ИПМ им.М.В.Келдыша РАН

Абрау-2024 • Труды конференции

Л.В.Городняя

Выбор решений для языка учебного программирования

Рекомендуемая форма библиографической ссылки

Городняя Л.В. Выбор решений для языка учебного программирования // Научный сервис в сети Интернет: труды XXVI Всероссийской научной конференции (23-25 сентября 2024 г., онлайн). — М.: ИПМ им. М.В.Келдыша, 2024. — С. 57-72.
https://doi.org/10.20948/abrau-2024-3
https://keldysh.ru/abrau/2024/theses/3.pdf

Видеозапись выступления

Выбор решений для языка учебного программирования

Л.В. Городняя[1, 2]

[1] *Институт систем информатики им. А.П. Ершова СО РАН*
[2] *Новосибирский государственный университет*

Аннотация. Статья посвящена выработке решений в проекте тренажёра на базе языка учебного программирования, предназначенного для начального ознакомления с базовыми понятиями взаимодействия процессов и управления вычислениями. На этапе перехода к многопроцессорным архитектурам возрастает актуальность развития особой языково-информационной поддержки введения в программирование. Сколь ни сложен мир параллелизма, системе подготовки программистов предстоит его освоить и создать методику полноценного ознакомления с его не очевидными явлениями. Это достаточная причина для разработки языка учебного программирования, ориентированного на начальное обучение школьников младших и средних классов, а также студентов младших курсов и непрофессионалов, оперированию взаимодействующими процессами и программированию параллельных вычислений. В основу учебного языка СИНХРОН положен опыт кнопочного управления взаимодействием игрушечных роботов (Робик, Logo, Karel, Кумир, Роботландия, Вертун, ПиктоМир и другие), перемещающихся на клетчатой доске. Материал статьи представляет интерес для программистов, студентов и аспирантов, специализирующихся в области системного и теоретического программирования, и для всех тех, кто интересуется проблемами современной информатики, программирования и информационных технологий, особенно проблемами параллельных вычислений, суперкомпьютерами и вообще применением многопроцессорных комплексов.

Ключевые слова: учебное программирование, функциональное программирование, взаимодействие процессов, многопроцессорные конфигурации

Selecting solutions in an educational programming language simulator

L.V.Gorodnyaya[1,2]

[1] *A.P. Ershov Institute of Informatics Systems*
[2] *Novosibirsk State University*

Abstract. The article is devoted to the development of solutions in the project of a simulator for teaching programming, intended for initial familiarization with the basic concepts of process interaction and calculation management. No matter how complex the world of parallelism is, the programmer training system will have to master it and create a methodology for fully familiarizing itself with its non-obvious phenomena. The simulator is based on the experience of controlling the interaction of toy robots moving on a checkered board (Robik, Logo, Karel, KuMir, Robotlandia, PictoMir and others). The article material is of interest to programmers, students and graduate students specializing in the field of system and theoretical programming.

Keywords: educational programming, functional programming, process interaction, multiprocessor configurations

1. Введение. Пришло время изучать информатику и программирование, начиная с мира параллелизма. Компьютеры стали многопроцессорными, поэтому алгоритмы становятся параллельными, программы — многопоточными, очередное действие может начать выполняться до завершения предыдущего, средства управления кроме логических значений используют события и многое другое, что приводит к размежеванию синтаксиса и семантики параллельных вычислений (ПВ) на уровне выражений [1]. Ещё в начале 1960-ых при создании языка APL его автор, Кеннет Айверсон (*Kenneth Iverson)*, утверждал, что истинное программирование — это организация параллельных процессов [2]. И всё это теперь происходит в многоязыковой обстановке — мощность пространства языков программирования (ЯП) давно перевалила за десять тысяч[1]. Следует учесть, что моделей ПВ много больше, чем последовательных вычислений.

Полезно предусматривать развитие учебных моделей и уточнение основных понятий программирования по мере обучения, выделяя

[1] https://web.archive.org/web/20110220044217/http://hopl.murdoch.edu.au/ HOPL:an interactive Roster of Programming Languages, https://www.levenez.com/lang/ Computer Languages History

фрагменты выражений, действий, вычислений и данных, включая процессоры, в соответствии со стремительным развитием элементной базы, многопроцессорных комплексов, сетей и ИТ.

Многие понятия при таких переходах влекут дополнительные понятия, а также расширение конструкций, используемых при определении языка параллельного программирования. Программы ПВ обладают более длительным жизненным циклом и работают в более широком пространстве, чем обычные программы. Поэтому в проекте учебного языка учитывается мультипарадигмальность реальных языков программирования, а также, место функциональных прототипов в подготовке параллельных программ, включая механизм ленивых вычислений и расширяемость языков и систем программирования. Поддерживается формирование навыков редактирования фрагментов программ и протоколов их исполнения, что создаёт основу для понимания трансформационной семантики программ, умения их преобразовывать, выделять компоненты, контролировать синтаксическое подобие и другие механизмы, открывающие путь к владению методами метапрограммирования.

