Московский Авиационный Институт

(Национальный исследовательский университет)

Реферат

По курсу:

«Логическое программирование»

На тему:

«Современные языки и системы логического программирования»

Студент: Забарин Н.И.

Группа: 8О – 408Б

№ по списку: 21

Москва, 2016

Истоки логического программирования

PLANNER

Логическое программирование пошло от функционального, первый язык логического программирования был создан как расширение LISP. Это был PLANNER. Своеобразный диалект LISP’а расширяющий его возможности, к примеру, функции ELEM и REST обобщают функции CAR и CDR, позволяя выделять и отбрасывать из списка произвольные элементы. Одним из главных нововведений стало “моделирование” искусственного интелекта, с помощью описания среды в которой решается задача.

Описанием среды является база истинных утверждений о среде, а так же описание логических отношений между используемыми понятиями и действий которые можно осуществлять. Такие описания называют теоремами, для использования такой теоремы необходимо что бы были выполнены условия ее осуществения. Результатом выполнения теоремы являются изменения в базе описания среды, удаление записей ставших ложными, а так же добавление “истиных” записей.

Но самым важным нововведением является режим возвратов. Он позволяет во время работы программы откатывать часть примененных изменений, если оказывается что они отдаляют нас от цели. Этот механизм является встроенным в язык механизмом перебора, который характерен для искусственного интелекта.

Так же одним из прижившихся новшеств является поиск данных по образцу.

Prolog

Prolog появился как упрощение PLANNER’a, в котором не требовалось плана перебора вариантов. Язык построен вокруг небольшого набора основных механизмов, таких как сопоставление с образцом, древовидное представление структур данных и автоматический перебор с возвратами. Для решения задачи необходимо описать объекты и отношения между ними. Благодаря своим особенностям Пролог нашел широкое применение в задачах искусственного интелекта, компьютерной лингвистики и нечислового программирования. Для решения этих задач на стандартных языках необходимо создавать довольно объемные программы, когда на прологе такая программа будет занимать гораздо меньше места.

Что бы реализовать на прологе сложную и практически полезную программуприходится прибегать к процедурным приемам что приводит к возрастанию сложности создания и отладки приграммы. Так же одним из минусов языка является сложность распараллеливания.

Современные языки логического программирования

Не смотря на то что Пролог до сих пор активно используется, он имеет ряд важных проблем:

1. Производительность. Современные языки логического программирования заметно уступают в производительности императивным языкам.
2. Отсутствие отладочной системы. Пролог не имеет строгой типизации и осуществляет очень мало проверок на этапе компиляции, что вынуждает разработчика искать ошибки в ручную, почти без помощи от отладчика

Для решения этих двух проблем был создан Mercury.

Mercury

Язык был разработан весной 1995 года, и получил свое название от бога скорости Меркурия. Он частично унаследовал синтаксис от Пролога, а систему типов от Haskell. Из языка полностью убрали все императивные возможности, что позволило улучшить встроенные в компилятор возможности оптимизации. Имеется поддержка трансляции языка на многие популярные, такие как C и Java.

В настоящее время язык активно дорабатывается, новые версии выходят каждые пол года.

Erlang

Этот язык был предназначен для создания отказоустойчивых, распределенных систем, синтаксис и некоторые концепции он унаследовал от Пролога, хотя и является функциональным языком. В Erlang встроенны механизмы создания параллельных легковесных процессов, взаимодействия между ними с помощью асинхронных сообщений. Одной из интересных особенностей языка является замена кода без остановки программы.

В настоящее время Erlang используется для создания “облачных” сервисов, в телекоммуникационных и Интернет приложениях, к примеру бэкенд WhatsApp. А так же он очень удобен при решении задач искусственного интелекта, основанных на нейронных сетях, благодаря тому, что он обладает пятью свойствами “языка программирования нейронных сетей”: изолированные процессы-нейроны, параллелизм, механизм обнаружения сбоев, независимость от местоположения и горячая замена кода.

Oz

В отличии от Mercury Oz включается в себя большинство популярных парадигм программирования, среди них логического, функционального, императивного, объектно-ориентированного, программирования с ограничениями, распределённого и параллельного программирования.

Язык отличается хорошими возможностями распределения вычислений, а так же поддерживает программирование в ограничениях.

Visual Prolog

Является объектно-ориентированным расширением языка Пролог, с этим диалектом связанна система визуального программирования.

Visual Prolog автоматизирует построение сложных проуедур и освобождает программиста от выполнения тривиальных операций. С помощью Visual Prolog проектирование пользовательского интерфейса и связанных с ним окон, диалогов, меню, строки уведомлений о состояниях и т. д. производится в графической среде. Мощность языка Prolog в сочетании с системой пользовательских интерфейсов упрощает разработку систем, основанных на знаниях, систем поддержки принятия решений, планирующих программ, развитых систем управления базами данных и т. д.

Язык программирования, реализованный в Visual Prolog`e отличается от классического пролога тем, что он основан на строгой статической типизации. В него также добавлены средства объектно-ориентированного программирования, анонимные предикаты, факты-переменные и разрушающее присваивание для них, аргументы-домены и параметрический полиморфизм, мониторы, императивные конструкции, коллекторы списков и пр.

Строгая типизация позволяет программам быть скомпилированными непосредственнно в машинные коды, при это, скорость выполнения сравнима и даже превышает скорости аналогичных программ на императивных языках.

Среда разработки приложений системы Visual Prolog включает текстовый редактор, различные редакторы ресурсов, средства разработки справочных систем в гипертекстовом представлении, систему отслеживания изменений, которая обеспечивает перекомпиляцию и перегенерацию только измененных ресурсов и модулей, ряд экспертов Кода, оптимизирующий компилятор, набор средств просмотра различных типов информации о проекте и отладчик. Полная интеграция всех средств обеспечивает повышение скорости разработки приложений. Полученные приложения являются исполняемыми .EXE программами. В коммерческой версии Visual Prolog 7.x возможно создание .DLL-файлов, в персональной версии такая возможность существовала вплоть до версии 5.x.