5. Дунаева К.В., Якушев Н.М. Анализ проблем организации совместной работы в проектных компаниях Удмуртии // Выставка инноваций - 2022 (весенняя сессия) Сборник материалов XXXIII Республиканской выставки-сессии студенческих инновационных проектов. - Ижевск: Издательство УИР ИжГТУ имени М. Т. Калашникова, 2022. - 118-125 с.

6. Симченко О.Л., Семенова А.Д., Чазов Е.Л. Совершенствование управления инвестиционным портфелем предприятий нефтегазового комплекса // "Фотинские чтения - 2021" (осеннее собрание) Сборник материалов VIII Международной научно-практической конференции, приуроченной к 70-летию ИМИ - ИжГТУ. - Ижевск: Ижевский государственный технический университет имени М.Т. Калашникова, 2022. - 216-221 с.

7. Симченко О.Л., Чазов Е.Л., Сунцов А.С., Губкина А.Д. Мониторинг системы типового проектирования компании // Вестник Челябинского Государственного университета. - 2020. - №6 (440). - 127-133 с.

8. Романова О.А., Чененовa Р.И., Коновалова Н.В., Вагин Г.Т., Ченчевич С.Г. Теоретико-методологические проблемы формирования институциональной матрицы научно- технологического развития региона // Экономика региона. - 2005. - №1 (1). - 100-113 с.

9. Федеральный закон от 23 августа 1996 г. №127 – ФЗ «О науке и государственной научно-технической политике» //СЗ РФ от 26 августа 1996 г. №35. - 4137 с.

10. Гарнер Даниел Роберт Оуэн Конвей. Привлечение капитала / пер. с англ. – М.: «Джон Уайли энд Санэ», 1995. - 464 с.

11. Дежина И.Г. Оценка мер государственной политики России в области науки // ЭКО. 2012. № 2. - 108-120 с.

12. Валовые внутренние расходы на НИОКР // Данные ОЭСР: [Электронный ресурс]. URL: https://data.oecd.org/rd/gross-domestic-spending-on-r-d.htm. (Дата обращения: 17.10.2022).

13. Исследователи // Данные ОЭСР: [Электронный ресурс]. URL: https://data.oecd.org/-rd/researchers.htm#indicator-chart. (Дата обращения: 25.10.2022).

14. Наука, инновации и технологии // Росстат: [Электронный ресурс]. URL: https://rosstat.gov.ru/statistics/science. (Дата обращения: 05.10.2022).

УДК 004.82

ПОДХОД К ПОСТРОЕНИЮ ОЦЕНКИ КАЧЕСТВА ОБСЛУЖИВАНИЯ ВЕБ-СЕРВИСА

В. И. Хведчук

К.т.н., доцент, доцент кафедры вычислительных машин и систем
УО «Брестский государственный технический университет»,
Брест, Беларусь, e-mail : liddan@mail.ru

Реферат

Предлагается интегрированный подход к прогнозированию QoS, который объединяет моделирование многомерных данных QoS с помощью концепций тензора, и позволяет рекомендовать эффективные веб-сервисы для мобильных клиентов с помощью алгоритмов тензорной декомпозиции и оптимизации реконструкции. Большинство современных методов прогнозирования QoS используют характеристики QoS для одного конкретного измерения, например, времени или местоположения, и не используют структурные взаимосвязи между многомерными данными QoS. Устраняются ограничения на точность и масштабируемость, на базе контекстно-зависимого прогнозирования надежности.

Для реализации контекстного описания необходимы элементы семантики. В качестве основы выбрана семантика HAL/S. Показана связь описания процессов с декларативным языком, приведены основные элементы языка. Назначением последнего является описание зависимостей процессов ОС. Предложены архитектура АРМ моделирования системы, приведена возможная архитектура классов реализации декларативного языка. Приведена модель параллельного выполнения логического программирования.

Ключевые слова: веб-сервис, контекстная зависимость, прогнозирование, надежность, семантика, декларативный язык, процесс ОС

AN APPROACH TO BUILDING A WEB SERVICE QUALITY ASSESSMENT

V. I. Khvedtchuk

Abstract

An integrated approach to QoS forecasting is proposed, which combines the modeling of multidimensional QoS data using tensor concepts, and allows us to recommend effective web services for mobile clients using tensor decomposition and reconstruction optimization algorithms. Most modern QoS forecasting methods use QoS characteristics for one specific measurement, such as time or location, and do not use structural relationships between multidimensional QoS data. The limitations on accuracy and scalability are eliminated, based on context-dependent reliability forecasting.

To implement a contextual description, semantics elements are needed. The semantics of HAL/S is chosen as the basis. The connection of the process description with the declarative language is shown, the main elements of the language are given. The purpose of the latter is to describe the dependencies of the OS processes. The architecture of the automated control system modeling is proposed, the possible architecture of classes for implementing a declarative language is given. A model of parallel execution of logic programming is given.problem of designing of the computer and in modern conditions is actual, not looking on development of architecture Intel, etc. Overcoming break between the equipment and program maintenance is still necessary.

Keywords: web service, contextual dependency, forecasting, reliability, semantics, declarative language, OS process.