Статья начинается с анализа тенденций в развитии языков и систем программирования и обзора решений, рассмотренных в проекте учебного языка СИНХРОН, далее описаны особенности программ для ознакомления с явлениями параллелизма и в заключении отмечено текущее состояние проекта тренажёра на базе языка СИНХРОН, ориентированного на обучение в форме разработки учащимися своих игр, выглядящих как взаимодействие роботов на клетчатой доске.

2. Общие тенденции

История языков программирования накопила ряд работоспособных идей по эффективному представлению естественного параллелизма при серийной обработке сложных структур данных (APL, Algol-68, SETL, Ada, БАРС, Sisal, Норма, mpC и др.). Для большинства таких ЯВУ характерно включение в семантику языка конкретной модели параллелизма. Так, функциональный язык APL нацеливает на обобщение скалярных операций до обработки однородных векторов произвольной размерности [2]. Algol-68 поддерживает управление процессами в терминах семафоров и критических участков, сохраняя привычки последовательного программирования. SETL сводит параллелизм к математической независимости элементов множества [10]. Язык Ada предлагает механизм «рандеву», требующий для обмена данными одновременное существование процессов. Язык БАРС представляет взаимодействия процессов с помощью синхросетей над комплексами данных и выражений и поддерживает программирование разных дисциплин работы с памятью. Функциональный язык параллельного программирования Sisal

предоставляет ряд средств формирования пространств итераций для параллельного исполнения циклов и для свёртки параллельно полученных значений в общий результат [11]. Язык Норма сосредоточен на проблеме представления сеточных вычислительных алгоритмов на основе непроцедурного описания решаемой задачи [12]. Интересен уникальный язык mpC, нацеленный на сетевое представление многопроцессорных конфигураций при выполнении многопоточных программ, допускающий синхронизацию потоков с помощью барьеров и перераспределение нагрузки процессоров [13]. Производственные инструменты MPI и Open MP связаны с аппаратными моделями параллелизма, что чревато трудоемкостью известной проблемы мобильности программ. Многие идеи организации процессов на уровне операционных систем слишком ограничены механизмами параллельной обработки на базе очередей и векторов [3]. На таблице 1 показана хронологии некоторых решений по представлению параллельных вычислений.

Таблица 1.
Решения по представлению ПВ в некоторых ЯП

Год	ЯП	*Средства представления параллельных вычислений*
1960	**APL**	Приоритет матрицам, их деструктуризация и просачивание скалярных операций до обработки однородных векторов произвольной размерности.
1968	**Algol-68**	Управление процессами в терминах семафоров и критических участков, позволяющих параллелизм сводить к однопроцессорным программам.
1970	**SETL**	Независимость элементов множества.
1979	**Ada**	Синхронизация процессов, обменивающихся данными — «рандеву» .
1984	**Clisp**	Мультизначения и пакеты — пространства имён.
1985	**БАРС**	Комплексы (размеченные множества с кратностью вхождения элементов), сети Петри и синхросети над комплексами данных и выражений, включая процессоры, просачивание, программируемые дисциплины доступа к памяти.
1986	**Sisal**	Базируется на языке Pascal, иерархия SSA-форм (участки однократного присваивания), матрицы, потоки, пространства итераций для параллельного исполнения циклов, просачивание, комплекты результатов выражений .
1991	**Норма**	Сеточные вычислительные алгоритмы с выделением разных схем параллелизма.
1994	**MPI**	Обмен данными между процессами.
1995	**MpC**	Расширение C, многопроцессорные конфигурации, допускающие синхронизацию в терминах сетей и барьеров, поддерживает перераспределение нагрузки процессоров.
1997	**OpenMP**	Распараллеливание программ.
2007	**Clojure**	Указательные переменные, разные дисциплины доступа к памяти и стратегии параллелизма, деструктуризация, синхронизация потоков, события, агенты.

