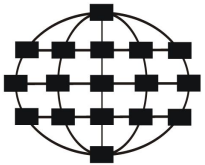
Всероссийская летняя XLI молодежная Школа-конференция по параллельному программированию с международным участием



Об автоматическом синтезе параллельных программ на основе баз активных знаний

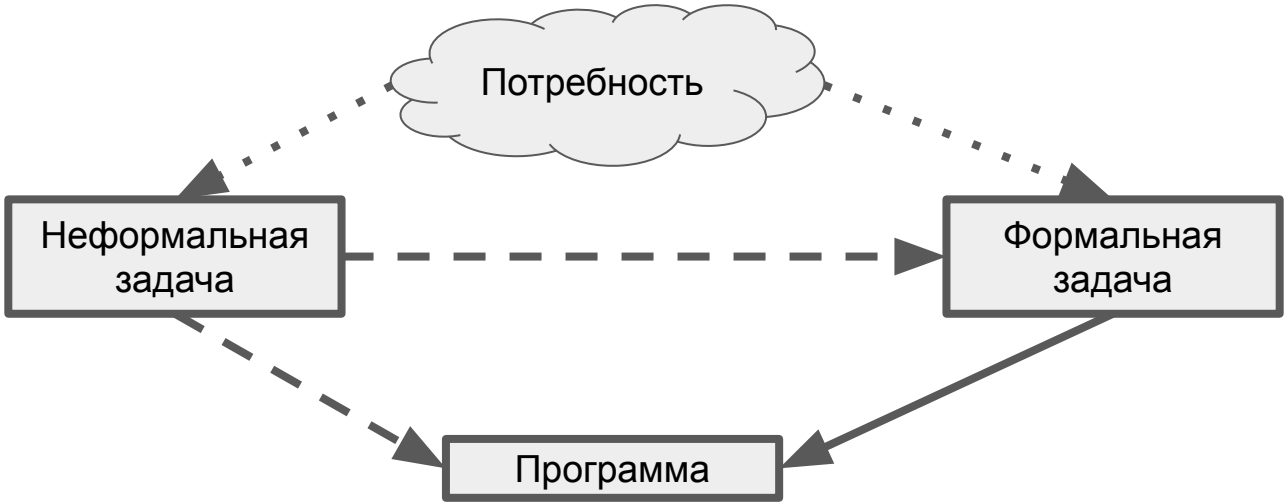
В.А. Перепёлкин

Новосибирск, 2023 г.

Компьютеры потенциально очень полезны

Но как заставить их делать что нам надо?

В чём проблема: компьютер не может просто взять и сделать, что человек от него хочет



Нельзя ли сделать так, чтобы компьютер *сам* создавал программу или хотя бы помогал в этом человеку?

В чём проблема: компьютер не может просто взять и сделать что человек от него хочет

- Человек что-то хочет
- Человек неформально говорит, что ему нужно
- Специалист формирует формальный запрос (не обязательно)
- Программист пишет программу, по которой компьютер может выполнить требуемое
  - Творческие подзадачи
  - Рутинные подзадачи
- Компьютер выполняет программу

Если компьютер мог бы делать программу сам, то какой нужен интерфейс?

Нужен язык для точной формулировки задачи: математика

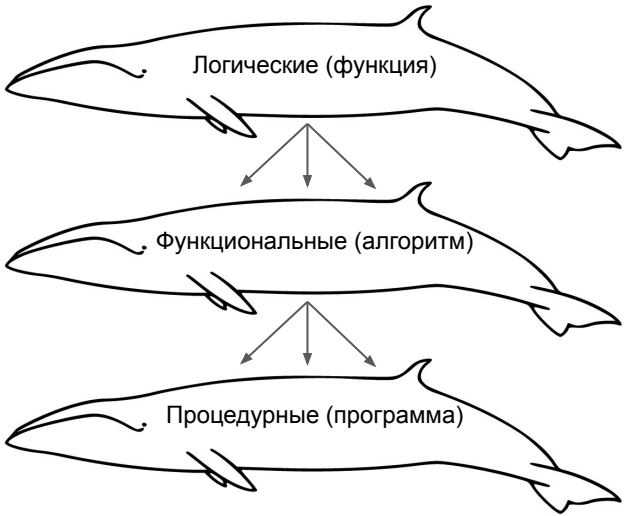
Задача создания программы — алгоритмически труднорешаемая (класс NP)

Идеально:

Универсальных решений нет, каждая* предметная область уникальна.