Введение

Достижения в области технологий Интернета позволили клиентам вебсервисов поддерживать свои сеансы обслуживания в рабочем состоянии, пока они находятся в движении. Услуги, потребляемые мобильным клиентом, могут отличаться с течением времени из-за изменения местоположения клиента, для анализа отношений потребления услуг необходима многомерная простран-

ственно-временная модель. Необходимо также прогнозировать неизвестные значения качества обслуживания (QoS), принимая во внимание время и местоположение запроса услуги целевого клиента, например, выполняя прогноз с помощью набора многомерных показателей QoS. Большинство современных методов прогнозирования QoS используют характеристики QoS для одного конкретного измерения, например, времени или местоположения, и не используют структурные взаимосвязи между многомерными данными QoS. Предлагается интегрированный подход к прогнозированию QoS, который объединяет моделирование многомерных данных QoS с помощью концепций тензора, основанных на мультилинейной алгебре, и позволяет рекомендовать эффективные веб-сервисы для мобильных клиентов с помощью алгоритмов тензорной декомпозиции и оптимизации реконструкции. Это необходимо для формирования рекомендаций для web на базе предсказания показателей QoS, а также обеспечения вопросов конфиденциадьности, Для формирования описания функционирования сетевого процессора используется семантическое описание его архитектуры на базе декларативного языка.

1. Модель

Формально можно собрать 3-мерную матрицу [1]

$$R \in R \qquad M \times N \times T$$

В данную матрицу записываются данные о надежности для M пользователей, N служб и T временных срезов. Ru,s,t = r(u, s, t), когда наблюдается значение надежности r(u, s, t) вызовов inv(u, s, t); в противном случае мы устанавливаем Ru,s,t = 0 как неизвестную запись. Матрица R на практике очень разрежена с большим количеством неизвестных элементов.

Цель прогнозирования надежности состоит в том, чтобы предсказать эти неизвестные записи, посредством чего надежность текущего вызова может быть дополнительно предсказанным. Построение автономной модели состоит из трех этапов: идентификация контекста, агрегирование данных в зависимости от контекста и факторизация матрицы в зависимости от контекста.

1) Идентификация контекста. Чтобы охарактеризовать и идентифицировать различные контекстные условия, используется метод кластеризации k-средних, чтобы сгруппировать данные о надежности R с T временными срезами в C кластеров, где каждый кластер представляет определенный контекст, а разные временные срезы, сгруппированные в один кластер, принадлежат одному и тому же контексту. Чтобы достичь этого, наблюдаемые данные о надежности между M пользователями и N службами на каждом временном отрезке t могут быть построены как вектор признаков для кластеризации k-средних. Однако из-за разреженной природы R векторы признаков стали бы многомерными и разреженными, что еще больше приводит к плохой производительности кластеризации. Определяется вектор признаков f(t) для временного отрезка t, используя среднее значение надежности каждой службы. Используя эти векторы признаков, мы выполняем кластеризацию данных и получаем C различными контекстными условиями.

2) Агрегирование данных в зависимости от контекста. Различные временные срезы могут быть сгруппированы в каждом контексте. Чтобы решить проблему разреженности данных, мы предлагаем объединить данные разных временных срезов в одном контексте. Таким образом, может быть получена матрица агрегированных данных

$$R \in R \qquad M \times N \times C$$

где каждая запись Ru,s,c обозначает среднее значение надежности между пользователем u и службами s в контексте c. В частности, Ru,s,c = 0 указывает на то, что надежность для вызовов inv(u, s, t), выполняемых в контексте c, неизвестна. Наблюдаемые данные о надежности, например, четырех временных срезов могут быть объединены, опять же к примеру, в два контекста (т.е. контекст c1 и c2), и, таким образом, агрегированные данные становятся намного более плотными.

3) Факторизация матрицы в зависимости от контекста. Проблема прогнозирования надежности с учетом контекста заключается в прогнозировании неизвестных записей (где Ru,s,c = 0) агрегированных данных R. Это может быть смоделировано как совместная задача фильтрации (CF), целью которой является восстановление полной матрицы из небольшого числа наблюдаемых записей. Принимая, в качестве примера, в агрегированной матрице для контекста c1, может быть четыре записи и пять неизвестных записей для прогнозирования (например, R(u1,s1,c1). Матричная факторизация (MF) [2] - это классическая модель CF, которая допускает аппроксимацию матрицы низкого ранга. В отличие от обычной факторизации матрицы, имеется 3-мерную матрицу данных о надежности $R \in R$ [$M \times N \times C$], включающую одну 2-мерную матрицу данных M-by-N R(c) в каждом контексте c ($1 \leq c \leq C$), где ее запись R(uc,)s = Ru,s,c. В такой ситуации возможна факторизация матрицы, зависящая от контекста.