Важно, что при решении одной задачи могут понадобиться решения из разных языков. Сравнение диалектов языка Lisp (Scheme, Clisp, Clojure) показывает, что в таких случаях происходит сначала моделирование недостающих средств на основном языке, затем их неявная поддержка в системах программирования для ряда диалектов и, наконец, явное включение в определение диалекта, обобщающего накопленный опыт. Примерно так в современные ЯП (Common Lisp, Haskell, F#, Kotlin, C#, Java, JavaScript, Scala, Python, Clojure) постепенно проникают понятия, отсутствующие на уровне концепций языка, но проявляющиеся в работе отдельных функций (символы, откладывание, ожидание, обещание, задания, потоки, очереди, мьютексы, семафоры, атомы и др.). Подключаются функции высшего порядка и специальные функции, поддерживающие разметку участков для распараллеливания и безопасного обмена данными. Появляются специальные примитивы, реакции на события, обратные вызовы, пересечение/совмещение возможностей, фильтры, отображения, изменение значений, а также методы и интерфейсы, дополнительные парадигмы и контроль состояний, реактивное программирование, побочные эффекты и асинхронность. На уровне системной поддержки предоставляется манипулирование временем, асинхронное выполнение, фоновый режим и блокировки вычислений, разделение пространств имен, включение иноязычных модулей и специализированных систем с управлением обработкой ошибок и выбором методов, включая параллелизм через создание объектов.

3. Основные решения

Проект реализации тренажёра по учебному программированию СИНХРОН отражает первые шаги процесса обучения, начинающегося оперированием многопоточной программой с приоритетом параллелизму. Проблемой является создание в рамках единой среды и языковой модели полного свода решений, имеющих прецеденты в разных языках и системах программирования.

Оперирование программой. По умолчанию тренажёр содержит интерпретатор, обеспечивающий пошаговое оперирование выполнением программ, сопровождаемым протоколом успешных действий. При выполнении заданий одновременно с решением поставленной задачи в форме взаимодействия потоков должен быть построен контекст, в котором синхронизация потоков корректна, иначе программа выполняется до обнаружения невыполнимого действия и приостанавливается. Подобный подход опробован в языках и системах Робик, Logo, Karel, Кумир, Роботландия, Вертун, ПиктоМир и других. Возможно переключение в режим редактирования программы с использованием протокола, подобно системам Delpfi, Е-пратикум, Рапира.

Приоритет параллелизму. Параллельные вычисления показали себя трудной задачей, требующей особой парадигмы, а новые парадигмы требуют предварительного иллюстративного материала для активизации и целенаправленного формирования скрытых моделей изучаемой парадигмы, отвечающей на вызовы новых проблем. Прецеденты в языках APL, SETL, БАРС, Sisal, mpC, Lisp2D[2] [1, 3, 4].

Мультипарадигмальность. Языки XXI века, как правило, поддерживают основные и фундаментальные парадигмы программирования, что позволяет программировать решения задач разной степени изученности, с разной длительностью жизненного цикла постановки задачи в рамках единой обстановки и получать навыки работы на всех фазах полного жизненного цикла программ как в языках Lisp и Ada. Стихийно сформированные интуитивные модели и сложившиеся в обычной жизни и учёбе навыки обычно успешны при обучении императивно-процедурному и объектно-ориентированному программированию при решении типовых задач. Они не достаточны при обучении функциональному, параллельному и логическому программированию, а также новым парадигмам, возникающим при обнаружении особо сложных, трудно решаемых задач, подход к которым может показать опыт применения тренажёра [5, 6].

Функциональные прототипы параллельных программ. Языки параллельного программирования занимают видное место среди функциональных языков. Программы на таких языках обычно свободны от побочных эффектов и ошибок, непредсказуемо зависящих от реального времени. Это существенно упрощает отладку программ ПВ благодаря выделению автономных фрагментов подобно мультизначениям или пакетам в Clisp [7-9].

Ленивые и опережающие вычисления. Результативность вычислений можно повышать с помощью специально устроенных данных — так называемых «рецептов» и мемоизации. Рецепт предназначен для отложенного или опережающего исполнения фрагмента программы (ленивые вычисления) и представлен как конструкция из функции и контекста для её выполнения, называемая замыканием функции. Мемоизация позволяет исключать дублирование сложных вычислений сохранением полученных результатов. Такая техника доступна в языках Lisp, Setl, Clisp. Выполнение параллельных процессов часто требует независимости порядка вычислений от последовательности представления действий в программе [7, 10].

[2] http://lisp2d.net/rus/ppfs.html Глебов А.Н. Параллельное программирование в функциональном стиле

Расширяемость языков и систем программирования. При определении реализации учебного языка ПВ, могут пригодиться идеи языков Lisp, F#, C#, допускающих динамическую обработку кода программ в рамках парадигмы функционального программирования. Это позволяет обеспечить лаконизм представления программ, облегчить отладку программ, использование инородных модулей и упростить реализацию специальных версий интерпретатора, компилятора и виртуальной машины языка. Решения проблемы расширяемости ЯВУ становятся актуальнее по мере развития средств и методов ПВ, обусловленного высоким темпом прогресса в области элементной базы и информационных технологий, они различаются в системах с препроцессорами или языках Lisp и Setl [4].

