" utilizada es variable (salvo **RNA-SSD** que permite fijar la semilla), se debe tener en cuenta el no determinismo.

A continuación, mostramos un ejemplo de una predicción inversa usando **RNAinverse** sobre la estructura obtenida anteriormente con **RNAfold**.

```
sancho@mulata:~$ RNAinverse

Input structure & start string (lower case letters for const positions)
    @ to quit, and 0 for random start string
....,....1....,....2....,....3....,....4....,....5....,....6....,....7....,....8
...(((....))).
aaaNNNNNNNNNNu
length = 15
aaaUAGUCAGCUACu    3 0
```

Nótese que la secuencia obtenida, conserva las bases que ya estaban definidas en la secuencia incompleta que se dio como parámetro y que además, es distinta a la secuencia sobre la que se hizo la predicción directa anteriormente.

Capítulo 4

Diseño de vacunas atenuadas

> Science is always wrong. It never solves a problem without creating ten more.
>
> George Bernard Shaw

Una vacuna atenuada es aquella que es creada reduciendo la virulencia de un patógeno pero aun así, manteniéndolo viable ("vivo"). Antes de adentrarnos en los detalles de lo que plantea este trabajo como metodología para racionalizar el diseño de vacunas atenuadas, veremos brevemente algunos antecedentes y cual fue, y sigue siendo, la metodología clásica para la producción de este tipo de vacunas.

4.1. Diseño clásico

La metodología para la producción de vacunas atenuadas ha sido históricamente, el pasaje del virus a través de cultivos de células distintas a las células huésped. De esta manera, el virus tiende a "evolucionar" para adaptarse al nuevo huésped y ser capaz de reproducirse.

Este concepto de "evolución", se traduce en alguna cantidad de mutaciones sobre la secuencia de nucleótidos del virus y que se espera, reduzcan su capacidad de reproducirse en el huésped original. Luego, son precisamente estas mutaciones las que le confieren la atenuación y dan lugar a la vacuna atenuada.

La principal desventaja en este proceso, es que las mutaciones que se producen son totalmente impredecibles y aun cuando estas mutaciones derivan en un virus atenuado, este podría revertir a la virulencia dependiendo de la naturaleza de las mutaciones que generan la atenuación[3]. Además, como vimos en la sección 2.3, la alta frecuencia de mutaciones que poseen los virus RNA aumenta la probabilidad de reversión a la virulencia. De hecho, esto es lo que ocurre con muchas vacunas atenuadas y en particular, con la OPV.

A pesar de este peligro de reversión a la virulencia, esta sigue siendo la principal metodología para la producción de vacunas atenuadas. Esto se debe,

fundamentalmente, a la falta de conocimiento sobre el significado o incidencia de las mutaciones en la atenuación de un determinado virus. Sin embargo, los recientes avances en la virología molecular, han permitido explorar nuevas técnicas que permitan controlar la replicación de un virus o su virulencia y esto abrió la puerta a lo que se denomina "diseño racional de vacunas atenuadas"[13].

4.2. Diseño racional

La idea central en esta nueva metodología, en la que se enmarca en este trabajo, es explotar el conocimiento que se ha producido en los últimos años acerca de la biología molecular de determinados virus. Pudiendo controlar la replicación de un virus o su virulencia, sería posible diseñar vacunas atenuadas "seguras" evitando la impredecibilidad de las atenuaciones empíricas obtenidas mediante el diseño clásico.

4.2.1. Antecedentes

Existen diferentes aproximaciones al diseño racional de vacunas atenuadas[13]. En general, todas estas aproximaciones se encuentran en fase experimental y todavía no se han aplicado en producción. A continuación hacemos mención a tan solo dos de ellas, fundamentalmente para marcar la diferencia con el diseño clásico que presentamos anteriormente.

Fidelidad en la replicación[18]

Como ya vimos en la sección 2.3, la alta frecuencia de mutaciones en los virus RNA se debe a la alta tasa de error en su RNA polimerasa. Luego, modificando la RNA polimerasa de tal manera que se reduzca su tasa de error, se obtendría un virus atenuado mas estable y con menor probabilidad de revertir a la virulencia en las sucesivas replicaciones.

La principal desventaja de esta aproximación radica en que las posibles variantes sobre la RNA polimerasa deben ser determinadas y evaluadas experimentalmente para cada virus en particular.

(De-)Optimización de codones[17, 6]

Si recordamos lo dicho en la sección 2.1 sobre el código genético, vimos que cada aminoácido puede ser codificado hasta por 6 codones distintos. Es decir, distintas secuencias de nucleótidos resultan equivalentes en términos de los aminoácidos que codifican. Concretamente, una proteína de 300 aminoácidos puede ser codificada por aproximadamente, 10^{151} secuencias de nucleótidos.

