МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ» (НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Кафедра систем информатики

Герман Андреевич Грехов

Разработка data-flow языка программирования и его инструментария для работы с данными секвенирования геномов

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ
по направлению высшего профессионального образования

230100.68 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Тема диссертации утверждена распоряжением по НГУ № 9 от «11» января 2012 г.
Тема диссертации скорректирована распоряжением по НГУ № 539 от «14» декабря 2012 г.

Руководители
Скопин И.Н.
к. ф.-м. н., с. н. с., доцент

Фурсов М.Ю.
б/с

Новосибирск, 2013 г.

МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ» (НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Кафедра систем информатики

УТВЕРЖДАЮ

Зав. кафедрой Лаврентьев Михаил Михайлович

…….

ЗАДАНИЕ
на магистерскую диссертацию

студенту Грехову Герману Андреевичу

факультета ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Направление подготовки 230100.68 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

Магистерская программа ТЕХНОЛОГИЯ РАЗРАБОТКИ ПРОГРАММНЫХ СИСТЕМ

Тема «Разработка data-flow языка программирования и его инструментария для работы с данными секвенирования геномов»

Цели работы: предоставить пользователю способ создания и настройки вычислительных конвейеров, а также набор программных инструментов (вычислительных блоков/этапов конвейеров) для решения задач анализа данных секвенирования геномов.

Руководители
Скопин И.Н.
к.ф.-м.н., с.н.с, доцент

……..

Фурсов М.Ю.
б/с

……..

Содержание

ВВЕДЕНИЕ

Многие задачи современной молекулярной биологии и биоинформатики требуют применения целого комплекса программных инструментов для своего решения. Это особенно касается задач анализа данных секвенирования геномов [1, 2, 3]. Каждый инструмент решает одну из подзадач, и полученные результаты используется в последующих вычислениях и анализе. Таким образом, этот комплекс инструментов образует вычислительный конвейер, и работа ученого заключается в том, чтобы в ходе многократных запусков этого конвейера подобрать входные данные и правильные параметры и на основе этого получить результаты исследования.

Стремительное увеличение количества задач, их нарастающая сложность и огромные объемы данных, используемых в анализе, делают необходимым применение специализированных программных систем, предоставляющих возможности построения и настройки вычислительных конвейеров. Такие системы обеспечивают работу пользователя с большим набором программ через единый графический интерфейс. Кроме того, системы построения вычислительных конвейеров позволяют наглядно представить, из каких этапов состоит конвейер, и как эти этапы связаны между собой.

Одной из таких систем является UGENE Workflow Designer [4] – подсистема открытой биоинформационной программной платформы UGENE [5], предоставляющая возможности построения и настройки вычислительных конвейеров.

Особенность UGENE Workflow Designer по сравнению с аналогами состоит в том, что эта система позиционируется как среда разработки на графическом языке программирования. Им является внутренний язык программирования системы под названием UWL [6] (UGENE Workflow Language). И всякое развитие системы является развитием UWL или его инструментария разработки.

Главная задача данной работы – это развитие UWL до полноценного языка программирования и развитие его среды разработки UGENE Workflow Designer для обеспечения возможности создания и гибкой настройки вычислительных конвейеров, решающих задачи анализа данных секвенирования геномов.

Результат работы будет распространяться как часть свободного кроссплатформенного программного пакета UGENE. Программная платформа UGENE имеет распространение среди ученых по всему миру и доступна для пользования как с

сайта проекта, так и непосредственно из таких дистрибутивов Linux, как Ubuntu и Fedora, что должно способствовать широкому распространению созданного в рамках данной работы решения.

Глава 1. Задача построения вычислительных конвейеров

1.1 Постановка задач

Общая цель данной работы – автоматизировать рабочий процесс молекулярного биолога с данными секвенирования геномов. То есть предоставить пользователю набор программных инструментов, необходимых для проведения исследований в этой области.

Зачастую, решение той или иной задачи молекулярной биологии или биоинформатики разделяется на решение нескольких подзадач и, таким образом, требует применения целого комплекса программных инструментов. Причем каждый инструмент работает либо с исходными входными данными, либо с данными, полученными в ходе решения других подзадач. Этот комплекс инструментов образует вычислительный конвейер [7].

Конкретизация цели работы – это предоставить пользователю способ создания и настройки вычислительных конвейеров, а также набор программных инструментов (вычислительных блоков/этапов конвейеров) для решения задач анализа данных секвенирования геномов.

Молекулярные биологи относятся к категории пользователей компьютера, требующих низкий порог вхождения и высокую запоминаемость [8] к интерфейсам программ. Поэтому удовлетворение этих требований, то есть предоставление удобного графического интерфейса ко всем предоставляемым инструментам по анализу данных секвенирования, является дополнительной целью работы.

1.2 Анализ и конкретизация задачи

Анализируя деятельность молекулярных биологов, уточняя и дополняя общие цели работы и переходя к поиску их решения, можно поставить более конкретные цели, которые частично или полностью достигаются в данной работе.

Упомянутая выше автоматизация заключается не столько в разработке инструментов для решения тех или иных задач по анализу данных секвенирования, сколько в разработке программной системы, предоставляющей возможности интеграции этих инструментов. Эта система должна предоставлять единый графический интерфейс доступа ко всем инструментам, что обеспечит требуемые низкий порог вхождения и высокую запоминаемость.

Кроме того, эта система должна быть расширяемой, то есть предоставлять возможности по добавлению в нее новых инструментов. Это позволит наращивать инструментарий пользователя системы и увеличивать круг решаемых с помощью нее задач.

1.3 Анализ существующих решений

Существуют десятки различных программных систем, позволяющих конструировать вычислительные конвейеры. Это как системы общего назначения, такие как LabVIEW [9], Agilent VEE [10], так и системы для работы с биологическими данными, самыми популярными из которых являются Taverna Workbench [11], Galaxy [12] и UGENE Workflow Designer [4]. Основная часть данного анализа была проведена именно по биоинформационным системам.

