Институт вычислительной математики и математической геофизики СО РАН
Лаборатория синтеза параллельных программ

Концепция активных знаний – методология автоматического конструирования программ в предметных областях

Перепёлкин Владислав Александрович
к.т.н., с.н.с. ИВМиМГ СО РАН

Летняя XLVI школа-конференция по параллельному программированию
29 июня – 10 июля 2026, Новосибирск

Задача автоматического конструирования программ

как заставить компьютеры делать что нам надо?

Программа — это:

- машинный код?
- код на ЯВУ?
- что-то другое?

Генератор, интерпретатор, исполнительная система

Системы параллельного программирования

- Система НОРМА (И.Б. Задыхайло, А.Н. Андрианов)
- DVM-система (В.А. Крюков)
- OpenTS (С.М. Абрамов)
- Charm++ (L. Kale)
- PaRSEC / DPLASMA (J. Dongarra)
- Legion / Regent (A. Aiken)

Конструирование программ на основе ИИ

- Большие языковые модели общего назначения
- Специализированные ИИ-ассистенты
- Агентный ИИ

Пакеты моделирования

- Fluent
- NAMD

Низкоуровневые инструменты

- Message Passing Interface
- OpenMP
- CUDA

Системы-конструкторы программ

- Galaxy
- MindSpore

Библиотеки подпрограмм

- NumPy
- MatLab

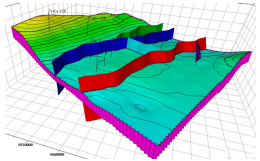
Много наработок, но проблема до сих пор открыта

Почему так сложно создать программу?

Разработка **эффективных численных** программ для **суперкомпьюеров** сложно, трудоёмко и требует **специфичных** знаний и умений.

Проблемы параллельного программирования:

- Декомпозиция данных и вычислений
- Распределение и динамическое перераспределение данных и вычислений по узлам мультикомпьютера
- Управление параллельной обработкой распределённых данных
- Обеспечение высокой эффективности



Осложняющие факторы:

- Различная конфигурация вычислительных узлов
- Использование GPU и ПЛИС
- Большой размер суперкомпьютера

Для достижения высокой эффективности приходится учитывать особенности:

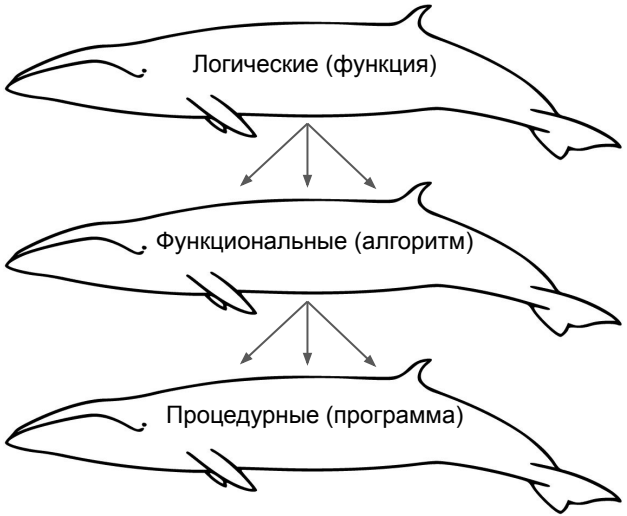
- прикладного алгоритма
- суперкомпьютера
- входных данных

Задача создания программы — алгоритмически труднорешаемая (класс NP)

Идеально:

Универсальных решений нет, каждая предметная область уникальна.