Sin embargo, experimentalmente se pudo comprobar que algunos codones son mas frecuentes que otros ("codon bias"). Similarmente, pero de manera independiente, se comprobó que determinados pares de codones son mas frecuentes que otros ("codon pair bias"). Aunque todavía no esta claro a que se debe esta "parcialidad" en el uso de codones o pares de codones, se supone que afectaría el proceso de síntesis de proteínas (traducción).

Lo que se propone con esta aproximación, es determinar las secuencias de nucleótidos que conserven la secuencia aminoacídica del virus pero que al mismo tiempo, tiendan a usar codones y pares de codones menos frecuentes. De

esta manera, la atenuación del virus se obtendría debilitando su capacidad de traducción y replicación.

Entre las ventajas que presenta esta metodología, se destaca por un lado que la atenuación es el resultado de un análisis sistemático y por lo tanto, aplicable a diferentes virus de manera automática. Por otro lado, la alta cantidad de cambios que se realizan sobre el virus original sugieren una menor probabilidad de revertir a la virulencia.

4.2.2. Propuesta

La propuesta que realizamos en este trabajo tiene algunos puntos en común con la "(de-)optimización de codones" que presentamos anteriormente. En particular, ambas aproximaciones comparten la idea de sistematizar el diseño de forma tal que pueda ser usado para diferentes virus. Esto implica fundamentalmente, plasmar la metodología en la implementación de un software que a partir de una serie de datos provistos por el usuario, devuelva como resultado una o varias secuencias de nucleótidos que representen posibles atenuaciones del virus.

De una forma mas abstracta, podemos pensar en un software para el diseño de secuencias de nucleótidos basado en restricciones. Fundamentalmente, lo que se busca es "generar" secuencias que satisfagan determinadas propiedades o restricciones. En particular, aquellas que permitan considerar a las secuencias generadas como virus atenuados.

En este sentido, se podría trazar una analogía con los algoritmos para la predicción inversa de estructura secundaria (*inverse folding*). En esencia, estos algoritmos "diseñan" secuencias de RNA que tengan como estructura secundaria mfe, la estructura dada por el usuario. De hecho, como mencionamos en la sección 3.2.2, el problema se plantea computacionalmente como un CSP.

La principal innovación de esta propuesta es que se pone el foco en la importancia del IRES y su estructura secundaria, en la traducción y replicación de diferentes virus RNA. En especial y tal como vimos en la sección 2.3, del poliovirus.

Lo que nos proponemos en este trabajo, es realizar un análisis sistemático de las posibles variantes al IRES de los virus atenuados Sabin (y eventualmente cualquier otro) que conserven la estructura secundaria y en consecuencia, la atenuación del virus. Luego, maximizando la cantidad de mutaciones necesarias para revertir a secuencias semejantes a las patógenas o revertantes, estaríamos reduciendo la probabilidad de que el virus atenuado sufra reversión a la virulencia.

En resumen, podríamos plantear el problema de la siguiente manera:

- **Entrada:** Genoma del virus atenuado, genoma del patógeno o revertantes y un conjunto de restricciones. Fundamentalmente, la conservación de la estructura secundaria del IRES del virus atenuado.
- **Objetivo:** Satisfaciendo las restricciones impuestas, maximizar la cantidad de mutaciones necesarias para convertir el virus atenuado en alguna secuencia patógena o revertante.
- **Salida:** Una o varias secuencias, candidatas a "mejorar" el virus atenuado.

Como mencionamos en la sección 1.5 esto puede ser visto como un problema de **"optimización combinatoria basado en restricciones"**. Pero para hacerlo, primero se deben identificar algunos elementos fundamentales que permitan definir el problema.

Sea N la longitud de la secuencia de RNA del virus atenuado, y para $k \in \mathbb{N}$ sea $\mathcal{S}_k$ el conjunto de secuencias de RNA de longitud k. Entonces definimos:

- **Espacio de soluciones:** $\mathcal{S}_N$
- **Componentes variables de una solución:** $s_1, s_2, \ldots, s_n$ tal que $s_i \in \mathcal{S}_{N_i}$ con $1 \leq i \leq n$ y $0 < N_i \leq N$
- **Restricciones sobre las componentes:** Conservación de la estructura secundaria o de la secuencia aminoacídica con respecto al virus atenuado, entre otras.
- **Función "objetivo" o de evaluación:** $f : \mathcal{S}_N \to \mathbb{R}$ tal que $f(s)$ calcula la distancia entre la solución s y alguna secuencia patógena o revertante.

Lo primero que podemos mencionar es que para cualquier virus RNA, recorrer el espacio de soluciones de manera exhaustiva es inviable ya que por ejemplo, para el caso del poliovirus, $N \simeq 7,500$. Es decir que existen aproximadamente $4^{7,500}$ posibles secuencias de RNA.

Por otro lado, la posibilidad de evaluar los posibles valores para cada componente s_i de manera exhaustiva, dependerá fundamentalmente del tipo de restricción impuesta sobre esa componente y de la longitud N_i. En particular para los virus atenuados Sabin y la conservación de la estructura secundaria del IRES ($N_i \simeq 400$), la evaluación exhaustiva sería inviable.