Первая программная система проведенного анализа – это Galaxy [12]. Данная система распространяется с открытым исходным кодом, а разрабатывается на языке Python. Galaxy обладает очень удобным интерфейсом, который легко осваивается пользователем. Также эта система предоставляет большой набор вычислительных блоков (возможных элементов конвейера).

Система Galaxy способна к расширению практически без дополнительного программирования. Для добавления нового инструмента его необходимо описать на языке XML [12].

Основной недостаток Galaxy состоит в том, что это веб-приложение. Данный вид распространения практически лишает ученых использовать собственную копию приложения. Ведь большинство биологов не смогут настроить веб-сервер и установить и сконфигурировать на нем Galaxy. К тому же, серверная часть Galaxy функционирует только на UNIX-подобных операционных системах, а значит, в большинстве случаев для ее установки биологу потребуется отдельная реальная или виртуальна машина.

Использованию же бесплатных публичных серверов с Galaxy сопутствует ряд ограничений на количество доступного дискового пространства и вычислительной мощности. Также в этом случае потребуется перекачка пользовательских данных на публичный сервер, что слишком трудоемко при работе с данными ассемблирования, так как их размеры зачастую составляют десятки гигабайтов.

Другая программная система – Taverna Workbench [11] (далее для краткости Taverna). Эта система разработана на Java и также распространяется с открытым исходным кодом. В отличие от Galaxy система Taverna является приложением, устанавливаемым на компьютер пользователя, и способна работать автономно.

Система Taverna предоставляет пользователю большой набор вычислительных блоков. Но главный недостаток этой системы – это плохой уровень юзабилити [8]. Интерфейс этой программы имеет очень высокий порог вхождения, то есть очень сложен для конечного пользователя.

Последняя анализируемая система – UGENE Workflow Designer [4] (далее для краткости WD). Это программное средство является одной из подсистем программной системы UGENE, которая разработана на языке C++ с использованием библиотек Qt 4 [13], что делает ее кросс-платформенной, то есть позволяет использовать на операционных системах семейства Windows, Linux и Mac OS. Как и описанные выше системы, WD является свободно-распространяемым программным обеспечением с открытым исходным кодом.

Преимущество WD над Galaxy заключается в том, эта система устанавливается на компьютер пользователя и после этого полноценно функционирует. Также WD обладает графическим интерфейсом, который намного удобнее, чем интерфейс Taverna.

Общим преимуществом WD над аналогами является то, что WD – это часть большей системы UGENE, которая помимо инструмента построения вычислительных схем обладает большим набором средств визуализации биологических данных и алгоритмов их анализа. То есть в результате работы с WD пользователь получает выходные данные, и ему не нужны другие программы для дальнейшего просмотра файлов с результатами, так как все полученные данные можно визуализировать в этой же системе за минимальное количество шагов.

Один их недостатков WD в том, что в нем нет вычислительных блоков по анализу данных секвенирования. Другой недостаток – это отсутствие механизма по расширению инструментария без программирования.

1.4 Предлагаемое решение

В рамках данной работы предлагается развить UGENE Workflow Designer в нескольких направлениях. Первое направление – это разработка механизма по

добавлению в WD новых инструментов. Этот механизм будет позволять расширять инструментарий WD без дополнительного программирования и компиляции кода UGENE.

Другое направление – это развитие инфраструктуры WD для работы с данными секвенирования. Это такие работы как добавление новых типов и форматов данных, вычислительных блоков для считывания и записи этих данных.

Третье направление – это развитие уровня юзабилити [8]. В рамках этого направления будет добавлен такой новый вид настройки вычислительных схем, как визард [13] (или помощник). Этот вид настройки обеспечивает удобный интерфейс взаимодействия с WD. Если биологу предоставить готовую вычислительную схему, снабженную визардом, то он сможет настроить ее и решить свою задачу, даже если он не знаком ни с WD, ни с понятием вычислительных конвейеров.

Глава 2. Введение в UGENE Workflow Designer

Система UGENE Workflow Designer в течение долгого времени подвергалась экстенсивному развитию, то есть простому наращиванию набора инструментов (вычислительных элементов, этапов конвейеров). Вычислительные схемы сохранялись в файлах бинарного формата, а задачи разработки средств управления потоками данных в конвейерах или просмотра временных результатов не ставилась. Впоследствии стало очевидным, что без решения этих задач система превратилась бы в чрезмерно тяжелую как для использования, так для сопровождения и дальнейшего развития.

На следующем этапе развития система UGENE Workflow Designer стала рассматриваться как среда разработки на графическом dataflow-языке [8] программирования, а вычислительные схемы — как программы на этом языке. Также, построенные вычислительные схемы сохраняются в файлы не в бинарном, а в текстовом формате. Текст схемы — это тоже программа на предлагаемом языке программирования, но написанная с помощью его нового текстового синтаксиса. Таким образом, появились два эквивалентных конкретных синтаксиса [14] (графический и текстовый) одного языка программирования, который стал называться UGENE Workflow Language или, сокращенно, UWL [6].

На каждом из дальнейших этапов развития WD соблюдалось следующее правило: прежде чем вносить какое-либо улучшение в Workflow Designer, нужно

проанализировать, как это улучшение будет отражено в языке, не нарушая его особенностей и избегая дублирования средств.

2.1 Вычислительная модель и абстрактный синтаксис

Язык UWL, как и любой другой язык программирования, имеет абстрактный синтаксис [14], который описывает его вычислительные возможности, т.е. модель вычислений, и конкретный синтаксис, определяющий, как эта модель представляется пользователям. Как уже упоминалось, UWL предлагает два варианта конкретных синтаксисов: текстовый, обеспечивающий традиционные возможности записи программ, и графический, конструкции которого изображаются как элементы схем.

Абстрактный синтаксис [14] предназначен для внутреннего представления программ и характеризуется, в основном, вычислительной моделью языка программирования. Конкретные синтаксисы [14] предназначены для программистов, то есть пользователей UGENE Workflow Designer, и непосредственно позволяют конструировать (программировать) конвейеры вычислений.

Основными понятиями, которые используются при описании абстрактного синтаксиса и вычислительной модели UWL, являются вычислительный элемент, порт, слот, шина данных, сообщение, граф соединения портов [6].