Следствие: придётся работать с частными решениями

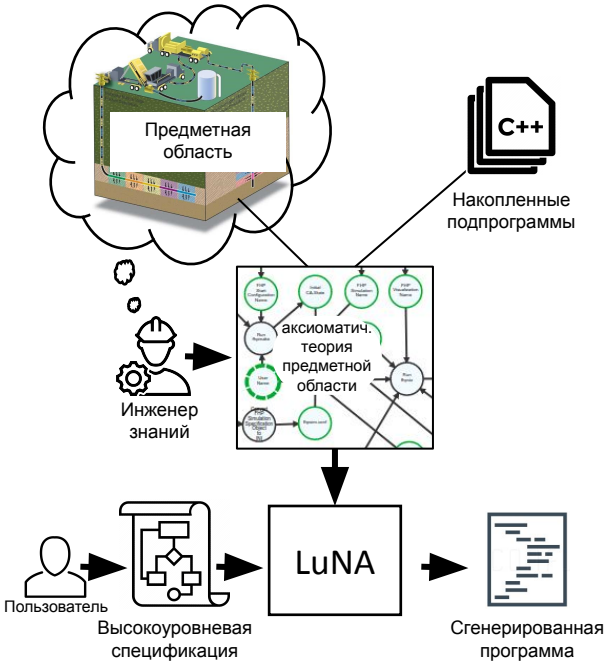


Даже LLM не станут универсальным решением

Концепция активных знаний: опора на формальное описание предметной области

Направляющие идеи:

- Отказ от поиска универсальных решений
- Использование существующих наработок в предметных областях
- Опора на специальную **аксиоматическую теорию предметной области**
- Идея активных знаний, «положил и забыл»



Особенности подхода:

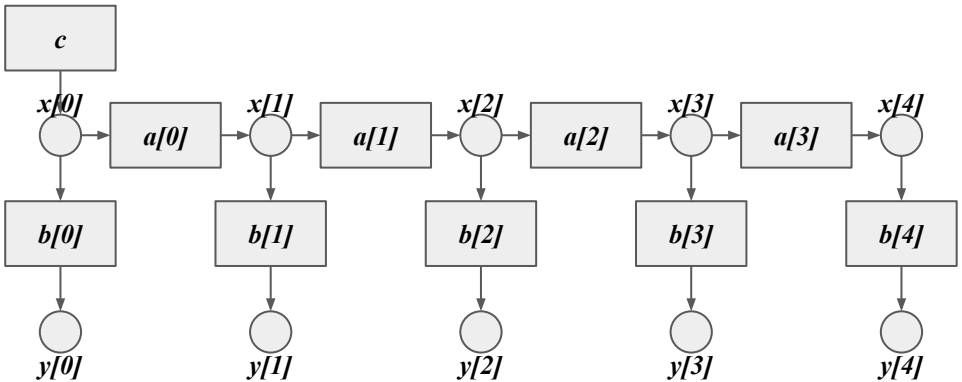
- Строится база активных знаний (**БАЗ**) для предметной области (в основе — аксиоматическая теория)
- Алгоритм и программа решения задачи строятся автоматически из готовых модулей
- Разработка БАЗ предметной области — сложная междисциплинарная задача
- БАЗ содержит различные решения множества задач
- Возможность решать практические задачи
- Компетентное применение модулей
- Параллельное программирование исключается из процесса разработки параллельной программы

БАЗ снижает сложность задачи до приемлемой (гид, словарь)

Почему двое из ларца делали всё неправильно?

Как по схеме алгоритма строить программу?

```
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]);
}
```



```
import inc(in int num, out int);
import "sum" (in int, in int, out int) as sum;

sub sum_inc(a,b,c) {
	name a, b, c;
	sum(a, b, c);
	inc(c, res);
}

interface "clif" main (
	"string" => int a,
	"string" => int b,
	"string" <= int res
) on sum_inc;