Формально факторизация матрицы данных R(c)∈R [M×N] заключается в отображении как пользователей, так и сервисов в d-мерное пространство скрытых факторов, поэтому значения R (c) могут быть записаны как внутренние

2. Тензорные элементы

Атрибуты QoS, такие как время отклика, пропускная способность и надежность, широко используются для оценки нефункциональных аспектов вебсервисов на основе QoS [3]. Для решения проблемы прогнозирования QoS вебсервисов используется метод совместной фильтрации (CF).Он используется для прогнозирования рейтинга в рекомендательных системах, и состоит из двух типов подходов, основанных на соседстве, и, основанных на моделях. Основанный на соседстве может быть использован для прогнозирования качества обслуживания (или надежности). Модельный подход учитывает только факторы, зависящие от конкретного пользователя и сервиса, что приводит к низкой точности прогнозирования. Дополнительно включается временная информация в модели и используется тензорная факторизация для прогнозирования QoS с учетом времени. Но тензорная факторизация страдает от проблемы масштабируемости и недостаточно эффективна для оперативного прогнозирования надежности.

Используются исторические данные от пользователей для прогнозирования надежности и моделируют это как проблему совместной фильтрации. Предлагается гибридный подход, основанный на соседстве, который сочетает в себе два традиционных подхода к CF: CF на основе пользователя (UPCC) и CF на основе элементов (IPCC). Дополнительно расширяется основанный на модели подход, на базе матричной факторизации (PMF).

Эти модели учитывают только параметры, зависящие от конкретного пользователя и сервиса. Включаются также специфические для окружающей среды параметры для прогнозирования надежности. Этот подход обеспечивает масштабируемость за счет кластеризации данных о надежности в соответствии с параметрами, специфичными для конкретного пользователя, службы и среды, но жертвует точностью прогнозирования (которая хуже, чем PMF). Устраняются эти ограничения на точность и масштабируемость, на базе контекстнозависимого прогнозирования надежности. Для реализации контекстного описания необходимы элементы семантики. В качестве основы можно выбрать семантику HAL/S.

3. Семантика HAL/S

На сегодняшний день имеются мощные средства описания и моделирования СБИС, выполненные на базе VHDL или других аналогичных языковых средств. Доступны в свободном доступе в Internet и модели современных семейств аппаратуры, выполненные с их использованием. Для наиболее сложных предлагаются, также, и соответствующие средства разработки (kit), и программные инструментальные средства. Вместе с тем, имеется необходимость реализации отдельных функций операционных систем (ОС) в аппаратуре. При этом, соответствующих средств моделирования не выявлено. Поэтому, была поставлена задача разработки необходимых средств описания архитектуры микропроцессоров с учетом поддержки ОС. Такого рода устройства уже предлагаются к использованию. Так в 1985 году была разработана микросхема Intel 80150, реализующая базовые примитивы ОС. В последующем эта возможность была частично включена в процессоры типа Pentium.

Необходимость определения процессов ОС присутствует, прежде всего, в антивирусных пакетах. Описание возможно на различных уровнях, включая процессы ядра ОС. При этом возможно замедление основных операций на пользовательском уровне, выполняемых на современных быстродействующих процессорах до уровня 16-разрядных процессоров с тактовой частотой 4 МГц.

Еще одним направлением использования средств описания архитектуры является описание HAL – Hardware Abstraction Layer. Одним из его приложений является определение элементов драйвера. Благодаря последовательному процессу по их стандартизации и модификации выполняется реальная поддержка операционной системы. В наибольшей степени она осуществлена для ОС Windows, в меньшей степени Linux, что и отразилось на их реальной распространенности.

Также традиционным направлением, где используется описание архитектуры микропроцессора является задача компиляции, и обратная ей задача декомпиляции.

Задача проектирования ЭВМ и в современных условиях является актуальной, не смотря на развитие архитектур Intel и др. По-прежнему необходимо преодоления разрыва между аппаратурой и программным обеспечением [3].

HAL/S [4,5] - универсальный язык программирования в реальном масштабе времени несколько подобный АДА. Его главные приложения - встроенные системы реального времени.

3.1. HAL/S

HAL/S - предшественник АДА, разработанный в начале 1970-ых компанией Intermetrics, подготовившей один из двух заключительных вариантов языка АДА. Язык включает:

а) законченный набор возможностей управления в реальном масштабе времени параллельными задачами, включая определение задачи, использование приоритетов, синхронизацию по времени или событиям, отмена задачи, основанная на временных интервалах или событиях, критические разделы с блокировкой общих данных, и ожиданием освобождений;

б) механизмы обработки особых ситуаций, включая определение программ обработчиков особых ситуаций и исключений;

в) встроенные типы данных, включая действительный, целочисленный, векторный, матричный, массив, запись, указатель, bitstring, символьная строка, и событие;

г) определение создания подмассивов, включая описание слоев и произвольных компонент;

д) примитивные операции и выражения, определенные для операндов массива, включая изменение и преобразования типов;

е) "повторно используемые" процедуры (общедоступные одновременно многим задачам) и исключительные процедуры (монопольный доступ одной задачи);

ж) много разных особенностей: макроопределение, in-line функции, примитивы ввода-вывода, управление памятью, распределении представлений и хранения и т.д.

Основное применение HAL/S использует встроенный XPL генератор промежуточного кода HALMAT.