UWL — это dataflow-язык [7], поэтому основную часть его вычислительной модели составляют вычислительные элементы и соединяющие их каналы связей. **Вычислительный элемент** (Worker или actor) — это некоторая многократно исполняющаяся программа, являющаяся одним из этапов конвейера. Это может быть реализация какого-либо биологического алгоритма, или программа, считывающая или записывающая данные на жесткий диск или удаленную базу данных, или какой-либо служебный элемент, управляющий потоками данных.

Порт (Port) — это средство, с помощью которого вычислительный элемент обменивается данными с другими элементами. Соответственно, не имея портов, вычислительный элемент не имеет и смысла, так как его невозможно включить в какой-либо конвейер вычислений. Порты бывают входные, в которые приходят данные из других вычислительных элементов; и выходные, куда отправляет данные вычислительный элемент, хозяин этого порта.

Соединяя между собой выходной и входной порты каких-либо двух вычислительных элементов, пользователь создает шину данных [7]. **Шины данных** (Bus) служат для передачи данных между вычислительными элементами. Совокупность всех шин данных образует **граф потоков данных.**

Данные в шине передаются в сообщениях. **Сообщение** — это способ упаковки данных для передачи по шине.

Данные в шинах всегда передаются в контексте своего существования (вместе с теми данными, на основе которых были произведены эти данные). То есть если вычислительные элемент принял входные данные и на основе них (в их контексте) произвел новые данные, то эти новые данные поступают в выходной порт вместе со своим контекстом. При этом изменить данные в шинах невозможно, а можно только создать новые данные в контексте старых.

Эти соглашения влекут за собой ряд следствий, которые являются особенностями системы WD. Главное следствие – это то, что при движении от начала конвейера к его концу шины данных наращиваются, так как контекст существования данных увеличивается от элемента к элементу. Ведь после того как вновь воспроизведенная единица данных и контекст ее существования переданы в выходной порт, они образуют новый контекст существования для данных, которые будут воспроизведены следующими элементами.

Таким образом, сообщение в шине данных может содержать несколько единиц данных. Это достигается тем, что данные из сообщений не удаляются, а только добавляются. И, в конце концов, в порт какого-либо этапа конвейера может прийти сообщение, содержащее различные единицы данных, предназначенные для разных вычислителей. Некоторых из них предназначены именно для этого вычислителя, а другие нужно просто передать в выходной порт вместе с воспроизведенными данными. Для разграничения таких единиц данных предназначены слоты.

Слот — это параметр порта, описывающий данные, приходящие во входной порт или выходящие из выходного порта. Таким образом, порт — это не просто сущность, принимающая или отправляющая сообщения в шину, а набор слотов, описывающих данные, которые этот порт принимает или отдает. Соответственно, как и порты, слоты бывают входные и выходные.

Пользователь при конструировании схемы, помимо соединения портов, настраивает, как соединены и слоты. То есть пользователь указывает, какие именно данные в шинах предназначены для того или иного вычислительного элемента. Например, на рисунке 1 пользователь настраивает входной слот «Sequence», выбирая, какая из входящих последовательностей предназначена для этого слота.

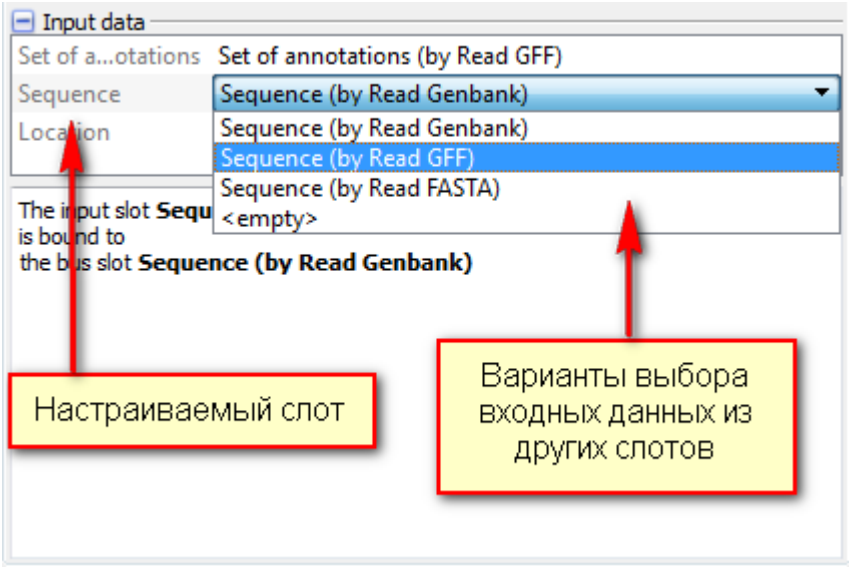

Рис. 1. Пример настройки соединения слотов.

2.2 Модель данных

В модели данных UWL поддерживается строгая статическая типизация. Все данные, передаваемые по шинам, имеют конкретный тип. И каждый слот описывается типом передаваемых/принимаемых данных. Соединять два слота можно только в том случае, если их типы совпадают. Таким образом, единица данных, помещенная в какое-либо сообщение, идентифицируется типизированным слотом, а значит, может быть передана только слоту с таким же типом.

Соответствие типов проверяется статически, т.е. перед стартом вычислений на стадии валидации. При несовпадении типов пользователь будет об этом извещен, а вычисления не будут начаты.

UGENE Workflow Designer позволяет оперировать следующими типами данных:

1) Text — любые текстовые данные.

2) Sequence — биологическая последовательность. Содержит символы, описывающие нуклеотиды ДНК, РНК или белковых последовательностей, и некоторые метаданные.

3) MSA — множественное выравнивание. Содержит список последовательностей и информацию о расположении их друг относительно друга.

4) Annotation table — набор аннотаций. Описывает участки в последовательностях, выравниваниях дополнительными сведениями.

2.3 Конкретный синтаксис

Язык программирования UWL имеет два конкретных синтаксиса: графический и текстовый.