```

```
void inc(int num, int *res) {
	*res = num+1;
}

```

```
void add(int num, int what, int *res) {
	*res = num+what;
}

```

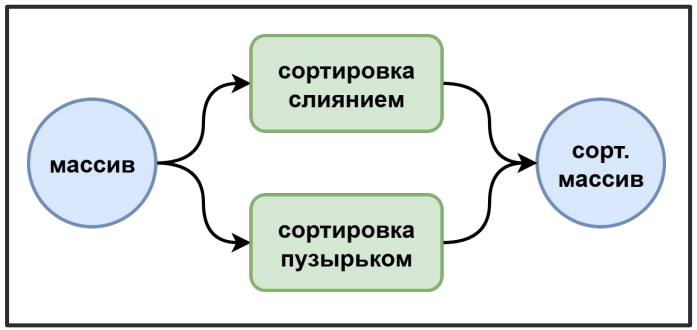
```
import inc(in int num, out int);
import "sum" (in int, in int, out int) as sum;

sub sum_inc(a,b,c) {
	name a, b, c;
	sum(a, b, c);
	inc(c, res);
}

interface "clif" main (
	"string" => int a,
	"string" => int b,
	"string" <= int res
) on sum_inc;

```

```
void inc(int num, int *res) {
	*res = num+1;
}

```

```
void add(int num, int what, int *res) {
	*res = num+what;
}

```

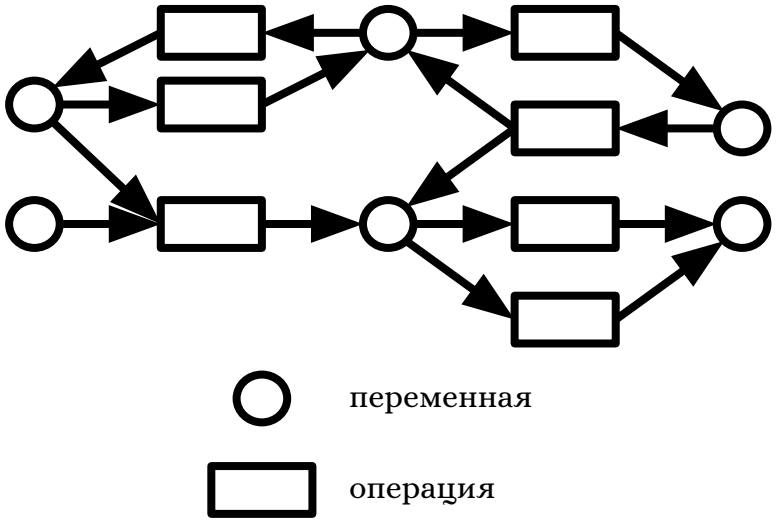
```
// dependencies
void add(int, int, int *);
void inc(int, int *);

// wrappers


void _cpf_auto_0(int _v0, int _v1, int *_v2) {
	// local vars
	int _v3;

	// invokes
	add(_v0, _v1, &_v3);
	inc(_v3, _v2);
}


int main() {
	const char **stdin_line = parse_stdin_lines();
	int _v0; // int32 a
	int _v1; // int32 b
	int _v2; // int32 res

	_v0 = parse_int32(stdin_line[0]);
	_v1 = parse_int32(stdin_line[1]);

	_cpf_auto_0(_v0, _v1, &_v2);

	write_line_int32(_v2);

	free(stdin_line);
	return 0;
}