Редактирование фрагментов. Кроме простого сцепления текстов при подготовке и отладке программ используются синтаксические макросы, вид параметров которых следует задавать с помощью синтаксического подобия вхождению переменной в заданную строку согласно синтаксису используемого языка. Это можно рассматривать как задачу выбора форм или шаблонов проектирования для представления и организации семейств допустимых параллельных процессов на примере языков Setl и Clisp [14].

Фрагментация программ и синтаксическое подобие. При определении функций кроме обычных переменных используются так называемые «фрагментные», используемые для параметризации схем управления или шаблонов проектирования и наполнения их подходящим содержанием, возможно требующим контроля вида фрагмента. Все вхождения фрагментной переменной в схему должны быть синтаксически эквивалентны её вхождению в строку, задающую вид значения переменной при определении макроса в отличие от типа переменной при определении функции [14, 15].

Трансформационная семантика. Общие решения по обеспечению лаконизма, универсальности, конструктивности и расширяемости приводят к трансформационной семантике ЯВУ, определяющей семейства допустимых преобразований программ, что является естественным полигоном для метапрограмирования, поддержанного в языках Lisp и его диалекте Clisp [16].

Преобразование программ. Часть сложностей обучения практическому параллельному программированию вытекает из достаточно очевидного сдвига понятий, вызванного тем, что реально не существует возможности использовать ЯВУ в чистом виде. Системы программирования подвергают программы разным, не вполне эквивалентным, оптимизирующим преобразованиям, увидеть и осознать эффект которых не всегда удаётся в учебной практике. Такие примеры для многопоточных программ могут быть связаны с распределением действий

по процессорам или с синхронизацией потоков действий, что технически выполнимо на базе Lisp, Clisp и других языков функционального программирования [16].

Метапрограммирование для преобразования программ. Достаточно простые преобразования сети потоков позволяют варьировать схемы выполнения вычислений и сводить многие конструкции языка программирования к взаимодействию простых потоков. Обратимость преобразований и чувствительность их результата к информационным связям между фрагментами программы требуют умения формализовать критерии применимости трансформаций и выбора подходящего варианта, включая перераспределение нагрузки как в mpC. Метапрограммирование при решении таких задач в основном может использовать символьную обработку, дополненную средствами проверки синтаксической эквивалентности и верификации программ [15, 16].

4. Специфика многопоточности

Переход к многопоточности резко расширяет пространство допустимых процессов выполнения программ и подразумевает разнообразие архитектур. Возникают вопросы взаимодействия локальной и общей памяти и фильтрации данных при обработке структур данных. Сложность вычислений может быть замаскирована просачиванием операций по структурам данных. Отладка взаимодействующих процессов происходит в условиях варьирования порядка асинхронных действий. Управление рассредоточенными действиями приводит к учёту событий и условий их готовности. Дополнительные сложности вытекают из задач синхронизации фрагментов программы. Всё это формирует навыки предвидеть проблемы, выходящие за пределы первичного представления программы.

Отделение порядка вычислений от порядка представления выражений. Для развития навыков сознательного выбора решений, влияющих на изменение данных в общей памяти, предложено в представлении структур данных учебного языка СИНХРОН разнести смысл скобок и разделителей. Скобки означают порядок доступа к элементам структур данных в памяти, а разделители — порядок вычисления элементов при исполнении программы. В результате структуры данных наполняются элементами, порядок вычисления которых может отличаться от порядка вхождения в программу. Многопоточная программа допускает асинхронное выполнение потоков и действий, что означает независимость порядка вычисления элементов выражений от порядка их вхождения в структуры данных, рассматриваемые как функции над этими выражениями, что выполнимо в рамках модели языков Lisp и Prolog.

Разнообразие архитектур. Проблемы параллельного программирования дополнительно осложнены дистанцией между уровнем абстрактных понятий, в которых описываются решения сложных задач, и уровнем конкретных аппаратных средств и архитектурных принципов управления параллельными вычислениями (потактовая синхронизация, совмещение действий во VLIW-архитектурах, сигналы, семафоры, буферы со временем ожидания сообщения, прерывания и т. п.). Проблемы подготовки параллельных программ для всех столь разных моделей обладают общностью, но есть и существенная специфика, требующая понимания разницы в критериях оценки программ и информационных систем для различных применений [1, 4, 17].

Взаимодействие локальной и общей памяти. Работа с данными при организации параллельных вычислений потребовала дополнительных решений. На уровне ядра языка необходим механизм взаимодействия локальной и общей памяти, отчасти смягчающий проблемы освобождения памяти для многопоточных программ [15, 18, 19]. Обычная система команд виртуальной машины пополнена средствами для решения проблем работы с общей памятью и внешними устройствами. Это команды пересылок данных и обмена данными в общей памяти многопроцессорного комплекса, нужные для профилактики возникновения временных интервалов между взаимосвязанными присваиваниями в общей памяти. Возможен побочный эффект присваивания, допускающий при необходимости восстановление прежних значений переменных — транзакционная память или персистентность, характерная для операционных систем и баз данных. Кроме того, предложена реализация императивной синхронизации взаимодействия локальной и общей памяти как пары запрос и его двойник, выполняемые неразрывно в стиле рандеву языка Ada.