Графический конкретный синтаксис с помощью визуальных средств UGENE Workflow Designer позволяет конструировать вычислительную схему. Пользователю предоставляется графическая сцена и набор вычислительных элементов (палитра), которые он может с помощью мыши добавлять на сцену, перемещать для удобного визуального расположения и соединять порты элементов стрелками, создавая, таким образом, шины данных. Также, существуют графические элементы для регулирования параметров и соединения слотов.

Текстовый конкретный синтаксис предоставляет пользователю все возможности конструирования схем, используя текстовый редактор. Он подробно описан в статье «Языковые средства организации вычислений в области биоинформатики» [6].

Графический и текстовый синтаксисы являются эквивалентными. То есть любая вычислительная схема, сконструированная с помощью одного синтаксиса, может быть сконструирована и с помощью другого синтаксиса. Более того, можно создать какую-либо схему с помощью графического синтаксиса, сохранить ее в файл, и этот файл будет семантическим эквивалентом графической схемы в текстовом синтаксисе. Изменив текстовый файл, можно переоткрыть его в UGENE Workflow Designer и наблюдать изменения.

Глава 3. Развитие языка UWL

3.1 Разработка служебных элементов

Одним из недостатков UGENE Workflow Designer является отсутствие элементов управления потоками данных. Это лишает систему возможности построения сложных вычислительных схем, решающих задачи комплексного анализа данных, и любое дополнительное условие в вычислительной схеме требует доработки WD.

Предлагаемое решение проблемы связано с разработкой набора вычислительных элементов из категории служебных. Эти элементы отвечают за такие процессы, как маркировка данных в потоке, фильтрация, группировка нескольких сообщений в одно, мультиплексирование двух потоков данных в один (эти процессы подробно описаны в пунктах 3.1.1-3.1.4).

Появление этих элементов вводит в UWL такое понятие как конструкции языка. Служебные элементы являются конструкциями языка, то основой, из чего пользователь может строить каркас и сложную логику вычислительных схем.

Остальные вычислительные элементы образуют стандартную библиотеку. Введение же механизма расширения системы новыми вычислительными элементами добавляет в систему понятие пользовательской библиотеки.

Далее представлено описание созданных в рамках данной работы служебных элементов будет, разделенное на пункты: предназначение, принцип работы, выражение в языке UWL.

3.1.1 Маркировщик

Предназначение

Этот служебный элемент предназначен для маркировки данных в потоке, то есть сопоставляет каждому сообщению определенную метку (или маркер) по заданному пользователем условию.

Например, в потоке данных передаются биологические последовательности, а исследователю требуется разделить эти последовательности на те, которые больше определенной длины, и те, которые меньше, чтобы затем совершить над этими группами последовательностей разные действия. Элемент маркировщик позволяет задать критерий на длину последовательности, и каждому сообщению, содержащему последовательность, будет присвоена своя метка: «длинная» или «короткая».

Принцип работы

Маркировщик имеет один входной и один выходной порт. Во входной порт приходят сообщения с данными, а из выходного порта выходят эти же сообщения, но дополненные одной или несколькими маркерами.

Каждому маркеру соответствует заданное пользователем условие на данные во входном сообщении. Маркеры объединены в группы, и для каждой группы создается отдельный выходной слот, в который будет помещен результат маркировки. Пользователь может создать неограниченное количество групп маркеров, содержащих неограниченное количество маркеров и условий их назначения.

Можно наглядно пояснить принцип работы маркировщика на примере решения задачи о длинных и коротких последовательностях, которая описана выше. В этом примере пользователю нужно создать одну группу маркеров, которая называется, например, «маркеры длины последовательностей». Для этой группы маркеров создается одноименный выходной слот, в который будет помещен результат маркировки для каждого входного сообщения. Далее, необходимо добавить два маркера и два условия в эту группу маркеров:

1) *Маркер*: «длинная». *Условие*: длина последовательности больше 1000 символов.

2) *Маркер*: «короткая». *Условие*: все остальные последовательности.

Если длина последовательности во входящем сообщении меньше 1000 символов, то в выходной слот «маркеры длины последовательностей» будет помещен маркер «короткая», иначе – маркер «длинная».

Выражение в текстовом синтаксисе

Маркировщик является вычислительным элементом, значит, его описание на языке UWL должно быть в разделе описания этих конструкций и подчиняться тем же синтаксическим правилам: блок с названием переменной элемента, в котором перечислены его параметры и их значения. Устройство маркировщика сложнее устройства остальных вычислительных элементов, поэтому описание его параметров пришлось расширить с помощью введения дополнительных внутренних блоков. Схема описания маркировщика выглядит следующим образом (в скобках <> указаны значения, которые задает программист):

```
<variable-name> {
    type : marker;
    name : <marker-name>;
    markers {
        <marker-group-name-1> {
            marker-type : <marker-type>;
            <clause-1> : <marker-1>;
            …
            <clause-n> : <marker-n>;
        }
        …
        <marker-group-name-n> {
            …
        }
    }
}
```

Названия групп маркеров должны быть уникальными в контексте своего элемента-маркировщика, так как каждой группе соответствует одноименный выходной слот, а названия слотов должны быть разными.

Значения маркеров <marker-i> должны быть уникальными в контексте своей группы маркеров, так как маркерам соответствуют разные условия на данные.

Одно из значений <clause-i> в группе маркеров может быть “rest”, которое предназначено для маркировки данных, не подходящих ни под одно из остальных условий.

Значение <marker-type> описывает, какими данными и какими свойствами данных оперирует эта группа маркеров. Некоторые из доступных типов маркеров: sequence-length, sequence-name, annotations-count.

Пример описания маркировщика длинных и коротких последовательностей:

```
Sequence-marker {
    type : marker;
    name : “Маркировщик последовательностей”;
    markers {
        length {
            marker-type : sequence-length;
            ">=1000" : long;
            "rest" : short;
        }
    }
}
```

3.1.2 Фильтр

Предназначение

Фильтрация – один из основных и необходимых средств управления потоками данных. Исследователям часто приходится решать задачи, в которых требуется отделить одни данные от других и совершить над ними различные действия. Для этой цели в WD был разработан служебный элемент фильтр.