```

Выбор из альтернатив и нефункциональные свойства



Ключевые идеи

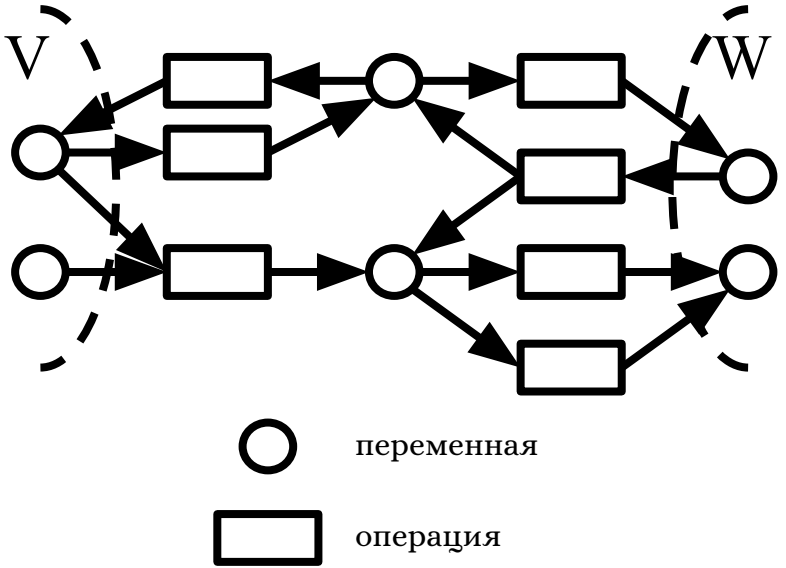
- Простая вычислительная модель (ВМ) – двудольный орграф, долями которого являются операции и переменные
- ВМ описывает величины предметной области и возможности вычислять одни из других готовыми модулями
- На ВМ ставится задача (множества V и W входных и выходных переменных)
- Система автоматически строит алгоритм и программу решения задачи



Так обеспечивается **активность** знаний – свойство их автоматического применения для решения новых задач без необходимости человека знать об их существовании.

Ключевые идеи

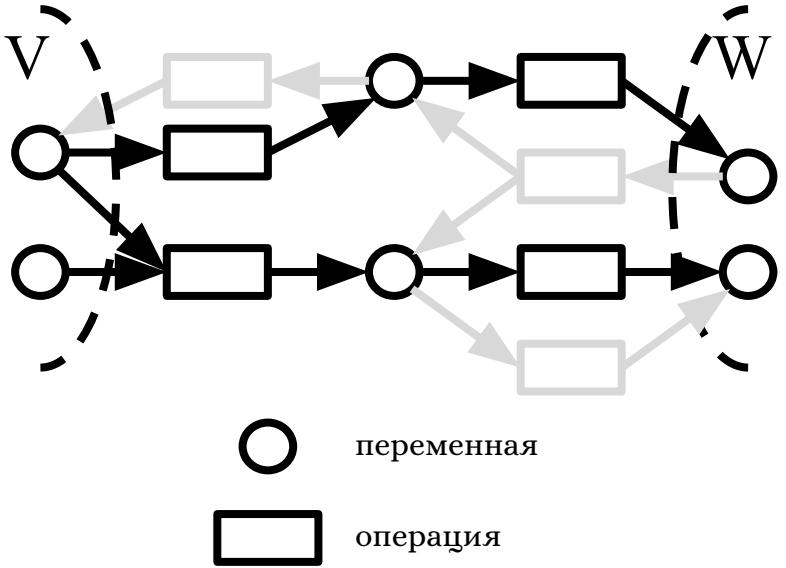
- Простая вычислительная модель (ВМ) – двудольный орграф, долями которого являются операции и переменные
- ВМ описывает величины предметной области и возможности вычислять одни из других готовыми модулями
- На ВМ ставится задача (множества V и W входных и выходных переменных)
- Система автоматически строит алгоритм и программу решения задачи



Так обеспечивается **активность** знаний – свойство их автоматического применения для решения новых задач без необходимости человека знать об их существовании.