3.2. Семантика Н-графа

Формальное семантическое определение HAL/S использует метод семантики Н-графа**.** Формальная модель представляет абстрактное выполнение языка. Определение имеет две части – трансляция и выполнение программ.

Выполнение оформлено в терминах абстрактной машины Н-графа, используя понятия состояния и перехода между состояниями. Состояния представлены как Н-графы, и представляются иерархиями направленных графов, формирующих различные структуры данных и команд в ходе выполнения программы. Класс возможных структур состояний определен формальной грамматикой Н-графа, в которой продукции определяют различные типы структур данных и кода, используемые в модели. Переходы состояний определены наборами Н-графа, определяющими возможное местное преобразование в состоянии Н-графа в ходе выполнения, и функцией перехода, которая определяет следующее состояние. Преобразование определяет примитивные действия абстрактной машины, а функция перехода представляет цикл интерпретации машины.

Трансляция смоделирована как машина Н-графа, обычно с двумя основными переходами, соответствующими (1**)** парсингу, используя контекстно-свободную спецификацию, и (2) статической проверке типа, операции перегрузки, и других "семантических действий". Первый шаг представлен грамматикой пары, которая определяет перевод, соединяя продукции в BNF грамматике, определяющей синтаксис с продукциями в грамматике Н-графа, определяющей промежуточный код.

3.3. Семантическое определение HAL/S

Законченное формальное семантическое определение включает все части языка, за исключением некоторых или строго зависящих от выполнения элементов нижнего уровня. В частности, включает все особенности в реальном масштабе времени, обработку особых ситуаций, задачи, программы, процедуры и функции, структуры данных, и другие части высокого уровня языка. Определение трансляции включает только трансляцию к начальному состоянию времени выполнения (использование парной грамматики, чтобы отобразить каждую синтаксическую конструкцию в код и/или данные времени выполнения). Не моделируется статический контроль соответствия типов и другой семантический анализ компилятора. В качестве основных выделено моделирование следующих задач:

1) массивы в качестве индексов.
2) видимость временных переменных.
3) взаимное исключение из разделяемых данных.
4**)** управление задачами в реальном масштабе времени.
5) сроки жизни переменных Событие.
6) обработка особых ситуаций.
7) спецификации параметров.
8) индексы символьных строки.
9) ввод-вывод и позиционирование файла.

Использование семантического определения HAL/S (опыты с Н-графом) позволяет:

а) построение формального семантического определения, снятие неопределенностей, неоднозначностей и противоречий спецификации языка;

б) обеспечение абстрактной модели выполнения, определение упрощенной разработки выполнения, поддержку на ранней стадии реализации выполнения, уточненный анализ крупномасштабных проблем.

Преимущество (б) обычно не реализуется через формальные семантические определения, так как другие методы определения типа денотационной и аксиоматической семантики обычно не имеют прямых моделей выполнения. Развитие абстрактной модели выполнения может быть применено для решения задач времени выполнения.

4. Язык описания процессов

Для возможности хотя бы частичной автоматизации связи ресурсов и процессов предлагается описывать процессы в ОС следующим набором:

(Func, CPU, MEM, HM),

где Func – набор функций процесса;

CPU –ресурс центрального процессора;

MEM – ресурс оперативной памяти

HM – ресурс внешней памяти.

Для определения информационной зависимости процессов в ОС используется отношение R: $P_i \, R \, P_j$, если для выполнения хотя бы одной из функций $Func_{i,k}$ процесса P_i требуется выполнение хотя бы одной из функций $Func_{j,s}$ процесса P_j.

При этом нижним сечением $R^-(P_i)$ этого отношения будет являться множество всех процессов P_m, зависящих от P_i, и открытых для изменения. Положим $A = R^-(P_i)$, тогда $R_i \, A \, M_j$, если хотя бы одному отношению из R_i заданы права доступа процессов M_j.

Для описания и анализа иерархии процессов предлагается использовать язык (R, M, A) описания информационной зависимости R. В этом языке задаются и отношения разграничений процессов M, а также отношения возможности изменений A. Описание имеет декларативный характер.

Задача компиляции рассматривается как набор правил (теорем) для построения исполняемого кода. Задача декомпиляции менее разработана, и, в силу этого, не имеет столь мощной автоматизации, как задача компиляции. Тем не менее, как задача, обратная к задаче компиляции, она также представима набором декларативных правил.

Для поддержки декларативного характера языка (R, M, A), а также инструментов декомпиляции, возможно добавление на аппаратном уровне средств выполнения языка логического программирования.

Архитектура АРМ моделирования процессов на микропроцессорах представлена на рис. 1. Возможная архитектура классов для поддержки декларативного описания [4,5] приведена на рис.2.