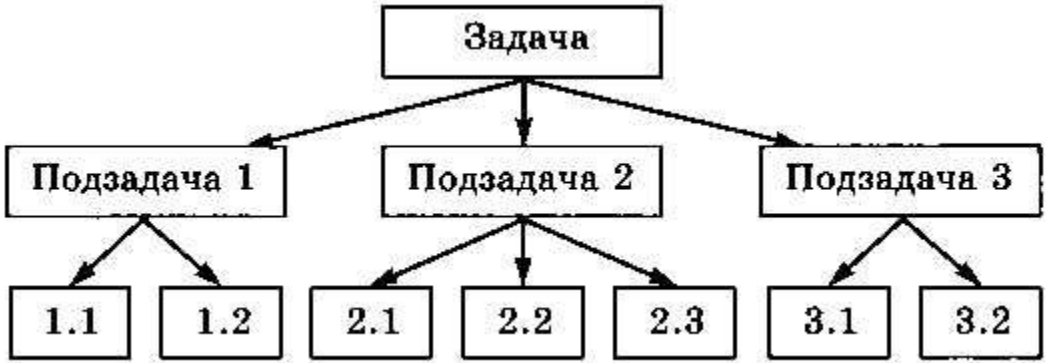
Следствие: придётся работать с частными решениями

«Все счастливые семьи — похожи друг на друга, каждая несчастливая семья — несчастна по-своему» (Л. Н. Толстой, Анна Каренина)



*Кроме одинаковых

Задача и подзадачи (рутинные и творческие)



Максимум автоматизации рутины: декомпозиция и насыщенность

Со временем «творческие» предметные области могут «обрутиниваться» и «закрываться»

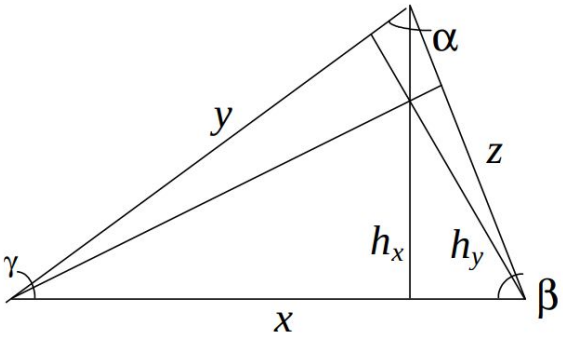
Промежуточный итог:

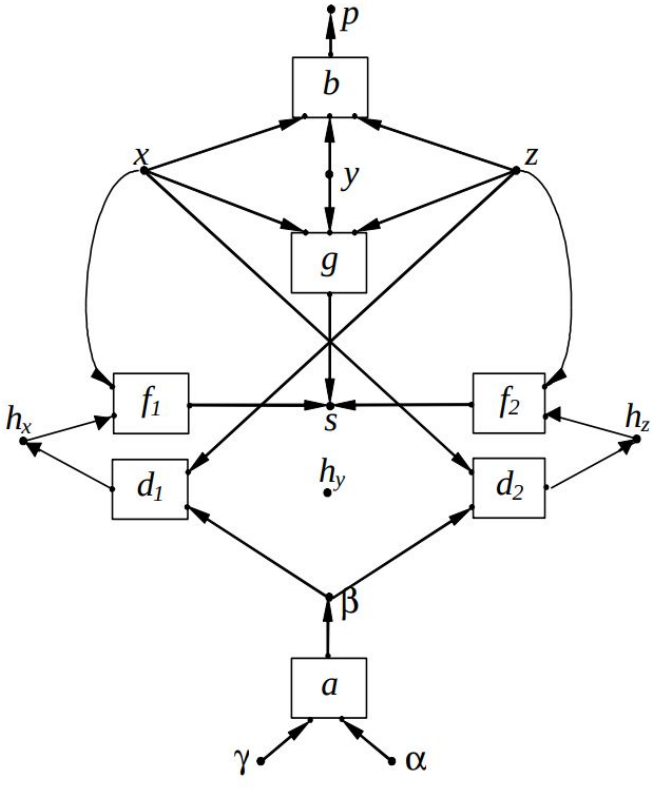
- Создание программы — ключевой элемент
- Компьютер может помочь в создании программы лишь отчасти

Ключ: автоматизация применения накопленных решений рутинных подзадач

Вычислительные модели

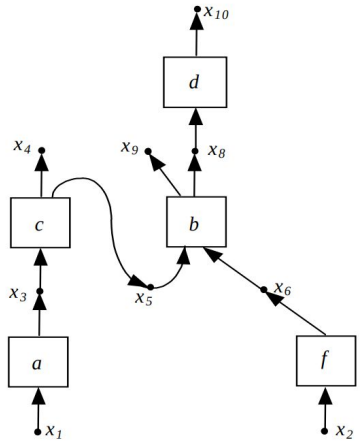
Функциональная постановка задачи: как задавать?