Ключевые идеи

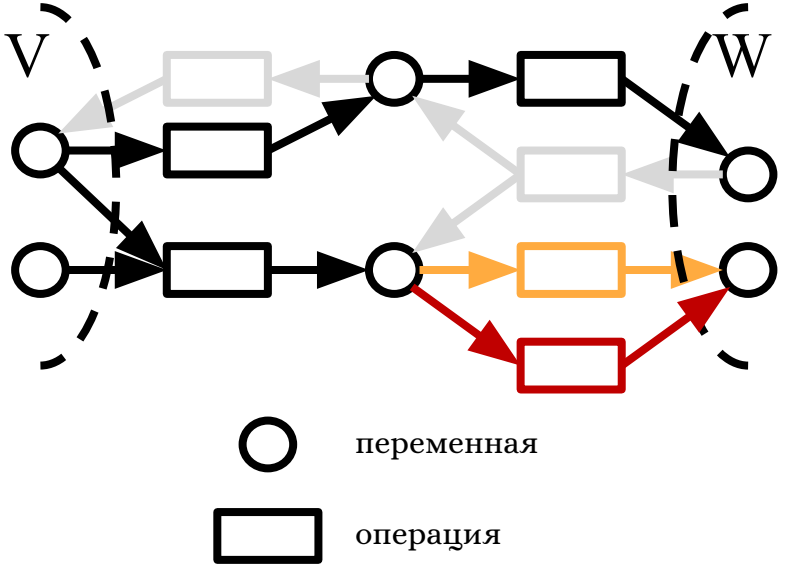
- Простая вычислительная модель (ВМ) – двудольный орграф, долями которого являются операции и переменные
- ВМ описывает величины предметной области и возможности вычислять одни из других готовыми модулями
- На ВМ ставится задача (множества V и W входных и выходных переменных)
- Система автоматически строит алгоритм и программу решения задачи



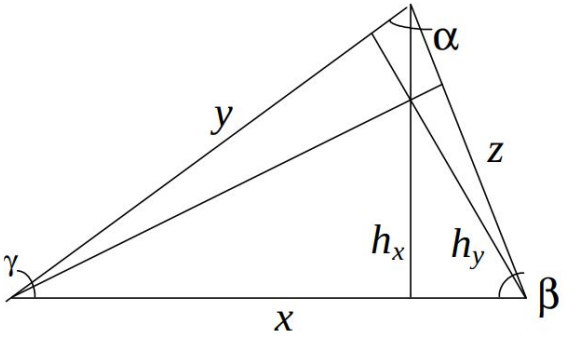
Так обеспечивается **активность** знаний – свойство их автоматического применения для решения новых задач без необходимости человека знать об их существовании.

Ключевые идеи

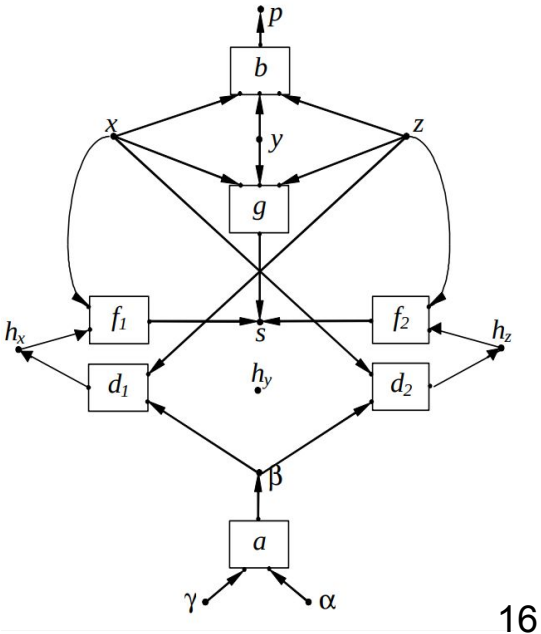
- Вычислительная модель – это способ задания **функциональной спецификации** как модулей, так и требуемой программы
- Если задать **нефункциональные свойства** модулей, операций и переменных, то можно ставить задачу оптимизации решения по нефункциональным критериям
  - Эффективность
  - Точность
  - …