Рис. 1. Архитектура средств системы описания процессов

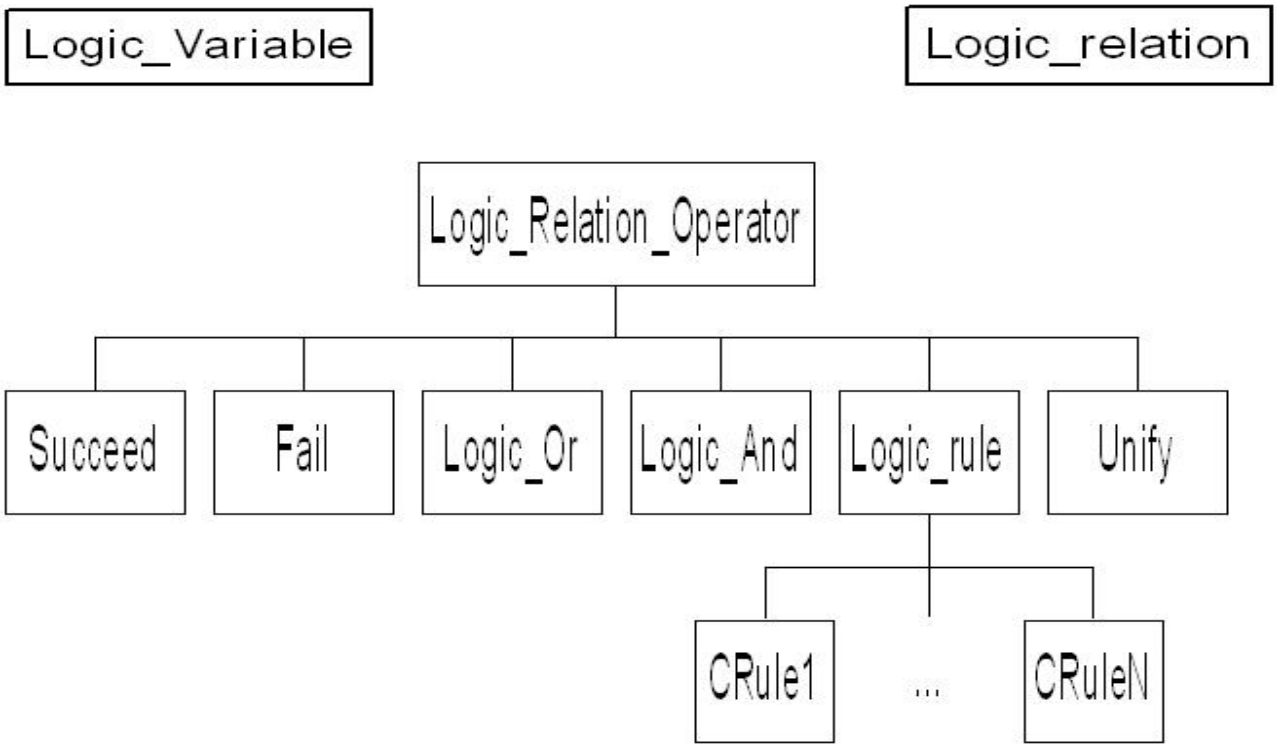

Рис. 2. Архитектура классов для декларативного описания

5. Элементы структуры средств АРМ моделирования процессов

В обобщенную структуру предлагаемого учебного антивирусного комплекта, в числе прочих элементов, входят: язык описания иерархии процессов, средства декомпиляции, аппаратные средства поддержки декларативного языка.

5.1 Язык описания иерархии процессов (R, M, A)

Используются следующие основные группы функций примитивов процессов:

- управление заданиями и задачами;
- управление прерываниями;
- управление памятью;
- управление сообщениями между задачами;
- управление синхронизацией между процессами;
- управление окружением.

Примерами функций группы управления заданиями и задачами являются:

- создание задачи;
- удалить задачу;
- приостановить задачу;
- снять задачу.

Функции $Func_{j,s}$ процесса P_j описываются набором своих примитивов $\{N(P_j)\}$. Права доступа назначаются на процесс или на ресурсы.

Введение примитивов необходимо для описания основных элементов системы защиты. Пример примитива:

$P_j(CPU_j, ME_j, HM_j) :- Func_j\{N(P_j)\}$.

Отношение *R* задается в виде отношений: $P_i(CPU_i, ME_i, HM_i) :- P_j(CPU_j, ME_j, HM_j)$.

Множество A задается как $A(cl(P_i)) :- clall(P_i, LP_j)$. Здесь LP_j – список клозов (правил), которые имеют в заголовке одинаковое имя предиката P_j.

Предикат clall является истинным, если истинны все клозы, имеющие в заголовке предикат P_j. Аргумент cl является истинным, если является истинным предикат его аргумента

5.2 Средства декомпиляции

Возможно использование, наряду со стандартными средствами декомпиляции, дополнительных средств. Теоремы (правила) компиляции представимы в виде

$T_z(LA) :- cl(LT_z)$.

Здесь T_z – один из предикатов, формирующих выходной язык;

LA – список его аргументов;

LT_z – список клозов (правил), которые имеют в заголовке одинаковое имя предиката T_z.

По аналогии с построением компилятора, возможно построение правил декомпиляции. Предполагается усиление возможностей декомпилирующих средств и инструментов (R, M, A) за счет их поддержки аппаратурными средствами.

5.3 Модель для параллельного логического вывода.