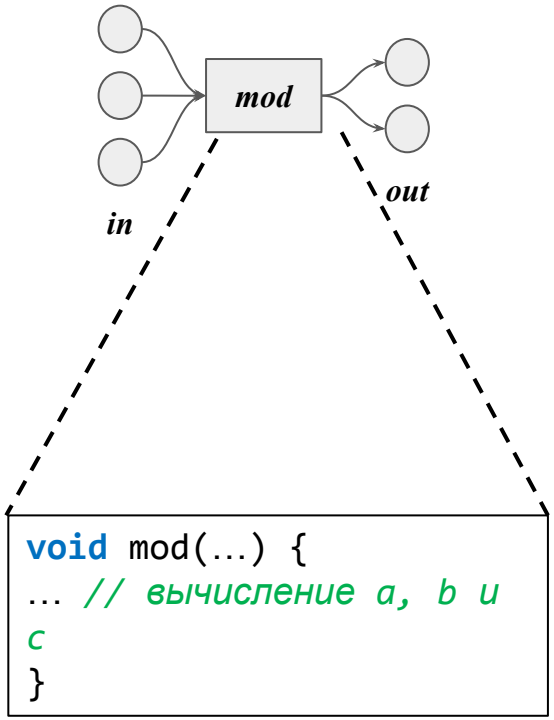


Синтез программ на вычислительных моделях

- Описать вычислительную модель
- Определить множества входных и выходных переменных (задачу)
- Вывести (оптимальный) план решения задачи (алгоритм)
- Сгенерировать (эффективную) программу, реализующую алгоритм
- Задать значения входных переменных
- Запустить программу

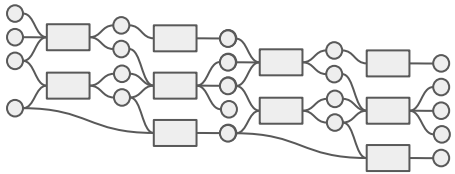


Конструирование параллельной программы по описанию алгоритма (фрагментированное программирование)



Алгоритм представляется как множество триплетов вида ‹in, mod, out›, где in и out – иммутабельные переменные, а mod – процедура без побочных эффектов

Выгода: возможность автоматически конструировать параллельную программу с различными нефункциональными свойствами



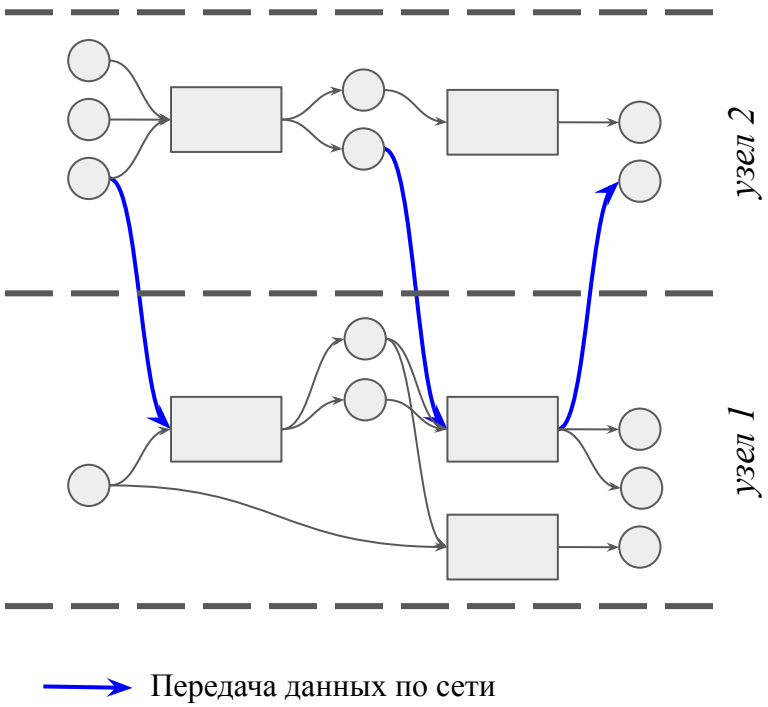
Распределённое исполнение такой «фрагментированной» программы и автоматическое обеспечение требуемых свойств

Система автоматически распределяет триплеты по узлам мультикомпьютера и исполняет их по готовности входных аргументов.