Принцип работы

Фильтр имеет один входной и один выходной порт. Во входной порт приходят сообщения с данными, а из выходного порта выходят эти же сообщения, но только те из них, которые прошли фильтрацию.

Фильтр имеет один параметр – значение фильтрации, и один входной слот типа Text, в который приходят те данные сообщения, которые нужно сравнивать со значением фильтрации. Те сообщения, данные сравнения которых совпадают со значением фильтрации, направляются в выходной порт. Остальные сообщения блокируются.

Так как входной слот и значение фильтрации имеют тип Text, то получается, что с помощью этого элемента невозможно совершать сложную фильтрацию, например, разделять последовательности на те, которые аннотированы, и которые не аннотированы. Для этого фильтр необходимо использовать в связке с маркировщиком. Маркировщик позволяет генерировать тестовые данные на основе любых признаков данных, будь то количество аннотаций или длина последовательности. И полученный слот маркера нужно соединить со слотом фильтра для получения результата.

Такое условие, как использование двух элементов для сложной фильтрации вместо одного, делает систему более простой для развития. Можно развивать ее, добавляя новые типы маркеров, но способ фильтрации останется прежним.

Выражение в текстовом синтаксисе

Фильтр ничем не отличается от остальных вычислительных элементов стандартной библиотеки, поэтому его описание на языке UWL имеет обыкновенный вид.

```
<variable-name> {
    type : filter-by-values;
    name : <filter-name>;
```

```
    text : "<filter-value-1>; … ; <filter-value-n>";
}
```

Значения <filter-value-i> – это значения фильтрации. Если сравниваемое значение совпадает хотя бы с одним из них, то сообщение передается в выходной порт.

3.1.3 Мультиплексор

Предназначение

Зачастую вычислительные элементы работают с несколькими данными. Например, элементу поиска подпоследовательностей требуются две единицы данных: последовательность и искомая подпоследовательность.

Для этого долгое время в WD применялось следующее решение: добавление в элемент нескольких входных портов вместо одного. В каждый порт приходят отдельные данные, и элемент работает. Но это создает проблему организационного характера: каждый такой элемент управляет мультиплексированием [7] потоков данных по-своему. Действительно, есть ряд элементов, которые помимо своей конечной цели (вычисления) выполняют также работу по разбору входящих сообщений из своих портов и сопоставлению их друг с другом.

Идея мультиплексора состоит в том, что все дополнительные действия по разбору сообщений из разных потоков выделены в отдельный элемент. Именно мультиплексор имеет два входных порта и один выходной, и именно он занимается соединением (мультиплексированием) потоков данных. Сформированный поток же поставляется в единственный входной порт другого элемента.

Введение мультиплексора – это типичное соблюдение принципа separation of concerns [15] (разделение ответственностей).

Принцип работы

Мультиплексор имеет два входных (условно, «первый» и «второй») и один выходной порт. Во входные порты приходят сообщения с данными, которые соединяются в новое объединенное сообщение и отправляются в выходной порт.

Соединение сообщений происходит по правилу, заданному пользователем. Предоставлены три правила соединения:

1) Один к одному. Из первого и второго входных портов берется по одному сообщению, и их объединение поступает в выходной порт.

2) Один ко многим. Для каждого сообщения из первого порта по очереди берется каждое сообщение из второго порта, и их объединение поступает в выходной порт.

3) Много к одному. Такое же правило, как и предыдущее, но первый и второй порты поменяны местами.

Для первого режима: если один поток длиннее другого (содержит больше сообщений), то мультиплексирование происходит, до тех пор пока не кончится меньший поток.

Для второго и третьего режимов: если один из потоков пуст, то не будет сформировано ни выходного одного сообщения.

Выражение в текстовом синтаксисе

Мультиплексор ничем не отличается от остальных вычислительных элементов стандартной библиотеки, поэтому его описание на языке UWL имеет обыкновенный вид.

```
<variable-name> {
    type : multiplexer;
    name : < multiplexer-name>;
    multiplexing-rule : <rule>;
}
```

Значения `<rule>` – это номер правила (0, 1 или 2).

3.1.4. Группировщик

Предназначение

Одна из часто возникающих задач при работе с потоками данных – это соединение нескольких единиц данных. Примером может быть соединение нескольких последовательностей в новую последовательность или в множественное выравнивание, формирование объединенной таблицы аннотаций из нескольких таблиц аннотаций и другое.

Служебный элемент группировщик позволяет разделять сообщения в потоке данных на группы по заданному критерию. Для каждой группы сообщений формируется одно

выходное сообщение, содержащее объединение всех единиц данных группы по заданному пользователем принципу.

Принцип работы

Группировщик имеет один входной и один выходной порт. Во входной порт поступают сообщения с данными, из которых формируются объединенные сообщения и направляются в выходной порт.

Пользователь выбирает, какой из входных слотов определяет группировку (слот группировки). Элемент собирает сообщения в группы на основе данных из слота группировки. То есть очередная уникальная единица данных в слоте группировки инициирует создание новой группы.

Другая настройка, задаваемая пользователем, – это действия, совершаемые над группированными данными, то есть способ соединения данных из нескольких сообщений в одно.

Группировщик уникален тем, что этот элемент порождает новый контекст, ведь данные исходных сообщений не передаются в выходной порт.

Если слот группировки не задан, то все входные сообщения будут сгруппированы в одно новое сообщение.

Выражение в текстовом синтаксисе

Выражение группировщика в текстовом синтаксисе языка UWL сложнее предыдущих элементов, так как в его описании два уровня вложенности.

```
<variable-name> {
    type : grouper;
    name : <grouper-name>;
    group-slot : <group-slot-id>;
    out-slot {
        name : <out-slot-name-1>;
        in-slot : <in-slot-id-1>;
        action {
            type : <action-type>;
            <parameter-1> : <value-1>;
            …
            <parameter-n> : <value-n>;
        }
    }
```

```
    …
    out-slot {
        name : <out-slot-name-k>;
        in-slot : <in-slot-id-k>;
        action {
            type : <action-type-k>;
            <parameter-1> : <value-1>;
            …
            <parameter-t> : <value-t>;
        }
    }
}
```

Значение <group-slot-id> - это идентификатор слота группировки, состоящий из тройки идентификаторов «элемент.порт.слот».