Por ultimo, nos queda por definir la función de evaluación $f : \mathcal{S}_N \to \mathbb{R}$. En principio se podría pensar que la imagen de la función, sea $\mathbb{N}$ en lugar de $\mathbb{R}$. De hecho, este es el caso si la función calcula la distancia de Hamming estándar. Esto es, sumar 1 por cada una de las bases en las que difiere la secuencia solución de la secuencia patógena. Pero de esta manera se estaría suponiendo que la probabilidad de mutación entre las bases es uniforme y esto no es así necesariamente.

Luego, determinando empíricamente la probabilidad de mutación de cada base hacia cualquiera de las otras tres, se podría incluir esta información en la función de evaluación. Eventualmente, también se podrían considerar cálculos de distancia mas complejos como es el caso de la "Distancia de Levenshtein".

Con el problema planteado de esta manera, se pueden utilizar diferentes algoritmos de búsqueda local para que partiendo de una secuencia de RNA inicial, en este caso el virus atenuado, recorrer el espacio de búsqueda $\mathcal{S}_N$ teniendo como objetivo, maximizar la función de evaluación f.

Parte III

Desarrollo del Software

Capítulo 5

Proceso de desarrollo

> How does a project get to be a year late?... One day at a time.
>
> Fred Brooks

A esta altura ya hemos presentado los principales conceptos biológicos involucrados en este trabajo, el diseño de vacunas atenuadas y la formalización del problema que nos propusimos resolver. En los capítulos restantes veremos los aspectos técnicos mas relevantes sobre el desarrollo del software.

5.1. Modelo de desarrollo

Para llevar adelante el desarrollo del software, se opto por el clásico modelo de cascada. Fundamentalmente debido a su simplicidad y a que los requerimientos con que debía cumplir el software estaban bien definidos desde el inicio. No solo a nivel funcional, sino también en cuanto a principios del diseño orientado a objetos.

5.1.1. Etapas de la cascada

Las etapas que se llevaron a cabo durante el desarrollo de este trabajo y que veremos con mayor detenimiento en los siguientes capítulos, fueron las siguientes:

1. Especificación de requerimientos.
2. Diseño.
3. Implementación.
4. Verificación.

Quedaron fuera del alcance, las etapas “Instalación” y por supuesto, “Mantenimiento”.

5.2. Ecosistema

El "ecosistema" de herramientas que se utilizaron para llevar adelante el proceso de desarrollo son las siguientes:

- **Lenguaje de programación:** El software se implemento en **C++**[1].
- **Lenguaje de diseño:** El diseño del software se hizo utilizando **UML**[2].
- **Control de versiones:** Se utilizo el sistema de control de versiones SVN[3] y puede ser consultado en `http://vac-o.googlecode.com`.
- **Sistema de "construcción":** Para automatizar el proceso de compilación del código fuente se utilizo **CMake**[4].
- **Automatización de pruebas:** Para realizar la verificación del software, se opto por implementar pruebas unitarias utilizando **google-test**[5] y **google-mock**[6].
- **Análisis estático:** Se utilizaron las herramientas **astyle**[7] y **cppcheck**[8].
- **Análisis dinámico:** Se utilizaron las herramientas **valgrind**[9] y **gcov**[10] para verificar la ausencia de "memory leaks" y la cobertura de las pruebas.

[1] `http://cplusplus.com`
[2] `http://www.uml.org`
[3] `http://subversion.apache.org`
[4] `http://www.cmake.org`
[5] `http://googletest.googlecode.com`
[6] `http://googlemock.googlecode.com`
[7] `http://astyle.sourceforge.net`
[8] `http://sourceforge.net/apps/mediawiki/cppcheck`
[9] `http://valgrind.org`
[10] `http://gcc.gnu.org/onlinedocs/gcc/Gcov.html`

Capítulo 6

Requerimientos

> If you think it's simple, then you have misunderstood the problem.
>
> Bjarne Stroustrup

Los requerimientos del software fueron provistos por miembros de FuDePAN y quedaron documentados en la "Especificación de Requerimientos de Software" que se anexa a este trabajo. A continuación haremos mención a los requerimientos mas relevantes y que nos permitan introducir los principales aspectos del software.

6.1. Descripción general

Retomando lo dicho en la sección 4.2.2, podríamos resumir los requerimientos funcionales de vac-o de la siguiente manera:

- **Entrada:** Genoma del virus atenuado, genoma del patógeno o revertantes y un conjunto de restricciones sobre partes de la secuencias del virus atenuado.
- **Objetivo:** Satisfaciendo las restricciones impuestas, maximizar la cantidad de mutaciones necesarias para convertir el virus atenuado en alguna secuencia patógena o revertante.
- **Salida:** Una o varias secuencias, candidatas a "mejorar" el virus atenuado.