Иллюстративный пример: треугольник из школьной геометрии

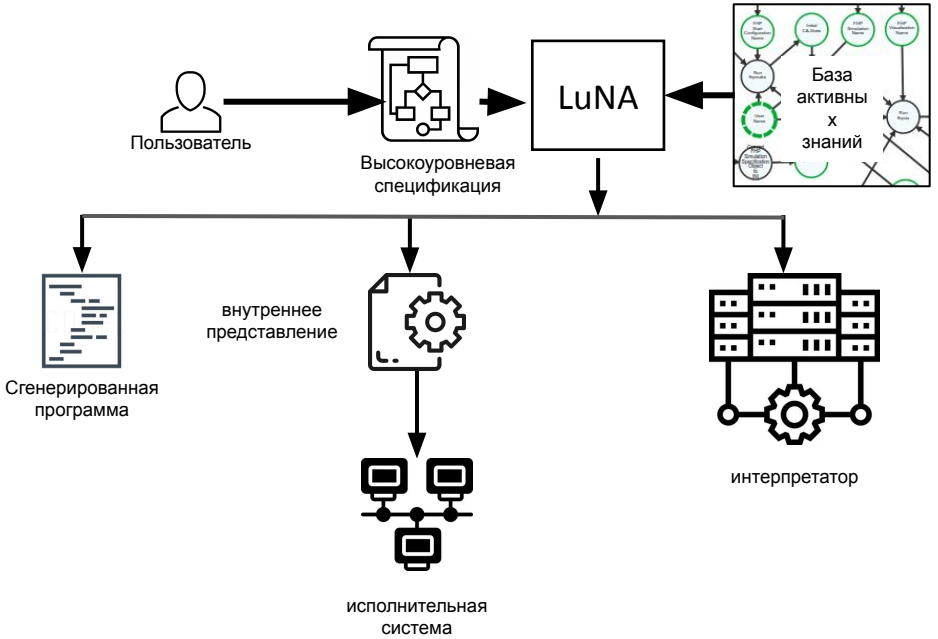


- Описать вычислительную модель
- Определить множества входных и выходных переменных (задачу)
- Вывести (оптимальный) алгоритм решения задачи
- Сгенерировать (эффективную) программу, реализующую алгоритм
- Задать значения входных переменных
- Исполнить программу



Язык и система **LuNA** автоматического конструирования программ (**L**ang**u**age for **N**umerical **M**odels)

- Разработка ИВМиМГ СО РАН
- Базируется на концепции активных знаний
- Разрабатывается сотрудниками, аспирантами и студентами
- Основные компоненты:
  - Язык LuNA
  - Транслятор
  - Генераторы программ
  - Исполнительные системы
  - Анализаторы и оптимизаторы
  - Средства отладки



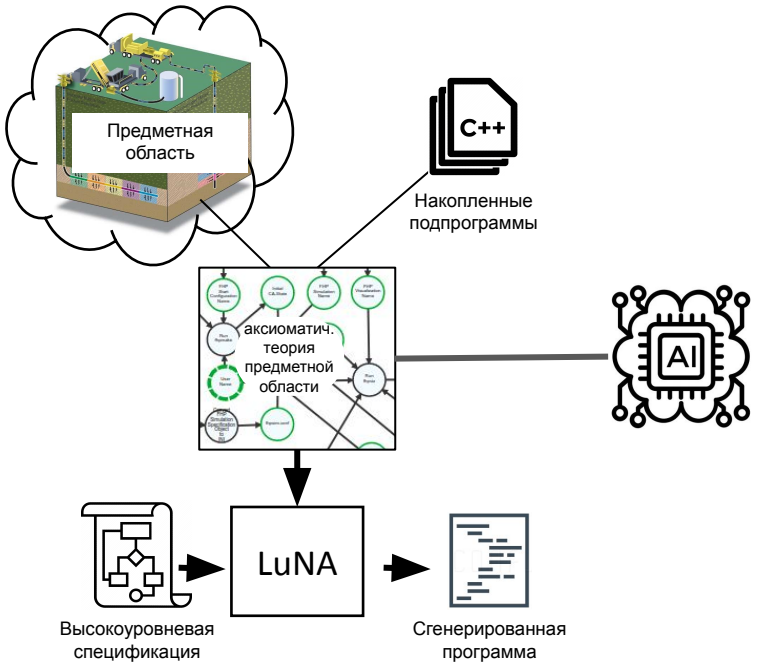
Сайт проекта: `https://ssd.sscc.ru/luna`