После описания общих параметров следует список описаний новых выходных слотов (блоки «out-slot»). Каждый блок содержит информацию о том, как называется новый выходной слот (<out-slot-name-i>), данные какого входного слота сгруппированы в единицы данных этого слота (<in-slot-id-i>), и какие действия должны быть совершены над этими данными для соединения их в новую единицу данных (блок «action»).

Пользователю предоставлены несколько типов действий (<action-type-i>) над различными типами данных (такие как «merge-sequence», «merge-string», «sequence-to-msa»). Каждое действие имеет свой набор параметров (<parameter-i>) .

3.2 Пользовательская библиотека

Введение служебных элементов разделило все вычислительные элементы на две группы: конструкции языка и стандартная библиотека.

В рамках данной работы был разработан механизм расширения инструментария WD пользователями, который позволяет добавить в систему новые типы вычислительных элементов без дополнительного программирования на C++ и компиляции UGENE. Эти вычислительные элементы в процессе своей работы позволяют запускать сторонние программы.

Созданные пользователем вычислительные элементы составляют третью группу элементов – пользовательская библиотека.

Для того чтобы быть интегрированной в UGENE, сторонняя программа должна быть совместима для работы в WD. Для этого должны быть выполнены условия:

1. Программа имеет интерфейс командной строки.
2. Она оперирует входными и выходными файлами, форматы которых поддерживает UGENE.
3. Данные в файлах этих форматов имеют один из типов, которые поддерживает WD.

Если требуемая программа удовлетворяет этим условиям, то пользователь может запустить специальный диалог (визард), в котором, пройдя шаг за шагом, он интегрирует программу со всей системой. Этот визард предложит настроить параметры, входные и выходные порты и слоты нового элемента. Последний шаг настройки — указание того, как запускать эту программу через интерфейс командной строки. Указанная строка запуска программы может содержать имена только что созданных параметров и слотов.

В результате работы визарда в WD появится новый полноценный вычислительный элемент, который можно использовать в любой схеме. Связь между сторонней программой и WD во время исполнения схемы происходит с помощью временных файлов.

3.3 Интерфейс базы данных

Архитектура всей системы UGENE (не только Workflow Designer) предоставляет способ взаимодействия с базой данных. В ней описан интерфейс работы с базой данных (database interface, dbi), реализуя который программист может единым способом использовать ту или иную СУБД.

WD предназначен для организации вычислений над данными большого объема, которые не помещаются в оперативной памяти. Поэтому использование базы, как временного хранилища данных, является очень эффективным средством для достижения этой цели.

В рамках данной работы было произведено внедрение dbi в работу вычислительных схем. Результатом применения этой технологии стало то, что вычислительные элементы обмениваются не данными, а их идентификаторами во временной базе данных. Эта работа

завершена для самых ресурсоемких типов данных в WD: sequence, msa, assembly, variation track.

Использование описанного механизма необходимо для вычислений над данными большого объема, ведь он позволяет, например, не хранить в оперативной памяти все 3.2 Гб генома человека, а извлекать из базы данных только нужную в данный момент его часть. Это актуально для анализа данных секвенирования геномов.

В настоящее время интерфейс базы данных UGENE реализован для СУБД SQLite [16], которая в полной мере обеспечивает возможности хранения временных данных.

Переход на использование интерфейса базы данных – это первый шаг к распределенным вычислениям. Следующим шагом будет реализация этого интерфейса для какой-либо сетевой системы управления базой данных (например, MySQL), так как сетевая СУБД – это один из способов коммуникации между узлами распределенной вычислительной системы.

Глава 4. Анализ данных секвенирования

Модель данных WD поддерживает типизацию, поэтому первым шагом поддержки анализа данных секвенирования было добавление новых типов данных. Эта область молекулярной биологии в первом приближении оперирует следующими данными:

1) Короткие чтения, полученные в результате работы секвенатора. Эти данные являются обычными последовательностями, и для них используется тип Sequence.

2) Данные ассемблирования (выровненные короткие чтения). Для их описания был создан новый тип данных Assembly, который содержит информацию о коротких чтениях и их взаиморасположении.

3) Геномные вариации. Для их описания был создан новый тип данных Variations Track, который содержит набор описаний вариаций (позиция, производимая замена, ссылки на публичные базы данных и др.).

Таким образом, в WD были введены два новых типа данных: Assembly и Variations Track. Для их поддержки были поддержаны новые форматы данных и разработаны вычислительные элементы для чтения и записи этих данных:

1) Элементы «Read Assembly» и «Write Assembly» для чтения и записи данных типа Assembly соответственно. Поддерживают форматы данных BAM и SAM [3].

2) Элементы «Read Variations» и «Write Variations» для чтения и записи данных типа Variations Track соответственно. Поддерживают форматы данных SNP и VCF4.

В результате WD стал способен оперировать данными секвенирования геномов, и следующей задачей стала интеграция новых вычислительных элементов для их анализа. В рамках данной работы были интегрированы 3 сторонних конвейера для анализа данных секвенирования:

1) Call variants with Samtools [3]. Этот вычислительный конвейер предназначен для поиска вариаций в данных ассемблирования.

2) Tuxedo pipeline [2]. Предназначен для анализа данных RNA-seq, поиску новых генов в данных секвенирвоания.

3) Cistrome pipeline [1]. Предназначен для анализа данных ChiP-seq, определению экспрессии генов.

В результате интеграции этих сторонних конвейеров в WD появилось 11 новых вычислительных элементов.

Глава 5. Графический интерфейс пользователя

Для того чтобы облегчить работу с системой неопытным пользователям и обеспечить низкий порог вхождения в интерфейс программной системы был разработан новый способ настройки вычислительных схем: визард настройки схем (на рис. 2 изображена одна из страниц визарда для настройки Tuxedo pipeline). Этот визард позволяет настроить готовую вычислительную схему (задать значения параметров и входные данные) минуя интерфейс WD.