Desde el punto de vista del diseño, el sistema debía ser altamente modular, de forma que sea apto para encontrar secuencias genómicas para diferentes tipos de organismos. En este sentido, se debieron seguir los principios de diseño conocidos por el acrónimo **SOLID**[15].

6.2. Extensiones

Uno de los principales requerimientos con que debía cumplir el software es que sea versátil. Es decir, que permita ser configurado y extendido de una forma simple y sin modificar sus componentes centrales.

En este sentido, se propuso desde un comienzo un software que funcionaría en base a la información provista por una extensión o *plugin*. Básicamente, la responsabilidad de una extensión sería la de proveer a vac-o la información relevante para cada virus atenuado que se desee optimizar. En particular, el genoma del virus atenuado, el genoma del patógeno y el conjunto de restricciones que se deben satisfacer durante la búsqueda. Las extensiones también serían responsables de evaluar las secuencias candidatas y de determinar la estrategia de búsqueda.

6.3. Regiones combinatorias

Como ya mencionamos anteriormente, el problema se puede ver como un problema de "optimización combinatoria basado en restricciones". Luego, un ingrediente que no podemos dejar de mencionar son precisamente, las restricciones o "regiones combinatorias". Fundamentalmente, representan las componentes variables de cada solución y definen en gran medida el espacio de búsqueda.

En este contexto nos referimos a las "variantes" de una región como los posibles valores (secuencias de RNA) que satisfacen la restricción impuesta. Luego podemos definir a una región combinatoria de la siguiente manera.

Definición 3. Dada una secuencia de RNA S de longitud n, una región combinatoria sobre S sera una 4-upla $(inicio, fin, tipo, eval)$ tal que:

- $0 \leq inicio < fin \leq n$
- *tipo* determina la restricción sobre la región y por ende, queda definido un conjunto $\mathcal{V}$ de posibles variantes a la región.
- $eval : \mathcal{V} \rightarrow (0, 1)$ es una función de evaluación local a la región que determina la bondad de una variante determinada.

Si suponemos $N \geq 1$ y $R_1, \ldots, R_N$ regiones combinatorias. Entonces tendremos que el número de posibles secuencias que vac-o "debería" evaluar, sera igual al producto cartesiano $\mathcal{V}_1 \times \ldots \times \mathcal{V}_N$ con $\mathcal{V}_1, \ldots, \mathcal{V}_N$ variantes de cada región combinatoria respectivamente.

Inicialmente, se contemplaron dos tipos de regiones combinatorias aunque se debe tener en cuenta la posibilidad de que en el futuro se agreguen mas. Estos tipos de regiones combinatorias o restricciones son:

- Conservación de la estructura secundaria de RNA.
- Conservación del código genético.

Además, para el caso de la "conservación de estructura secundaria de RNA" se debía contar con la posibilidad de utilizar diferentes algoritmos de predicción directa e inversa de manera "transparente".

6.4. Estrategias de búsqueda

Otro de los aspectos "configurables" del software debía ser la estrategia de búsqueda utilizada. En este sentido, la extensión es responsable de proveer

al sistema la "forma" de recorrer el espacio de búsqueda y al mismo tiempo determinar cuando se debe terminar el recorrido.

Vale aclarar que todos los algoritmos de búsqueda local son "incompletos" en el sentido de que no recorren exhaustivamente el espacio de búsqueda como si lo hacen los algoritmos mas tradicionales o sistemáticos. Luego, es necesario contar con un criterio de terminación que suele estar relacionado con la cantidad de iteraciones realizadas, la "calidad" de las soluciones encontradas o la cantidad de iteraciones sin que se produzcan mejores soluciones.

6.5. Control de calidad

Cada secuencia encontrada por vac-o durante el recorrido del espacio de búsqueda, debe ser sometida a un "control de calidad". Básicamente se pretende simular las mutaciones acumuladas que se producen en las sucesivas replicaciones del virus en la naturaleza y sobre cada secuencia mutante, verificar determinadas propiedades que brinden mayor seguridad sobre la atenuación del virus.

Análogamente a las regiones combinatorias, definimos las regiones de validación de la siguiente manera.

Definición 4. Dada una secuencia de RNA S de longitud n, una región de validación sobre S sera una 4-upla $(inicio, fin, prueba, criterio)$ tal que:

- $0 \leq inicio < fin \leq n$
- *prueba* determina la forma en que se producen las mutaciones.
- *criterio* determina la propiedad que deben cumplir todas las secuencias mutantes generadas.

Se puede pensar al control de calidad como la generación de un árbol de secuencias. La raíz del árbol sera la secuencia candidata y los nodos del árbol serán las secuencias mutantes generadas. Luego, diremos que una secuencia candidata pasa el control de calidad sobre una región de validación, cuando sea posible generar todos los nodos del árbol para una profundidad dada. Por ultimo, solo serán consideradas aquellas secuencias candidatas que pasen el control de calidad sobre todas las regiones de validación.