Преимущества подхода

- Пользователю не нужно знать о том, какие модули есть в базе активных знаний, и какие из них лучше использовать в конкретной ситуации.
- Применение модулей осуществляется в соответствии с особенностями и сложившейся практикой в этой предметной области.
- Система «видит», какая задача решается, на каком оборудовании и данных, и может адаптироваться.
- Прозрачность: человек может контролировать, какое решение было синтезировано, и при необходимости корректировать его.
- Функциональная спецификация задачи задаётся просто и практично — в виде композиции модулей (но решение может состоять из совсем других модулей).
- Есть дополнительные возможности отладки.
- Обеспечивается долгосрочное накопление и автоматическое переиспользование моделей алгоритмов и программ

Augmented LLM: LLM + база активных знаний

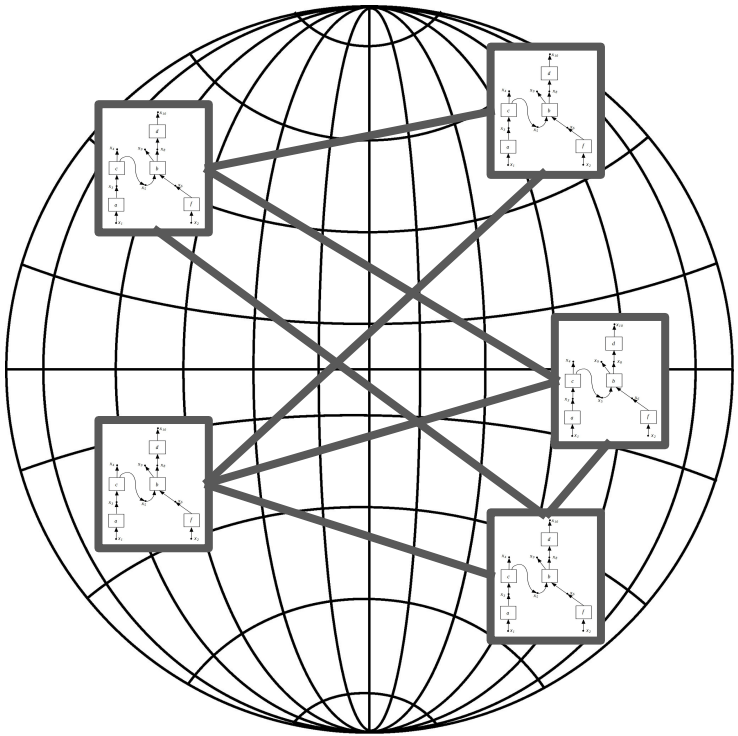
- Даже если LLM может посчитать 2+2, простой калькулятор для этого подойдёт лучше
- Методическое сопровождение пользователя
- Перевод постановки задачи из неформальной в формальную
- ИИ для решения оптимизационных задач
- LLM-анализ построенного решения
- Автоматическое конструирование БАЗ по неформальным текстам, обычным программам
- Объяснимый, безопасный, проверяемый ИИ
- Суррогатное моделирование функциональных и нефункциональных свойств
- Разметка входных и вычисляемых данных на основе ВМ



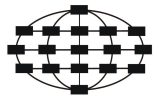
Долгосрочная перспектива

Направление работ: создание баз активных знаний в конкретных предметных областях (решение задач, накопление и переиспользование решений)

- Базы активных знаний естественным образом объединяются в сеть
- Эта сеть по своей природе — глобальная
- Создание такой сети неизбежно, вопрос лишь в том, какой она будет



Конец



г. Новосибирск
ИВМиМГ СО РАН
лаб. синтеза параллельных программ

Перепёлкин В.А.

perepelkin@ssd.sscc.ru