Планирование вычислений, передача данных по сети и сборка мусора выполняются автоматически.

Динамическая балансировка нагрузки достигается перераспределением триплетов по узлам мультикомпьютера.

Сохранение контрольных точек и обеспечение отказоустойчивости возможно путём сохранения промежуточных значений.



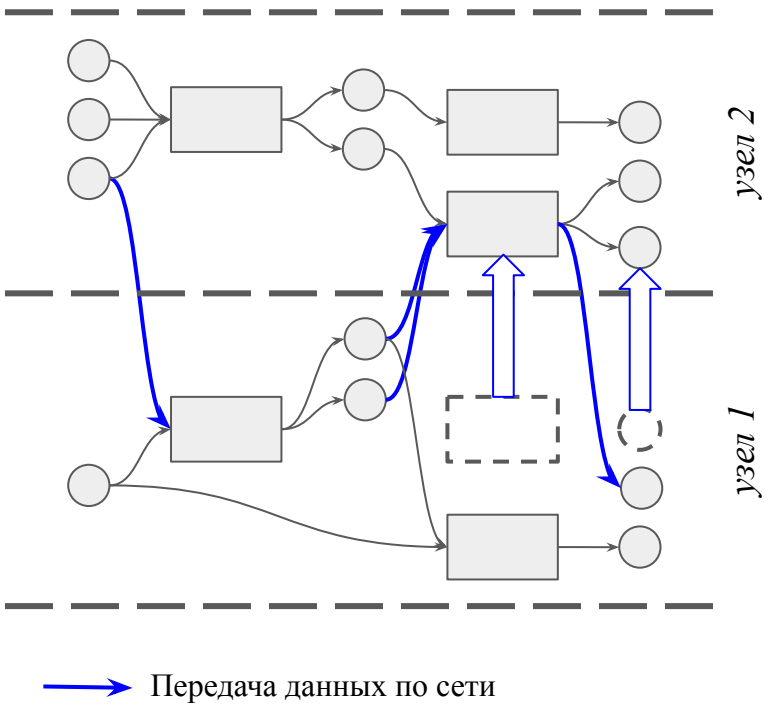
Распределённое исполнение такой «фрагментированной» программы и автоматическое обеспечение требуемых свойств

Система автоматически распределяет триплеты по узлам мультикомпьютера и исполняет их по готовности входных аргументов.

Планирование вычислений, передача данных по сети и сборка мусора выполняются автоматически.

Динамическая балансировка нагрузки достигается перераспределением триплетов по узлам мультикомпьютера.

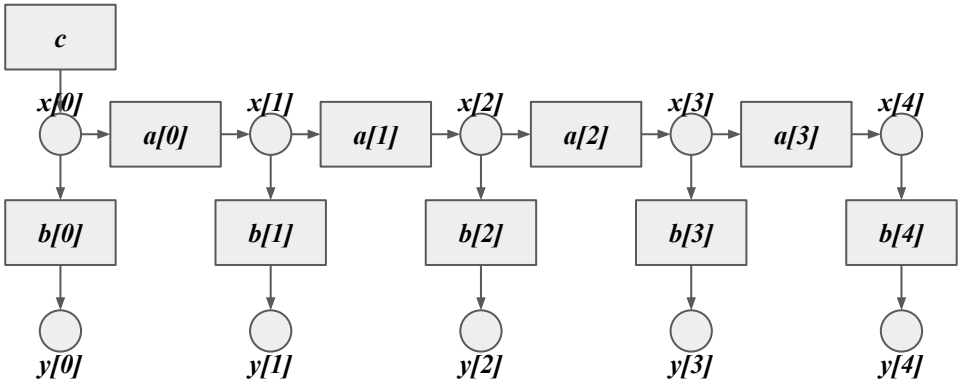
Сохранение контрольных точек и обеспечение отказоустойчивости возможно путём сохранения промежуточных значений.