Para las formas de producir mutaciones acumuladas se contemplaron dos alternativas:

- Mutaciones sistemáticas.
- Mutaciones aleatorias.

Por otro lado, las propiedades que se contemplaron como criterio de verificación son:

- Similitud estructural.
- Disimilitud estructural.

De todas formas, se debía contemplar la posibilidad de que en el futuro se agreguen otras formas de generar mutaciones y otros criterios de verificación.

6.6. Ranking de secuencias candidatas

Finalmente, nos queda por describir el ranking de secuencias candidatas y que representa fundamentalmente el resultado de la ejecución de vac-o.

Esto es simplemente un listado de secuencias de RNA, ordenadas según la evaluación que hiciera la extensión. Básicamente, las secuencias mejor evaluadas serán aquellas que estén a mayor distancia (para alguna definición de distancia) de la secuencia patógena o revertante y por ende, tendrán menor probabilidad de sufrir reversión a la virulencia.

Capítulo 7

Diseño

> The effective exploitation of his powers of abstraction must be regarded as one of the most vital activities of a competent programmer.
>
> Edsger W. Dijkstra

Al igual que con los requerimientos del software, la descripción detallada del diseño quedo documentada en la "Especificación de Diseño de Software" y que se agrega como anexo. Análogamente al capitulo anterior, nos limitaremos a mencionar solo los aspectos mas relevantes del diseño, dejando como lectura adicional el documento anexo.

7.1. Metodología

La metodología adoptada para realizar el análisis y descripción del diseño se denomina Responsibility-Driven Design (RDD)[19]. Esta técnica del diseño orientado a objetos, se enfoca en qué responsabilidades deben ser cubiertas por el sistema y en cuáles serán los objetos responsables de llevarlas a cabo.

Por "responsabilidades" se entiende fundamentalmente lo siguiente:

- Las acciones que realiza un objeto.
- El conocimiento o la información que mantiene un objeto.
- Las decisiones que realiza un objeto afectando a otros.

Objetivos del diseño

Como ya mencionamos anteriormente, uno de los requerimientos del software fue que se respetaran los principios del diseño orientado a objetos resumidos en el acrónimo **SOLID**[15].

- Single Responsability Principle (SRP)
- Open-Close Principle (OCP)

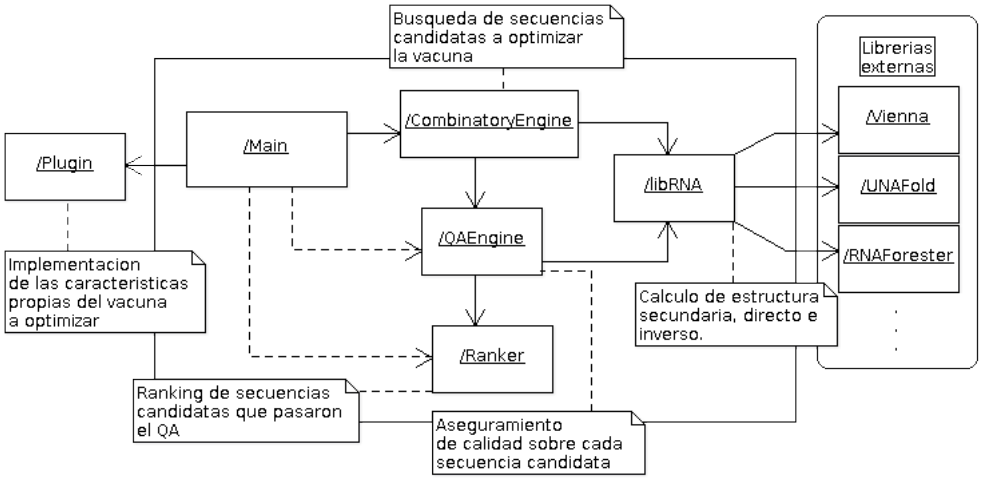

Figura 7.1: Arquitectura de vac-o.

- Liskov Substitution Principle (LSP)
- Interface Segregation Principle (ISP)
- Dependency Inversion Principle (DIP)

En particular se puso especial atención en respetar los principios SRP, OCP y DIP debido a que repercuten directamente en que el sistema sea mas simple de mantener y extender con nuevas funcionalidades. Además, el principio DIP es fundamental para lograr un software que sea verificable mediante la automatización de pruebas como se pudo comprobar mas adelante, en la etapa de "verificación".

7.2. Arquitectura

A continuación veremos los principales componentes del sistema y siguiendo la metodología adoptada, sus respectivas responsabilidades. En la Figura 7.1 se puede ver el diagrama UML correspondiente.

- **Main:** Representa el componente principal en términos de ejecución del sistema. Comprende principalmente, la inicialización y configuración de otros componentes.
- **Plugin:** Representa la implementación de las características propias de la vacuna que se desea optimizar. Brinda información para la configuración inicial como así también, los criterios para evaluar las secuencias candidatas.
- **CombinatoryEngine:** Representa el motor combinatorio del sistema encargado de encontrar, nuevas secuencias que sean candidatas a optimizar la atenuación del virus.