Просачивание операций и функций. Основные методы лаконичного представления массовых вычислений связаны с использованием неявных циклов, позволяющих избежать выписывания однотипных схем над стандартными структурами данных типа многомерных векторов. Так, например, результат операции над скалярами в языках параллельного программирования обычно автоматически распространен на произвольные однородные структуры данных, что поддержано в языках APL, БАРС, Sisal. Этот механизм естественно распространяется на технику применения функций, что повышает лаконизм выражений [2, 11, 15].

Фильтрация данных. Из бинарных операций можно конструировать фильтры. Результат фильтрации исчезает из аргумента — он переносится в другую структуру данных, подобно обработке множеств в языке теоретико-множественного программирования SETL [10], что удобно при подготовке программ алгоритмов перебора. Структура из фильтров создает

структуру из результатов их применения к одному и тому же аргументу [15].

Взаимодействующие процессы. Популярная модель Т. Хоара чувствительна к обнаружению достаточно тонких ошибок при создании программ взаимодействия процессов, её нередко используют в системах верификации свойств программ [3]. Разработка таких программ отличается от подготовки обычных программ на весьма глубоком уровне, что можно показать на специальном диалекте языка СИНХРОН [15, 18, 19].

Варьирование контекста асинхронных действий. Текст программы на языке ПВ строится в определённой обстановке или контексте из действий – выражений или директив, использующих данные из контекста, выполняющего роль общей памяти. Выражения могут использовать размещённые в общей памяти данные, но не изменяют их. Директивы могут изменять хранимые в общей памяти данные, сохраняя доступ к устаревшим данным, как это поддержано механизмом транзакционной памяти в языке Clojure.

События и условия готовности. При выполнении многопоточной программы достижение некоторых позиций рассматривается как событие. Объявление события позволяет выполняться ждущим его действиям, обладающим готовностью. Событие не сбрасывается, пока все ждущие его потоки не будут готовы или их выполнение не будет оценено как невозможное. Независимые описания процессов можно связывать в терминах разметки барьерами — синхросети. Узлы с одинаковой разметкой срабатывают одновременно. Полное представление об асинхронных процессах, их эффективности и проблемах организации дают работы по сетям Петри в языке БАРС.

По мере выполнения действия формируются сообщения о готовности к выполнению, т. е. началу, о собственно выполнении и о завершении действия. Условное выполнение команды приводит к сбросу соответствующего сигнала. Действия, связанные с изменением состояния памяти, подчинены механизму транзакций, т. е. признание их безуспешными влечёт восстановление памяти в состояние, предшествующее этому действию.

Синхронизация слоёв программы. Многопоточная программа представляется как набор потоков и может быть размечена барьерами на слои. Каждый поток состоит из очереди вычисляемых, возможно, помеченных барьерами, действий. Барьеры выполняют роль точек синхронизации, разбивающих программу на слои. Каждый слой начинает выполняться одновременно и завершается до выполнения следующего слоя. Выражения одного слоя из разных, представленных независимо, потоков с одинаковой пометкой выполняются одновременно, как в синхросетях языка БАРС.

Навыки предвидеть проблемы. Часть проблем взаимодействия потоков алгоритмически не разрешима, поэтому в задачи ознакомления с параллелизмом входит научиться обнаруживать, предвидеть и отлаживать программы при обнаружении неудачно взаимодействующих потоков, что в своё время было проиллюстрировано в языке Робик. Наполнение многопоточной программы может развиваться независимо от схем управления вычислениями в отдельных потоках, а схемы можно реорганизовывать без дополнительной отладки наполнения в соответствии с принципом факторизации на автономно развиваемые модули. Схемы работают подобно макросам, но с контролем соответствия параметров объявленным видам фрагментов. Директива при благополучном исходе обработки памяти даёт результат подобно выражению.

Когнитивные ошибки. Одной из проблем при создании игр детьми является сложность представления и понимания сложных логических и вероятностных условий управления действиями персонажей. Эта проблема обусловлена противоречием между интуитивной и формальной оценкой истинности логических и вероятностных формул. Интуитивно истинность или вероятность связки «&&» оценивается выше, чем истинность составляющих, а реально она ниже. Это противоречие, возможно имеющее лингвистический характер, замечено в исследованиях нобелевского лауреата Д. Канемана, утверждающего, что математическое образование от таких ошибок спасает редко [20]. В языке SETL предлагалось решение этой проблему для ветвлений с помощью таблиц решений, применяемых в бухгалтерии.