При логическом выводе имеет место ориентированный граф Q, в котором

$$P_j^i(P_e)\ Q\ P_e = Y_j^i(P_e)$$

при условии необходимости унификации $P_j^i(P_e)$ из тела предиката P_e. Здесь P_ζ - предикат запроса. P_j - j-ый предикат, одноименный P_ζ. $P_j^i(P_e)$ – j-ый предикат, находящийся на i-ом уровне унификации (число унификаций после унификации предикатов, одноименных P_ζ) из тела утверждения, в заголовке которого находится предикат P_e, являющийся e-м в теле предиката уровня i-2; $Y_j^i(P_e)$ – унификация уровня i для j – го предиката P_e. Существует возможность параллельного выполнения унификаций одного уровня, а также разных уровней i и j (i < j), но не имеющих путей перехода по дугам, которые исходят из вершин уровня i к вершинам уровня j.

Разработано модифицированное представление Пролог-программы, включающее множества PT, $CT_{\chi i}^l$, $CT_{\chi i}$, $HT_{\chi i}^l$, $BT_{\chi i}^l$, $AT_{\chi i}^q$. По каждому предикату тела (для факта создается пустой предикат тела) создаются следующие вершины: *ORFORWARD* – ссылка на следующий предикат тела; *ORBACK* – ссылка на последнюю для данной вершины доказанную альтернативу (по ИЛИ назад); *ANDBACK* - ссылка на доказываемый предикат тела. При переходе по *ORFORWARD* меняются *ANDFORWARD* и *ORFORWARD*. При переходе по *ANDFORWARD* меняется *ANDFORWARD*. При переходе по *ORBACK* меняется все. При переходе по *ANDBACK* меняется все, кроме *ORBACK*.

При назначении вычислительного элемента для участия в процессе вывода ему назначается предикат тела с 4 вершинами, описанными выше.

Для одноименных переменных используются разделение структур данных, собственно переменная указывается один раз в утверждении.

Множеством *PT* является множество

$PT=\{h_i\}$, $h_i = h_\rho$, $\rho=1,...,N$, $i=1,...,N$, $i \neq \rho$.

Множеством $CT^l_{\chi i}$ является для данного h_i множество

$$CT^i_{\chi i}=\{x_{i_1}, x_{i_2}, ..., x_{i_{\xi_i}},$$
$$b_i^1, b_i^2, ..., b_i^{N_i}, y^1_{i_1}, y^1_{i_2}, ..., y^1_{i_{K_1^i}}, y^2_{i_1}, y^2_{i_2}, ..., y^2_{i_{K_1^i}}, y^{N_i}_{i_1}, y^{N_i}_{i_2}, ..., y^{N_i}_{i_{K_1^i}}\}$$

где $h_i = h_p$, $p=1,...,N$, $i=1,...,N$, $i \neq p$, $j=1,...,G_i$,

x_{i_γ} - аргументы l-го с начала программы клоза с заголовком h_i,

$b_i^1, b_i^2, ..., b_i^{N_i}$- предикаты тела l-го с начала программы клоза с заголовком h_i,

$y^1_{i_1}, y^1_{i_2}, ..., y^1_{i_{K_1^i}}$ - аргументы предиката b_i^1 тела l-го с начала программы клоза с заголовком h_i,

$y^2_{i_1}, y^2_{i_2}, ..., y^2_{i_{K_1^i}}$ - аргументы предиката b_i^2 тела l-го с начала программы клоза с заголовком h_i,

$y^{N_i}_{i_1}, y^{N_i}_{i_2}, ..., y^{N_i}_{i_{K_1^i}}$ - аргументы предиката $b_i^{N_i}$ тела l-го с начала программы клоза с заголовком h_i.

Множеством $CT_{\chi i}$ является множество $CT_{\chi i} = \{ CT^l_{\chi i}\}$, $l=1,...N_{\chi_i}$, где N_{χ_i}, - число клозов, имеющих одинаковые h_i.

Множеством $HT_{\chi i}$ для данного h_i ($i=1,...,N$) является множество $HT_{\chi i}=\{x_{i,g}\}$, $i=1,...,N$, $g=1,...,G_i$. Данные элементы входят составной частью в одно из множеств $CT^l_{\chi i}$.

Множеством $BT_{\chi i}$ для данного h_i ($i=1,...,N$) является множество $BT_{\chi i}=\{b_i^r\}$, $r=1,...,N_i$. Данные элементы входят составной частью в одно из множеств $CT^l_{\chi i}$.

Множеством $AT_{b_i^q}$ для данного b_i^q ($i=1,...,N$, $q=1,...,N_i$) является множество $AT_{b_i^q}=\{y^r_{i,t}\}$, $t=1,...,K_q$. K_q. – число аргументов b_i^q.