- **QAEngine:** Representa el motor de control de calidad del sistema encargado de decidir, si una secuencia candidata pasa el control o no.
- **Ranker:** Representa el componente encargado de mantener un "ranking" de secuencias que sera además, el resultado final de la ejecución.
- **libRNA:** Representa el componente que provee al sistema funcionalidades para la manipulación de secuencias y estructuras de RNA ("folding" directo e inverso y comparación estructural, entre otras) utilizando librerías externas.

En las siguientes secciones profundizaremos sobre los componentes "libRNA" y "CombinatoryEngine" por ser probablemente los mas importantes en cuanto a sus responsabilidades y dejaremos como lectura adicional la "Especificación de Diseño de Software" que contiene una descripción detallada de todos los componentes de la arquitectura presentada. Al mismo tiempo, profundizar sobre estos componentes, nos permitirá introducir los principales patrones de diseño que se utilizaron de forma análoga en otras partes del sistema. Entre otros, se destacan "Iterator", "Observer", "Template Method" y "Visitor". Para una descripción de estos y otros patrones del diseño orientado a objetos, ver [8].

7.3. Librerías externas

Como vimos en la secciones 3.2.1 y 3.2.2, existen diversas implementaciones que resuelven el problema de la predicción (directa e inversa) de estructura secundaria de RNA y lo mismo ocurre para el caso de la comparación estructural. Además no se descarta la posibilidad de que en el futuro aparezcan nuevas y mejores implementaciones.

Por todo esto, uno de los requerimientos del software fue que sea posible el uso de cualquiera de estas implementaciones de manera transparente. El problema radica en que cada librería tiene diferentes formas de recibir los parámetros de entrada y diferentes formas de dar los resultados que genera. A raíz de esto se propuso el componente "libRNA" como una forma de unificar el acceso a estas librerías externas e integrarlas al resto del sistema. Las interfaces propuestas fueron las siguientes:

- **IFold:** Provee la predicción directa (*folding*) de secuencias de RNA.
- **IFoldInverse:** Provee la predicción inversa (*inverse folding*) de secuencias de RNA.
- **IStructureCmp:** Provee la comparación de estructuras secundarias de RNA.
- **ISequenceCmp:** Provee la comparación de secuencias de RNA.

En esto podemos ver la idea del principio de diseño DIP. Haciendo que el sistema dependa de estas interfaces en lugar de sus respectivas implementaciones conseguimos abstraer los detalles de cada librería externa y logramos un software mas versátil. En el capitulo 8 veremos algunos detalles sobre las implementaciones de estas interfaces en las que intervienen los patrones de diseño "Iterator" y "Visitor".

7.4. Motor combinatorio

El componente "CombinatoryEngine" contiene todo lo referente al recorrido del espacio de búsqueda por lo que es claramente, uno de los componentes principales de vac-o. Esto es fundamentalmente, las regiones combinatorias y las diferentes estrategias de búsqueda.

Para permitir que el software sea de utilidad para diferentes tipos de virus, tanto las regiones combinatorias como la estrategia de búsqueda utilizada para la optimización, debían ser altamente configurables. Luego, el objetivo sobre este componente, fue capturar la estructura general de los algoritmos de búsqueda local que suelen denominarse de "mejoramiento iterativo". Algunos de estos algoritmos son:

- First Improvement
- Best Improvement
- Simulated Annealing
- Tabu Search

En todos estos algoritmos, y en general en la búsqueda local, están presentes los conceptos de "vecindario" (neighborhood) y de "movimiento" (step) que veremos con mayor detenimiento en el capitulo 9. Mientras tanto, la siguiente introducción nos servirá para justificar las interfaces propuestas.

Si el espacio de búsqueda es $\mathcal{S}$, entonces un "vecindario" sera una relación $\mathcal{N} \subseteq \mathcal{S} \times \mathcal{S}$ y el "movimiento" (entre soluciones en el espacio de búsqueda) sera una función $step : \mathcal{S} \to \mathcal{D}(\mathcal{S})$. Donde $\mathcal{D}(S)$ denota el conjunto de distribuciones de probabilidad sobre un conjunto dado S y donde una distribución de probabilidad $D \in \mathcal{D}(S)$ es una función $D : S \to \mathbb{R}_0^+$ que mapea los elementos de S a sus respectivas probabilidades.

Luego, en cada iteración de la búsqueda, hay dos responsabilidades bien diferenciadas e independientes. Por un lado, se debe explorar el "vecindario" de la solución actual. Es decir, si la solución actual es s y el "vecindario" es $\mathcal{N}$, debemos generar el conjunto de soluciones $\mathcal{N}(s)$. Por otro lado, una vez que se tiene el conjunto $\mathcal{N}(s)$ de "vecinos" de s, se debe seleccionar uno de ellos utilizando la función *step*. Notar que no siempre la solución seleccionada sera mejor que la solución actual. Inclusive los algoritmos que permiten los llamados, "movimientos malos" (seleccionar con una probabilidad p una solución peor que la actual) suelen conducir a mejores resultados.