Визард можно добавить для любой вычислительной схемы. Визард описывается на языке UWL и сохраняется в одном файле со схемой в разделе описания метаинформации (блок «.meta {…}»).

После открытия схемы, снабженной визардом, пользователь может вызвать его с помощью кнопки на панели инструментов системы и, переходя по его страницам, настроить вычислительную схему.

Подробное описание синтаксиса UWL для написания визардов находится в приложении А.

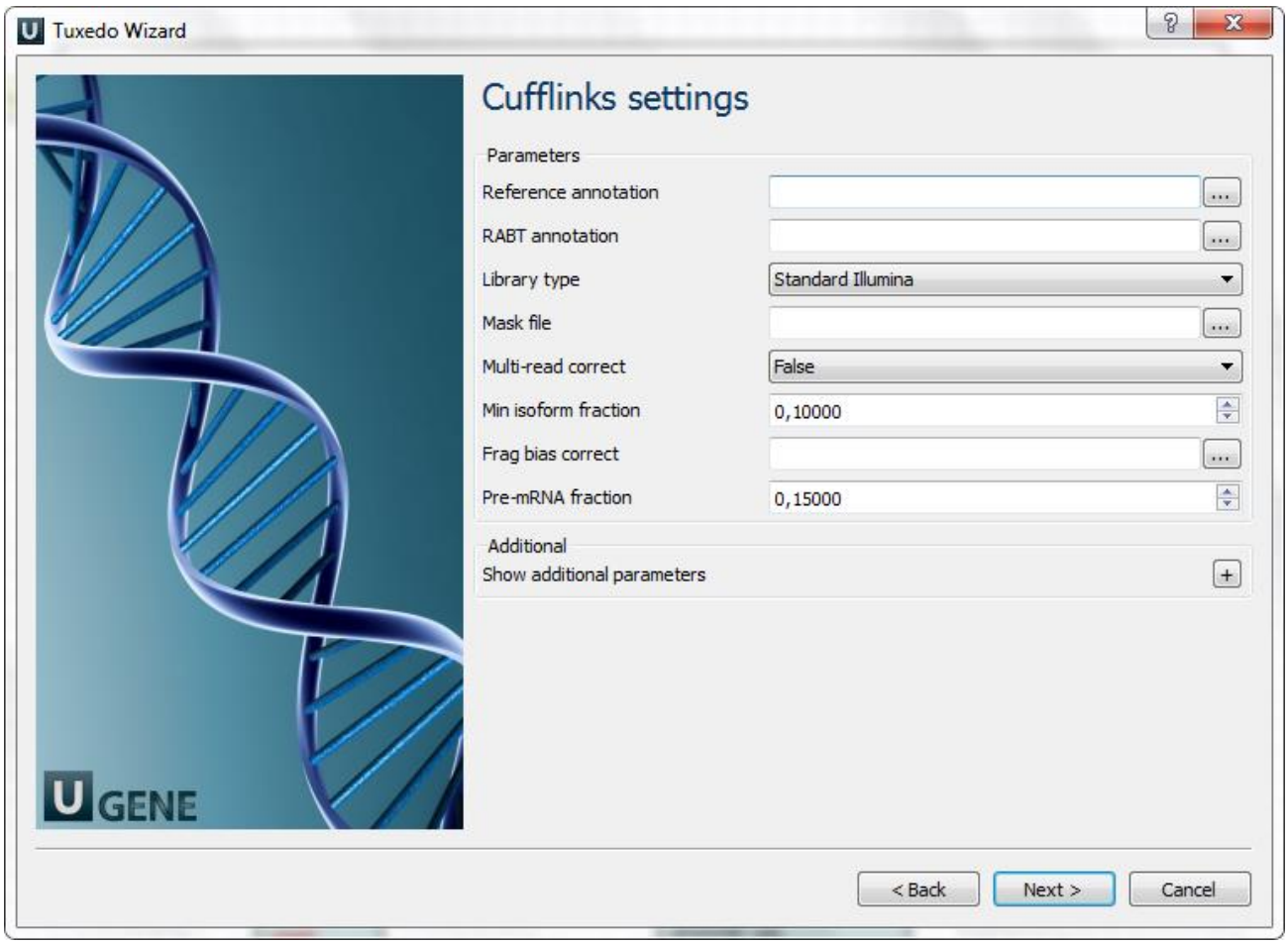

Рис. 2. Пример страницы визарда.

Заключение

В результате работы язык программирования UWL и среда разработки на этом языке приобрели серьезное развитие. UWL стал полноценным языком программирования: в нем появились конструкции языка, стандартная и пользовательская библиотеки вычислительных элементов. Это дает возможность гибкого построения сложных вычислительных схем за счет использования элементов управления потоками данных и интеграции сторонних программ.

В стандартной библиотеке UWL появились вычислительные элементы для анализа данных секвенирования геномов. Добавление же в модель данных новых типов данных, поддержка новых форматов данных и появление средств интеграции сторонних программ дают возможность расширения инструментария пользователя в этой области.

Таким образом, была достигнута цель данной работы – предоставить пользователю расширяемую программную систему по созданию и настройке вычислительных конвейеров для решения задач анализа данных секвенирования геномов. Система обладает единым графическим интерфейсом и за счет визардов настройки схем обеспечивает низкий порог вхождения.

Кроме того, за счет перехода на использование интерфейса базы данных был сделан шаг к распределенным вычислениям, то есть к возможности исполнения вычислений на нескольких компьютерах.