Элемент $ux_{n,g}$ или $uy^n_{g,t}$ является связанным, если имеется в некотором множестве HT_{χ_m} или $AT_{b_v^\delta}$, элемент deref ($x_{n,g}$), deref ($y^n_{g,t}$), deref ($ux_{n,g}$), deref ($uy^n_{g,t}$), изменяющийся точно также как и $ux_{n,g}$ или $uy^n_{g,t}$. В качестве deref ($x_{n,g}$) может выступать элемент множества $HT_{\chi i}$, элемент множества $AT_{b_i^q}$, элемент $ux_{n,f}$, элемент $uy^q_{i,t}$, deref ($x_{i,f}$), deref($y^q_{i,t}$), deref ($ux_{i,f}$), deref ($uy^q_{i,t}$). При рассмотрении deref ($x_{n,g}$), deref ($y^n_{g,t}$) или deref ($ux_{n,g}$), deref ($uy^n_{g,t}$) устанавливается флаг DEREF.

В случае связанного, один аргумент является элементом нескольких множеств. Выражение – является элементом (аргументом) некоторого множества – тождественно выражению – является связанным с некоторым элементом (аргументом) некоторого множества. Аргумент функтора также является множеством.

Для реализации алгоритма логического вывода с элементами распараллеливания выделены операции:

поиск *v*-го элемента множества $BT_{\chi i}$ (в случае i=N имеем предикаты запроса);

поиск элемента h_L множества PT, такого что $h_L = b_i^n$, где $b_i^n \in HT_{\chi i}$;

поиск $w-$ го элемента $CT_{\chi_L}^w$ множества CT_{χ_L};

поиск $z-$ го элемента HT_{χ_L};

сравнение $z-$ го элемента HT_{χ_L} и $z-$ го элемента $AT_{b_i^q}$.

При переходе *ORBACK* (b_i^v) возможны следующие случаи унификации аргументов:

1) аргумент $y_{i,\zeta}^v$ имеет элемент в отношении D-PLACE-UNIFY;

2) аргумент $y_{i,\zeta}^v$ имеет элемент в отношении PLACE-UNIFY, до вхождения в клоз для доказательства $b_i^v =$ *ORBACK* (b_i^v).

В первом случае, при возврате рассматривается аргумент

$$y_{i,\zeta}^v = \mathrm{D-PLACE-UNIFY}(y_{i,\zeta}^v)$$

который был уже унифицирован до вхождения в b_i^v для доказательства и находящийся в отношении D-PLACE-UNIFY с $y_{i,\zeta}^v$, во втором

$$y_{i,\zeta}^v = \mathrm{PLACE-UNIFY}(y_{i,\zeta}^v)$$

это элемент $y_{i,\zeta}^v$, который был до вхождения в b_i^v для доказательства и при этом не был унифицирован до вхождения в b_i^v для доказательства).

Задача планирования параллельного логического вывода сводится к решению следующих подзадач:

1) определение блоков данных (пулов) для выполнения одним вычислительным элементом (ВЭ);

2) выбор ВЭ для исполнения пула;

3) определение условия начала исполнения пула.

Для их решения используется, как статическое планирование (до начала доказательства запроса), так и динамическое (во время доказательства). Необходимость последнего вызвана невозможностью определения до начала выполнения Пролог-программы хода логического вывода.

Статическое планирование осуществляется при трансляции с Пролога во внутренние структуры данных. Первым начинает доказательство запроса ВЭ с первым сформированным пулом. Последующие ВЭ начинают работать по мере передачи им результатов унификации с ВЭ, приступивших к работе ранее.

Динамическое планирование параллельного логического вывода осуществляется диспетчером на HOST-ЭВМ и диспетчерами на однородных вычислительных элементах (ВЭ). ВЭ соединен с HOST-ЭВМ и другими ВЭ общей ши-

ной. Приоритет захвата динамический и соответствует приоритету назначений на ВЭ. Пул назначенный ранее придает ВЭ более высокий приоритет. Наибольший приоритет закреплен на HOST-ЭВМ. Для доступа к своей памяти HOST-ЭВМ и ВЭ используют свои внутренние шины. При выборе альтернатив используется их расположение в памяти в порядке возрастания адресов. При этом не является обязательным их следование непосредственно друг за другом. Элементы альтернативы также располагаются последовательно друг за другом.

Разработанное модифицированное представление Пролог-программы и алгоритм доказательства запроса позволили сформулировать требования к структурам представления данных для параллельного логического вывода, уменьшить число шагов унификации для двух аргументов. На основании алгоритмов динамического планирования определены основные функции для обеспечения протокола интеллектуального интерфейса. Предложенные алгоритмы статического и динамического планирования позволяют выделять области кода для независимого исполнения, что дает возможность параллельного исполнения различных участков Пролог-программы, по объему превышающей память одного ВЭ.

5.4 Аппаратные средства поддержки декларативного языка

Структура средств реализации логического вывода изображена на рис.3. Программа, написанная на декларативном языке (языке логического программирования типа Пролог), преобразуется в списковую структуру. Каждый из анализаторов логического вывода пытается выполнить унификацию запроса с одним из клозов с таким же именем предиката в заголовке, имеющимся в программе. Каждый из анализаторов логического вывода может получить свой клоз. После доказательства всех предикатов из тела запроса (получении пустого клоза в результате вывода) получаем результат истинности запроса.