En este sentido se propusieron la siguientes interfaces:

- **ICombinatoryRegion:** Genera las variantes de la región combinatoria que satisfagan la restricción asociada.
- **ISolution:** Representa una solución en el espacio de búsqueda.
- **INeighborhood:** Explora el vecindario de una solución.
- **IStrategy:** Decide como pasar de una solución a la siguiente y notifica las soluciones que "mejoran" a la solución anterior.
- **ICombinatoryEngine:** Inicia la búsqueda y notifica a otros componentes del sistema las sucesivas soluciones encontradas.

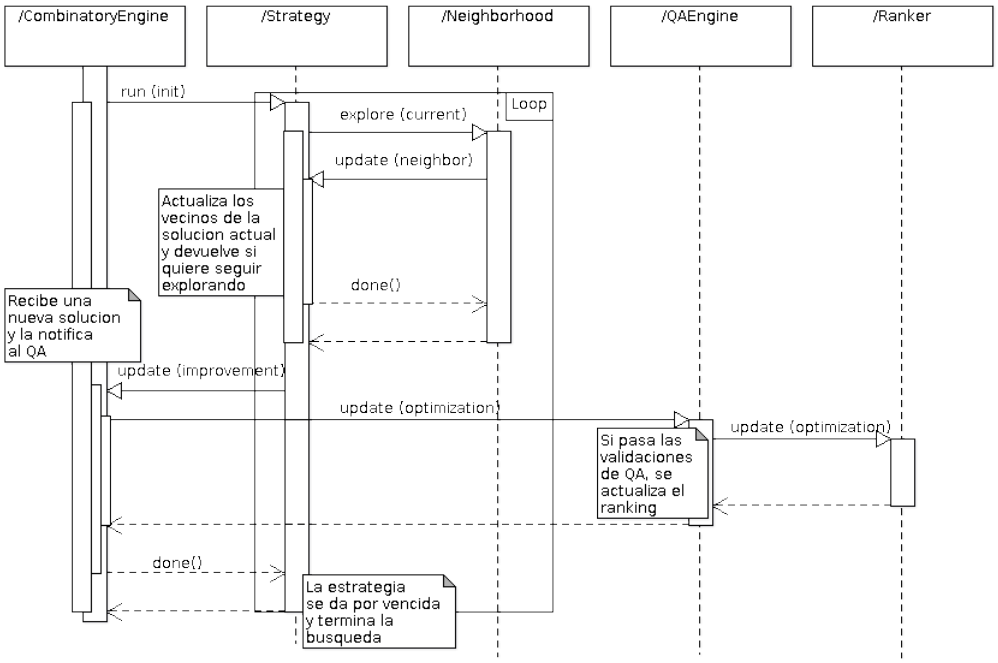

Figura 7.2: Motor combinatorio de vac-o.

Para terminar de entender la interacción entre estas interfaces, en la Figura 7.2 se puede ver el diagrama de secuencia del motor combinatorio.

El principal patrón de diseño que utilizamos en este componente es el que se denomina "Observer" y que sirve fundamentalmente para lograr la comunicación entre diferentes objetos sin que se produzca acoplamiento. La idea es establecer una relación entre un emisor y uno, o varios receptores sin necesidad de generar una dependencia entre el objeto emisor y el objeto receptor. Además para permitir el uso del patrón "Observer" entre diferentes tipos de objetos, se propuso el patrón de diseño "Template Method".

En nuestro caso, **INeighborhood** notifica a **IStrategy** por cada solución que se genera mientras se explora el vecindario de la solución actual. Posteriormente, si la solución seleccionada es mejor que la solución actual, **IStrategy** notifica a **ICombinatoryEngine** la solución seleccionada y este a su vez, la notifica a otros componentes de vac-o (control de calidad).

Una vez mas, remarcamos que las dependencias son entre interfaces y no entre implementaciones como establece el principio de diseño DIP. Esto nos brinda la versatilidad de definir diferentes tipos de regiones combinatorias y estrategias de búsqueda sin modificar en absoluto la implementación del motor combinatorio.

Capítulo 8

Predicción directa e inversa de estructura secundaria

Capítulo 9

Búsqueda local

Parte IV

Conclusiones

Capítulo 10

Todo concluye al fin

> Pase lo que pase, dirija quien dirija, todo el mundo sabe que la camiseta 10 de la selección será mía... Para siempre.
>
> Diego Armando Maradona

10.1. Aportes

10.2. Trabajo futuro

Bibliografía

[1] Mirela Andronescu, Rosalia Aguirre-Hernandez, Anne Condon, and Holger H. Hoos. Rnasoft: a suite of rna secondary structure prediction and design software tools. *Nucleic Acids Research*, 31(13), April 2003.