Заключение

Проект разработки тренажёра для обучения программированию на базе учебного языка СИНХРОН представляет собой эксперимент по выбору базовых средств для достаточно полного решения проблем обучения эффективной реализации параллельных алгоритмов, вынужденно требующих использовать весьма широкий спектр сложно совместимых средств: от управляющих действий более низкого уровня, чем в привычных ЯВУ, до манипулирования пространствами решений по обработке данных в памяти, типичных для языков сверх высокого уровня [10]. Обзор исследований и решений в смежных областях представлен в [3, 4].

Особый круг учебных проблем связан с навыками учёта особенностей многоуровневой памяти в многопроцессорных системах. Обычное программирование такие проблемы может не замечать, полагаясь на решения компилятора, располагающего статической информацией о типах используемых данных и способного при необходимости выполнить оптимизирующие преобразования программы для конкретной архитектуры. Новые, учебные и долгоживущие языки

программирования, как правило, имеют мультипарадигмальный характер, что приводит к идее их расширения на многие модели параллельных вычислений. Нужна система обучения, поддерживающая осознание особенностей взаимодействия процессов, удобная для экспериментов по формированию навыков подготовки работоспособных программ и понимания новых возможностей аппаратуры.

К настоящему времени в рамках специализации студентов на базе ФИТ и ММФ НГУ выполнена реализация виртуальной машины на базе языка Clojure, достаточной для поддержки специфики многопоточных программ при ознакомлении с разными явлениями параллелизма, и визуальная оболочка на базе языка Kotlin, поддерживающая учебную разработку учебных игр3. Предстоит реализация диалектов для асинхронного выполнения независимых потоков программ и синхронизации взаимодействующих потоков при подготовке учебных многопоточных программ.

Литература[4]

1. Воеводин В. В., Воеводин Вл. В. Параллельные вычисления. — СПб.: БХВ-Петербург. — 2002. — 608 с.
2. Магариу Н. А. Язык программирования АПЛ. — М.: «Радио и связь», 1983. — 96 с.
3. Хоар Ч. Взаимодействующие последовательные процессы. — М.: Мир, 1989. — 264 с.
4. Марчук А.Г. , Городняя Л.В. Развитие моделей параллелизма в языках высокого уровня // Научный сервис в сети Интернет: все грани параллелизма: Труды Международной суперкомпьютерной конференции (23-28 сентября 2013 г., г. Новороссийск). — М.: Изд-во МГУ, 2013. — С. 342-346. — http://agora.guru.ru/abrau2013/pdf/342.pdf
5. Городняя Л.В. От трудно решаемых проблем к парадигмам программирования // XXVI Байкальская Всероссийская конференция с международным участием «Информационные и математические технологии в науке и управлении» (июль 2021 года, Иркутск). — https://cyberleninka.ru/article/n/ot-trudno-reshaemyh-problem-k-paradigmam-programmirovaniya
6. Городняя Л.В. О неявной мультипарадигмальности параллельного программирования // Научный сервис в сети Интернет: труды XXIII Всероссийской научной конференции (20-23 сентября 2021 г.). — М.:

[3] *Реализован уровень абстрактной машины языка СИНХРОН и оболочки студентами ФИТ и ММФ НГУ Д.В. Мажугой и М.Н. Дубковым.*

[4] Отсутствуют ссылки на описания языков программирования, их можно найти через поисковики или Википедию.