Литература

1. Liu T. Cistrome: an integrative platform for transcriptional regulation studies. // Genome Biol. 2012. 12 (8): R83.
2. Roberts A. Identification of novel transcripts in annotated genomes using RNA-Seq. // Bioinformatics. 2011. doi: 10.1093/bioinformatics/btr355.
3. Li H. The Sequence alignment/map (SAM) format and SAMtools. // Bioinformatics. 2009. 25. – pp. 2078-2079.
4. Документация о UGENE Workflow Designer // UGENE v.1.11.5 documentation – 2013. [Электронный ресурс]. Режим доступа: http://ugene.unipro.ru/documentation/wd_manual/index.html (дата обращения: 29.04.2013).
5. Okonechnikov K., Golosova, O., Fursov, M., the UGENE team. Unipro UGENE: a unified bioinformatics toolkit. // Bioinformatics. 2012. 28 (8). – pp. 1166-1167.
6. Грехов Г.А., Скопин И.Н. Языковые средства организации вычислений в области биоинформатики // Системная информатика. 2013. № 1 (В печати).
7. Johnston W.M., Hanna J.R.P., Millar R.J. Advances in dataflow programming languages. // ACM Computing Surveys (CSUR). 2004. 36 (1). – pp. 1-34.
8. Nielsen, J. Usability Engineering. Morgan Kaufmann Publishers, 1994.
9. NI LabVIEW – Improving the Productivity of Engeneers and Scientists // National Instruments Corporation, U.S., 2013. [Электронный ресурс]. Режим доступа: http://ni.com/labview (дата обращения: 29.04.2013).
10. Agilent VEE Pro 9.3 // Agilent Technologies, California U.S., 2013. [Электронный ресурс]. Режим доступа: http://www.agilent.com/find/vee (дата обращения: 29.04.2013).
11. Hull D. Taverna: a tool for building and running workflows of services. // Nucleic Acids Research. 2006. 34. – pp. 729-732.
12. Jeremy G. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. // Genome Biology. 2010. 11: R86.
13. Шлее М. Qt4: Профессиональное программирование на C++. СПб. 2007. БХВ-Петербург. 854 стр.
14. Kleppe A. Software Language Engineering: Creating Domain-Specific Languages Using Metamodels. Addison-Wesley Professional. 2008 – 207 p.

15. Макконнелл С. Совершенный код. СПб. 2007. Питер. 896 стр.
16. Haldar S. Inside sqlite. O’Reilly. 2007. 76 p.

Приложение А. Разработка визардов на языке UWL

В данном приложении представлен синтаксис описания визардов настройки схем на языке UWL.

Описание визарда содержится внутри блока wizard {…} в разделе метаинформации схемы (блок .meta {…}). Блок визарда состоит из описания его параметров и страниц:

```
wizard {
    name : <wizard-name>;
    page {
        <page-parameters-1>
    }
    ...
    page {
        <page-parameters-n>
    }
}
```

<wizard-name> – имя визарда, которое отображается в заголовке диалогового окна.

Блоки «page» содержат описания страниц визарда. Каждая страница имеет параметры:

1) id – уникальный идентификатор страницы.
2) next – идентификатор страницы, которая следует после текущей. Если этот параметр не указан, то данная страница последняя. Последней страницей является ровно одна из страниц. Визарды могут быть только последовательными, то есть без циклов и ветвлений. Все эти ограничения проверяются при открытии файла схемы.
3) title – заголовок, который отображается в верхней части страницы. Необязательный параметр.
4) template – идентификатор шаблона страницы. Страница виздарда может иметь шаблон визуализации. От шаблона зависит остальной набор параметров. Необязательный параметр (если не указан, то используется шаблон «default»).

Страница, созданная по шаблону «default», состоит из двух частей: левая часть содержит стандартную картинку, правая часть содержит описания параметров схемы (блок parameters-area {…}). Общий вид описания страницы визарда следующий:

```
page {
        id : <page-id>;
        next : <next-page-id>;
        title : <title>;
        template : default;
        parameters-area {
            <parameters-list>
        }
    }
```

Для того чтобы пользователь мог настроить какой-либо параметр схемы в визарде нужно создать соответствующий виджет [13] настройки этого параметра. Внутри блока parameters-area {…} описывается то, какие параметры можно настроить на этой странице, какие виджеты для этого использовать, и как эти виджеты форматированы. Для каждой записи в списке <parameters-list> на странице визарда будет создан виджет. Порядок расположения виджетов вертикальный.

Типы виджетов и параметры, которые можно использовать в parameters-area:

1) Блок <element-id>.<parameter-id> {…} – описания виджета настройки параметра какого-либо элемента схемы. Здесь <element-id> и <parameter-id> – это идентификаторы вычислительного элемента и его параметра соответственно. Для такого блока будет создан виджет, содержащий лейбл с названием параметра и виджет настройки параметра. Текст лейбла задается с помощью необязательного параметра «label». Если он не указан, то лейбл будет содержать стандартное имя параметра.

2) Блок group {…} - группа других виджетов. Для такого блока будет создана рамка с заголовком, внутри которой могут содержаться другие виджеты.

3) label-size – размер (в пикселях) внутренних лейблов у виджетов параметров. Как было описано выше, виджеты настройки параметров имеют лейблы. Параметр «label-size» нужен, чтобы задать этим лейблам одинаковый размер внутри области «parameters-area». Необязательный параметр.

Группа других виджетов (блок group {…}) – это виджет, который группирует другие виджеты внутри себя (в том числе и другие группы). По существу, это расширенный блок описания параметров («parameters-area»). Группа виджетов имеет следующие параметры:

1) title – заголовка группы (по умолчанию, пустой). Необязательный параметр.
2) label-size – нужен для того же, что и одноименный параметр блока «parameters-area».
3) type – задает тип группы: hideable или default. Для типа «default» создается обычный виджет группы, а в случае «hideable» в него добавлена кнопка для сворачивания (и разворачивания) содержимого группы. Последнее нужно для создания групп необязательных параметров. hideable-группы появляются свернутыми, и затем пользователь может развернуть их. Необязательный параметр.

Пример описания визарда:

```
wizard {
    name : "Example wizard";
    page {
        id : 1;
        next : 2;
        title : "Page 1";
        parameters-area {
            group {
                title : "Input datasets";
                read-sequence.url-in {
                    label : "Input file";
                }
            }
        }
    }
    page {
        id : 2;
        parameters-area {
            write-sequence.url-out {
                label : "Output file";
            }
            group {
                type : hideable;
                label-size : 150;
                read-sequence.merge-gap {}
                write-sequence.document-format{}
            }
        }
    }
}
```