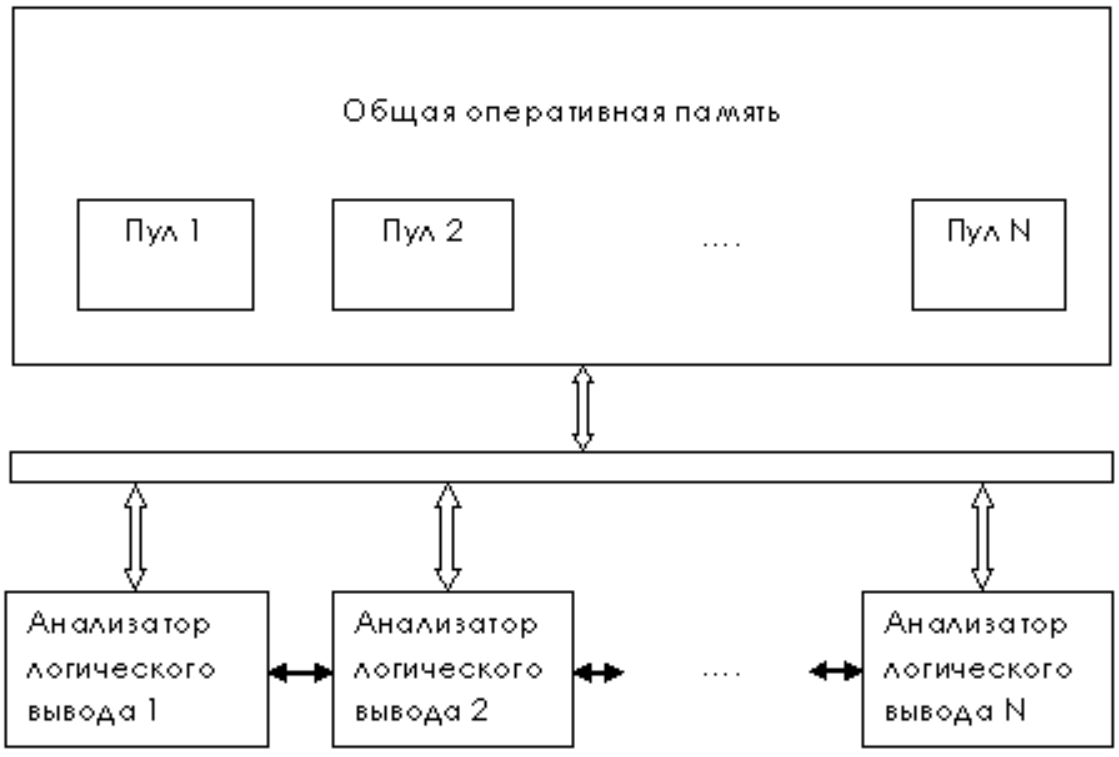

Рис. 3. Устройство управления логическим выводом

При выводе возможна организация в памяти пулов, относящихся к выводу, выполняемому соответствующим анализатором вывода, с целью получения независимых областей памяти, относящихся к данному анализатору.

Заключение

Контекстно-зависимый подход к прогнозированию надежности на базе пользовательской надежности служб "черного ящика" использует исторические данные об использовании пользователей для оценки наблюдаемой надежности сервисов и дальнейшего использования их для построения контекстно-зависимых моделей надежности. Благодаря контекстно-зависимому обучению моделей и прогнозированию можно решить проблему разреженности данных, которая сильно ограничивает существующие модели.

Использование подходов, основанных на данных, является интересным для управления качеством услуг "черного ящика" у пользователя.

Для формирования контекста функционирования “черного яшика” предложен подход к описанию архитектуры ЭВМ. В отличие, от обычно используемых средств, вводится описание процессов операционной системы, средств трансляции. Выделены средства HAL/S как аналог подхода для построения среды описания процессов в архитектуре ЭВМ. Показана связь описания процессов с декларативным языком, приведены основные элементы языка. Назначением последнего является описание зависимостей процессов ОС. Предложены архитектура АРМ моделирования микропроцессорной системы, приведена возможная архитектура классов реализации декларативного языка. Приведена модель параллельного выполнения логического программирования. При выполнении на параллельной аппаратурной реализации возможно линейное возрастание производительности с увеличением числа используемых ВЭ.

Список цитированных источников

1. Z. Zheng and M. R. Lyu. Collaborative reliability prediction of service-oriented systems. In Proc. of the International Conference on Software Engineering (ICSE), pages 35–44, 2010.

2. R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In Proc. of Annual Conference on Neural Information Processing Systems(NIPS), 2007 ACM Press, 2008, pp. 1257-1264.

3. L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. QoS-aware middleware for web services composition. IEEE Trans. Software Eng., 30(5):311–327, 2004.

4. Terence W. Pratt, George D. Maydwell Experience with formal semantic definition of HAL/S Symposium on Compiler Construction Proceedings of the 1982 SIGPLAN symposium on Compiler construction. Boston, Massachusetts, USA P: 327 – 333, 1982 [Электронный ресурс]. Режим доступа: http://ntrs.nasa.gov/archive/nasa/ casi.ntrs.nasa.gov/19890069053_1989069053.pdf

5. Майерс Г. Архитектура современных ЭВМ. В 2-х книгах. – М.:Мир, 1985.