ИПМ им. М.В.Келдыша, 2021. — С. 104-116. — https://doi.org/10.20948/abrau-2021-6 https://keldysh.ru/abrau/2021/theses/6.pdf
7. Городняя Л.В. О функциональном программировании // Журнал «Компьютерные инструменты в образовании» 2021, выпуск 3. — С. 57-75. — http://ipo.spb.ru/journal/index.php?article/2288/ . — https://cyberleninka.ru/article/n/o-funktsionalnom-programmirovanii
8. Городняя Л.В. Место функционального программирования в организации параллельных вычислений // Информационные и математические технологии в науке и управлении. — Выпуск №1(25) / 2022. — С. 102-119. (796.6 KB). — https://www.imt-journal.ru/archive/public/article?id=230 — https://cyberleninka.ru/article/n/mesto-funktsionalnogo-programmirovaniya-v-organizatsii-parallelnyh-vychisleniy DOI: 10.38028/Б81.2022.25.1.009
9. Городняя Л. В. Перспективы функционального программирования параллельных вычислений // «Электронные библиотеки» *24*(6). — С. 1090-1116. — https://doi.org/10.26907/1562-5419-2021-24-6-1090-1116 https://rdl-journal.ru/article/view/713
10. Jacob T. Schwartz. Abstract algorithms and a set theoretic language for their expression. Computer Science Department, Courant Institute of Mathematical Sciences, New York University. 1971. — https://www.softwarepreservation.org/projects/SETL/setl/doc/Schwartz-Abstract_Algorithms-1971.pdf
11. Cann D. C. SISAL 1.2: A Brief Introduction and tutorial. Preprint UCRL-MA-110620. Lawrence Livermore National Lab., Livermore, California, May, 1992. — 128 p.
12. Сайт проекта Норма (Андрианов А.Н.) — https://keldysh.ru/pages/norma/
13. Ластовецкий А.Л. Программирование параллельных вычислений на неоднородных сетях компьютеров на языке mpC (Интерактивный учебный курс) — https://parallel.ru/tech/mpc/mpC-rus.html
14. Малышкин В.Э. Технология фрагментированного программирования — http://omega.sp.susu.ru/books/conference/PaVT2012/short/212.pdf
15. Городняя Л.В. Работа с данными в учебном языке программирования СИНХРО // Суперкомпьютерные дни в России. Труды международной конференции. 26-27 сентября 2022 г., Москва / Под. ред. Вл. В. Воеводина. — М.: МАКС Пресс, 2022. — С. 87-97 — https://doi.org/10.29003/m3109.RussianSCDays2022 — https://doi.org/10.25205/1818-7900-2021-19-4-16-35
16. Адамович И.А., Климов Ю.А. Специализация интерпретаторов на объектно-ориентированных языках может быть эффективной // Научный сервис в сети Интернет: труды XXIV Всероссийской научной

конференции (19-22 сентября 2022 г., онлайн). — М.: ИПМ им. М.В.Келдыша, 2022. — С. 3-24. — https://doi.org/10.20948/abrau-2022-18, https://keldysh.ru/abrau/2022/theses/18.pd
17. Андреева Т.А., Городняя Л.В. Можно ли измерять вклад программистских решений в производительность программ? // Научный сервис в сети Интернет: труды XXV Всероссийской научной конференции (18-21 сентября 2023 г., онлайн). — М.: ИПМ им. М.В.Келдыша, 2023. — С. 12-24. — https://doi.org/10.20948/abrau-2023-2 https://keldysh.ru/abrau/2023/theses/2.pdf
18. Городняя Л.В. Модели работы с памятью в учебном языке программирования СИНХРО //Научный сервис в сети Интернет: труды XXIV Всероссийской научной конференции (19-22 сентября 2022 г., онлайн). — М.: ИПМ им. М.В.Келдыша, 2022. — С. 137-154. — https://doi.org/10.20948/abrau-2022-1 https://keldysh.ru/abrau/2022/theses/1.pdf
19. Городняя Л.В. Абстрактная машина языка программирования учебного назначения СИНХРО // Вестник НГУ. Серия: Информационные технологии. — 2021, Т.19, №4. — С. 16-35. — DOI: 10.25205/1818-7900-2021-19-4-16-35
20. Канеман Д. Думай медленно ... решай быстро. (Thinking, Fast and Slow) — М.: АСТ. — 2013. — 625 с.

References

1. Voyevodin V. V., Voyevodin Vl. V. Parallel'nyye vychisleniya. — SPb.: BKHV-Peterburg, 2002. — 608 s.
2. Magariu N. A. YAzyk programmirovaniya APL. — M.: «Radio i svyaz'», 1983. — 96 s.
3. Hoare C.A.R. Vzaimodeystvuyushchiye posledovatel'nyye protsessy. — M.: Mir, 1989. — 264 s.
4. Marchuk A.G., Gorodnyaya L.V. Razvitiye modeley parallelizma v yazykakh vysokogo urovnya // Nauchnyy servis v seti Internet: vse grani parallelizma: Trudy Mezhdunarodnoy superkomp'yuternoy konferentsii (23-28 sentyabrya 2013 g., g. Novorossiysk). — M.: Izd-vo MGU, 2013. —S. 342-346. — http://agora.guru.ru/abrau2013/pdf/342.pdf
5. Gorodnyaya L.V. Ot trudno reshayemykh problem k paradigmam programmirovaniya // XXVI Baykal'skaya Vserossiyskaya konferentsiya s mezhdunarodnym uchastiyem «Informatsionnyye i matematicheskiye tekhnologii v nauke i upravlenii» (iyul' 2021 goda, Irkutsk) — https://cyberleninka.ru/article/n/ot-trudno-reshaemyh-problem-k-paradigmam-programmirovaniya
6. Gorodnyaya L.V. O neyavnoy mul'tiparadigmal'nosti parallel'nogo programmirovaniya // Nauchnyy servis v seti Internet: trudy XXIII Vserossiyskoy nauchnoy konferentsii (20-23 sentyabrya 2021 g.). — M.:

IPM im. M.V.Keldysha, 2021. — S. 104-116. — https://doi.org/10.20948/abrau-2021-6 — https://keldysh.ru/abrau/2021/theses/6.pdf
7. Gorodnyaya L.V. O funktsional'nom programmirovanii // Zhurnal «KIO» 2021, vypusk 3. — S. 57-75. — http://ipo.spb.ru/journal/index.php?article/2288/ — https://cyberleninka.ru/article/n/o-funktsionalnom-programmirovanii
8. Gorodnyaya L.V. Mesto funktsional'nogo programmirovaniya v organizatsii parallel'nykh vychisleniy // IMT v nauke i upravlenii Vypusk №1(25) / 2022, — S. 102-119. — https://www.imt-journal.ru/archive/public/article — https://cyberleninka.ru/article/n/mesto-funktsionalnogo-programmirovaniya-v-organizatsii-parallelnyh-vychisleniy DOI: 10.38028/B81.2022.25.1.009
9. Gorodnyaya L. V. Perspektivy funktsional'nogo programmirovaniya parallel'nykh vychisleniy. // «Elektronnyye biblioteki» 24(6). — S. 1090-1116. — https://doi.org/10.26907/1562-5419-2021-24-6-1090-1116 — https://rdl-journal.ru/article/view/713
10. Schwartz, Jacob T., "Set Theory as a Language for Program Specification and Programming". — Courant Institute of Mathematical Sciences, New York University, 1970. SETL (1972 J.T.Schwartz)
11. Cann D. C. SISAL 1.2: A Brief Introduction and tutorial. Preprint UCRL-MA-110620. Lawrence Livermore National Lab., Livermore — California, May, 1992. — 128 p.
12. Sayt proyekta Norma (Andrianov A. N.) — https://keldysh.ru/pages/norma/
13. Lastovetskiy A.L. Programmirovaniye parallel'nykh vychisleniy na neodnorodnykh setyakh komp'yuterov na yazyke mpC (Interaktivnyy uchebnyy kurs) — https://parallel.ru/tech/mpc/mpC-rus.html
14. Malyshkin V.E. Tekhnologiya fragmentirovannogo programmirovaniya — http://omega.sp.susu.ru/books/conference/PaVT2012/short/212.pdf
15. Gorodnyaya L.V. Rabota s dannymi v uchebnom yazyke programmirovaniya SINKHRO // Superkomp'yuternyye dni v Rossii. — S. 87–97 // Trudy mezhdunarodnoy konferentsii. 26-27 sentyabrya 2022 g., Moskva / Pod. red. Vl. V. Voyevodina. — M.: MAKS Press, 2022. ISBN 978-5-317-06875-2 ye-ISBN 978-5-317-06876-9 — https://doi.org/10.29003/m3109.RussianSCDays2022. S. 87–97. — https://doi.org/10.25205/1818-7900-2021-19-4-16-35
16. Adamovich I.A., Klimov Yu.A. Spetsializatsiya interpretatorov na ob"yektno-oriyentirovannykh yazykakh mozhet byt' effektivnoy // Nauchnyy servis v seti Internet: trudy XXIV Vserossiyskoy nauchnoy konferentsii (19-22 sentyabrya 2022 g., onlayn). — M.: IPM im. M.V.Keldysha, 2022. — S. 3-24. — https://doi.org/10.20948/abrau-2022-18, https://keldysh.ru/abrau/2022/theses/18.pdf

17. Andreyeva T.A., Gorodnyaya L.V. Mozhno li izmeryat' vklad programmistskikh resheniy v proizvoditel'nost' programm? // Nauchnyy servis v seti Internet: trudy XXV Vserossiyskoy nauchnoy konferentsii (18-21 sentyabrya 2023 g., onlayn). — M.: IPM im. M.V.Keldysha, 2023. — S. 12-24. — https://doi.org/10.20948/abrau-2023-2 — https://keldysh.ru/abrau/2023/theses/2.pdf
18. Gorodnyaya L.V. Modeli raboty s pamyat'yu v uchebnom yazyke programmirovaniya SINKHRO // Nauchnyy servis v seti Internet: 2022. — S. 137-154.
19. Gorodnyaya L.V. Abstraktnaya mashina yazyka programmirovaniya uchebnogo naznacheniya SINKHRO // Vestnik NGU. Seriya: Informatsionnyye tekhnologii. 2021 T.19, №4. — S. 16-35. DOI: 10.25205/1818-7900-2021-19-4-16-35
20. Kaneman D. Dumay medlenno ... reshay bystro. (Thinking, Fast and Slow) — M.: AST, 2013. — 625 s.