[2] R. Bruce Aylward and Stephen L. Cochi. Framework for evaluating the risks of paralytic poliomyelitis after global interruption of wild poliovirus transmission. Technical report, World Health Organization, 2004.

[3] Marty R. Badgett, Alexandra Auer, Leland E. Carmichael, Colin R. Parrish, and James J. Bull. Evolutionary dynamics of viral attenuation. *Journal of Virology*, 76(20), October 2002.

[4] Anke Busch and Rolf Backofen. Info-rna - a server for fast inverse rna folding satisfying sequence constraints. *Nucleic Acids Research*, March 2007.

[5] Konstantin Chumakov and Ellie Ehrenfeld. New generation of inactivated poliovirus vaccines for universal immunization after eradication of poliomyelitis. Technical report, National Institute of Health, December 2008.

[6] J. Robert Coleman, Dimitris Papamichail, Steven Skiena, Bruce Futcher, Eckard Wimmer, and Steffen Mueller. Virus attenuation by genome-scale changes in codon pair bias. *Science*, 320:1784, June 2008.

[7] Dirección de Epidemiología. Programa Nacional de Erradicación de la Poliomielitis. Caso de poliomielitis sabin derivado en argentina. Technical report, Ministerio de Salud de la Nación, May 2009.

[8] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns Elements of Reusable Object-Oriented Software*. Addison-Wesley, January 2005.

[9] Paul P Gardner and Robert Giegerich. A comprehensive comparison of comparative rna structure prediction approaches. *BMC Bioinformatics*, September 2004.

[10] I. L. Hofacker, W. Fontana, P. F. Stadler, L. S. Bonhoeffer, M. Tacker, and P. Schuster. Fast folding and comparison of rna secondary structures. *Monatshefte für Chemie*, 125(2), 1994.

[11] Holger H. Hoos and Thomas Stützle. *Stochastic Local Search - Foundations and Applications*. Morgan Kaufmann / Elsevier, 2004.

[12] Nidia H De Jesus. Epidemics to eradication: the modern history of poliomyelitis. *Virology Journal*, July 2007.

[13] Adam S Lauring, Jeremy O Jones, and Raul Andino. Rationalizing the development of live attenuated virus vaccines. *Nature Biotechnology*, 28(6), June 2010.

[14] Rune B. Lyngsø and Christian N. S. Pedersen. Rna pseudoknot prediction in energy based models. *Journal of computational biology*, 2000.

[15] Robert C. Martin. Design principles and design patterns. www.objectmentor.com, 2000.

[16] Philip D. Minor. The molecular biology of poliovaccines. *Journal of General Virology*, 1992.

[17] Steffen Mueller, J Robert Coleman, Dimitris Papamichail, Charles B Ward, Anjaruwee Nimnual, Bruce Futcher, Steven Skiena, and Eckard Wimmer. Live attenuated influenza virus vaccines by computer-aided rational design. *Nature Biotechnology*, 28(7), July 2010.

[18] Marco Vignuzzi, Emily Wendt, and Raul Andino. Engineering attenuated virus vaccines by controlling replication fidelity. *Nature Medicine*, 14(2):154, February 2008.

[19] Rebecca Wirfs-Brock and Alan McKean. *Object Design: Roles, Responsibilities and Collaborations*. Addison-Wesley, 2003.

[20] Michael Zuker and David Sankoff. Rna secondary structures and their prediction. *Bulletin of Mathematical Biology*, 46(4), 1984.

[21] Michael Zuker and Patrick Stiegler. Optimal computer folding of large rna sequences using thermodynamics and auxiliary information. *Nucleic Acids Research*, 9(1), 1981.

Apéndice A

Acrónimos

RNA	Ácido ribonucleico . . . 1
DNA	Ácido desoxirribonucleico . . . 6
OPV	Oral Polio Vaccine . . . 6
WHO	World Health Organization . . . 7
cVDPV	poliovirus circulantes derivados de la vacuna . . . 7
VAPP	parálisis poliomielítica asociada a la vacuna oral . . . 7
FuDePAN	Fundación para el Desarrollo de la Programación en Ácidos Nucleicos . . . 7
vac-o	Combinatory Vaccine Optimizer . . . 7
OOP	Object Oriented Programming . . . 8
GPLv3	GNU General Public License v3 . . . 8
SVN	Subversion . . . 8
A	Adenina . . . 11
G	Guanina . . . 11
T	Timina . . . 11
C	Citosina . . . 11
U	Uracilo . . . 11
mRNA	messenger RNA . . . 13
rRNA	ribosomal RNA . . . 13
tRNA	transfer RNA . . . 13
PV1	Poliovirus type 1 . . . 14
PV2	Poliovirus type 2 . . . 14
PV3	Poliovirus type 3 . . . 14
mfe	minimal free energy . . . 17
CSP	Constraint Satisfaction Problem . . . 18
ORF	Open Reading Frame . . . 12
UTR	Untranslated Region . . . 12