Пример LuNA-программы

```
import init(name);
import calc(value, name);
import conv(int, name);

sub main() {
  df x, y, N;
  while x[i]>0, i=0..out N
    cf a[i]: calc(x[i], x[i+1]);
  for i=0..N {
    cf b[i]: conv(x[i], y[i]);
  }
  cf c: init(x[0]);
}
```



Язык LuNA — язык описания вычислительных моделей (почти)

Архитектура системы активных знаний

БП — библиотека подпрограмм

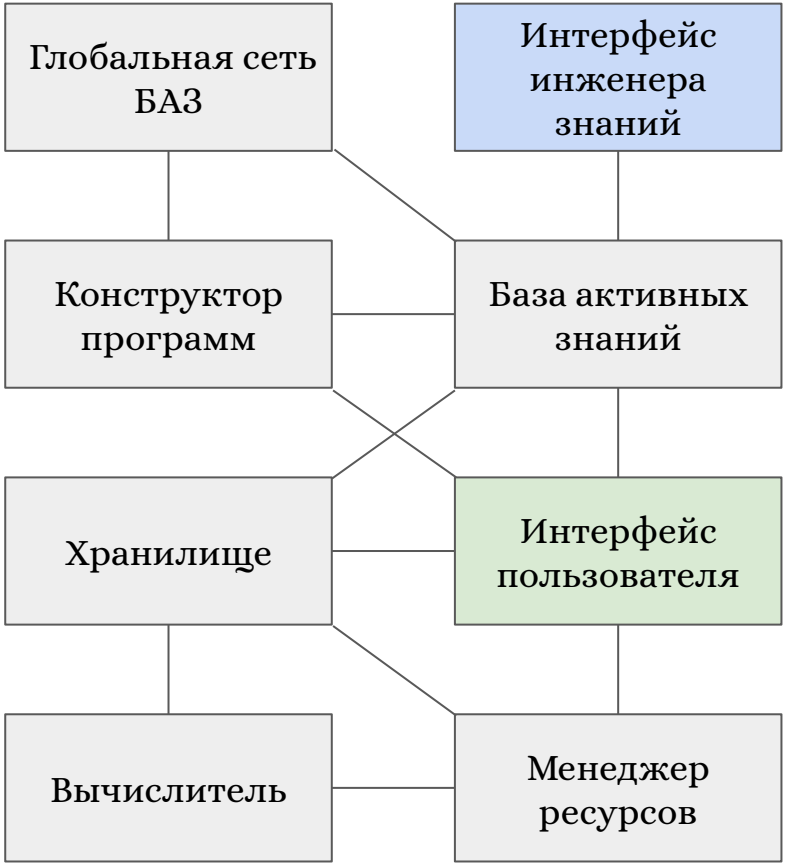
ВМ — вычислительная модель

БАЗ — база активных знаний

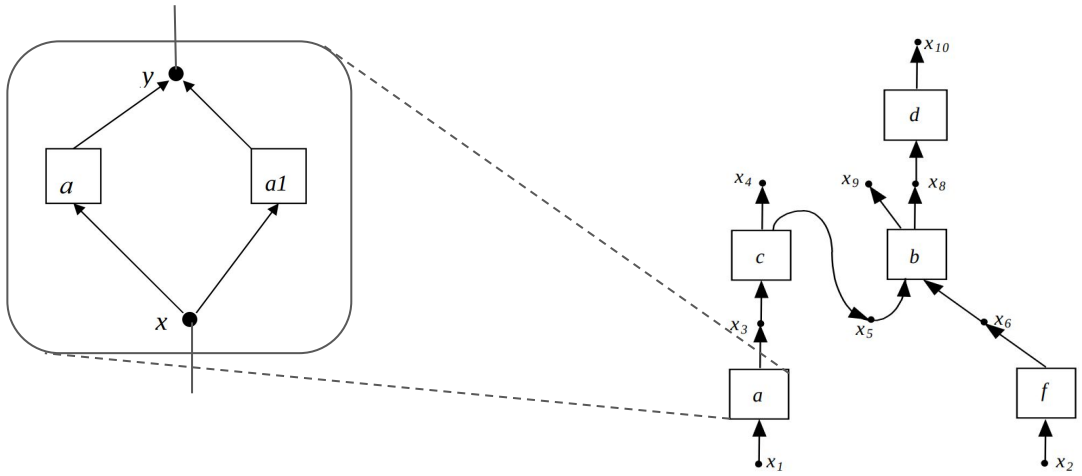
БП + ВМ = БАЗ

Ещё нужны:

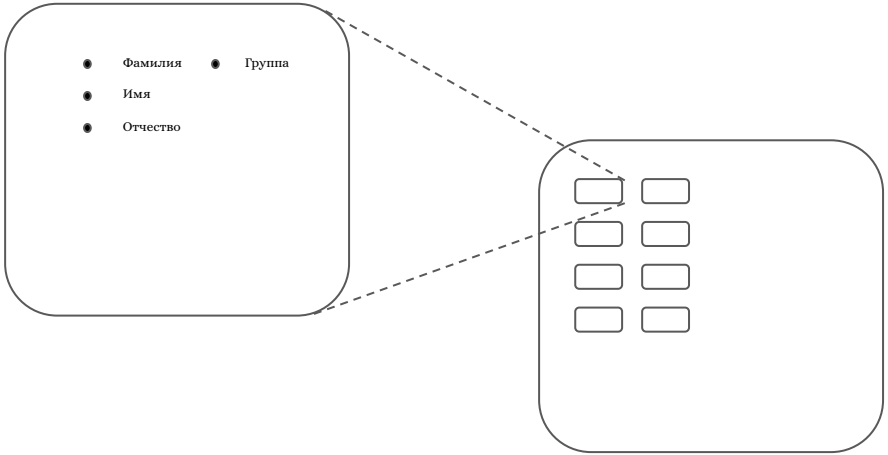
- специфичные алгоритмы
- нефункциональный слой
- система активных знаний (СУ БАЗ)



Структурированные операции и вложенные вычислительные модели



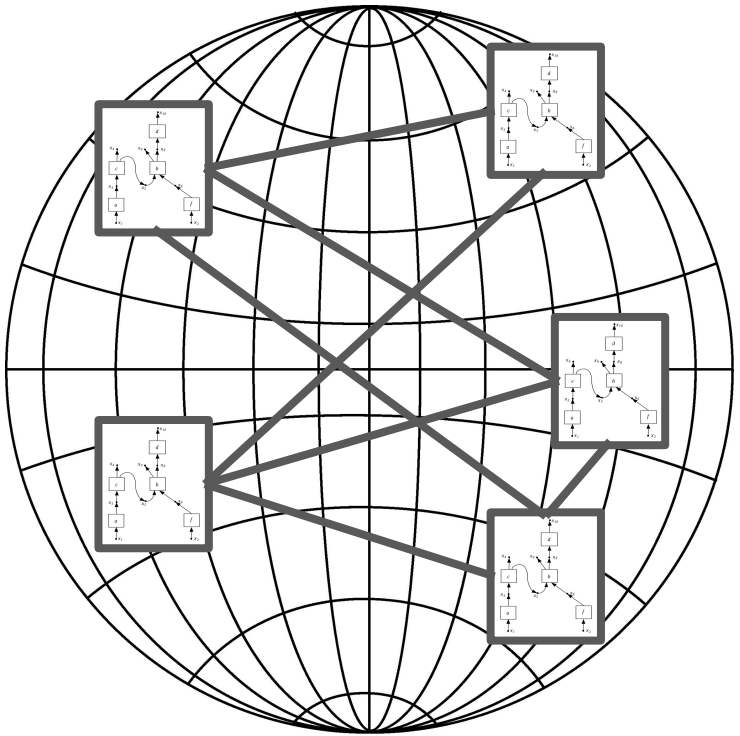
Вычислительные модели без операций?



Даже без операций вычислительные модели определяют функциональные отношения

Глобальная сеть активных знаний

- Базы активных знаний естественным образом объединяются в сеть
- Эта сеть по своей природе — глобальная
- Создание такой сети неизбежно
- Требуется решить ряда проблем...



Связанные вопросы

- Частный пользовательский интерфейс: умолчания
- Системная база активных знаний
- Документирование программ
- Динамическое конструирование программ и исполнительных систем: под задачу, вычислитель, данные, ход вычислений
- Глобальная сеть баз активных знаний
  - Ответственные организации
- Универсальный интерфейс на вычислительных моделях
- Цифровые двойники на основе вычислительных моделей

Заключение

Базы активных знаний позволяют накапливать частные решения подзадач в различных предметных областях и переиспользовать эти решения для создания новых программ