

Парадигмы программирования. Существуют ли монопарадигменные языки?

А.И. Легалов

Парадигмы программирования

1. Методы алгоритмизации (МА)

- императивное программирование
- функциональное программирование
- автоматное программирование
- декларативное программирование

2. Методы композиции (МК)

- абстрактные типы данных + функции
- классы = данные + методы
- модули
- пространства имен

3. Методы задания однозначности (МЗО)

- операционная однозначность (на уровне кодов операций компьютера — бестиповые ЯП)
- статическая типизация (однозначность на уровне описания типов данных — статически типизированные ЯП)
- динамическая типизация (однозначность на уровне вычисляемых тегов — динамически типизированные ЯП)

4. Методы управления вычислениями (МУВ)

- последовательное программирование
- параллельное программирования (разнообразие вариантов)

Парадигмы программирования

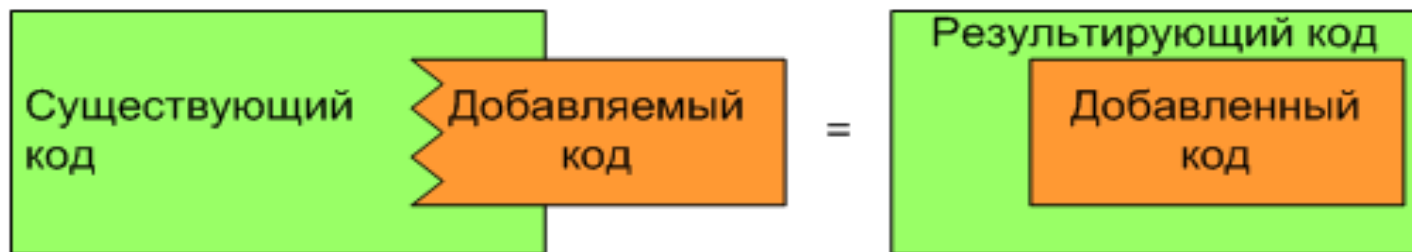
Парадигмы = |МА| * |МК| * |МЗО| * |МУВ| * |УА| * ...

Языки программирования: смесь парадигм

Парадигмы программирования

Методы композиции	Методы алгоритмизации			
	Императивный	Функциональный	...	Функционально- поточковый
Объектно- ориентированный стиль (классы)	Java, C#, ...	...	...	
	C++, Python, ...	Ocaml	...	
Процедурный стиль (АТД + функции)	C, Modula-2, Oberon-7, ...	Haskell, ML, ...	...	
Стиль Go (Структуры + интерфейсы)	Go, Rust	...	...	
...	...	...	...	
Процедурно- параметрический	!	+	...	!

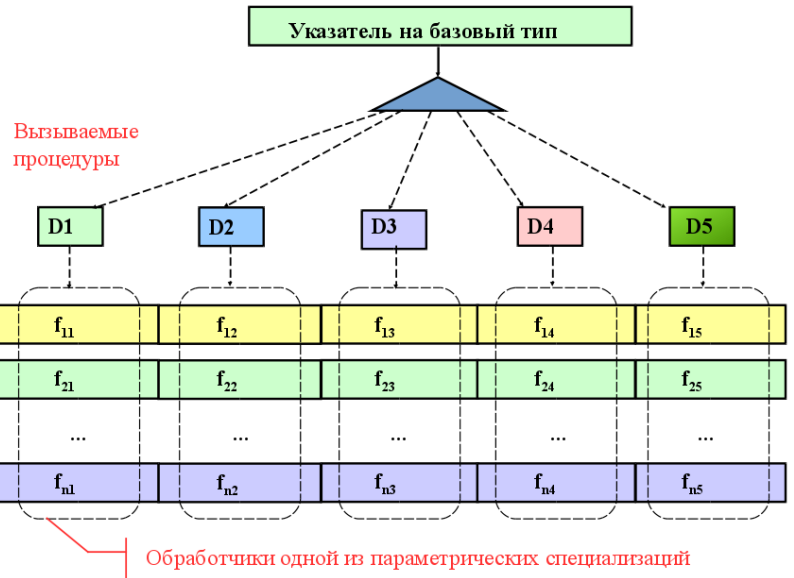
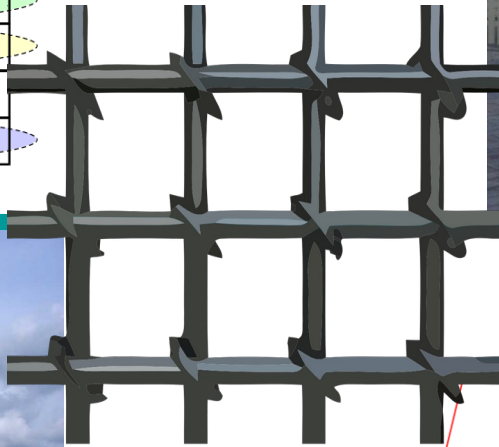
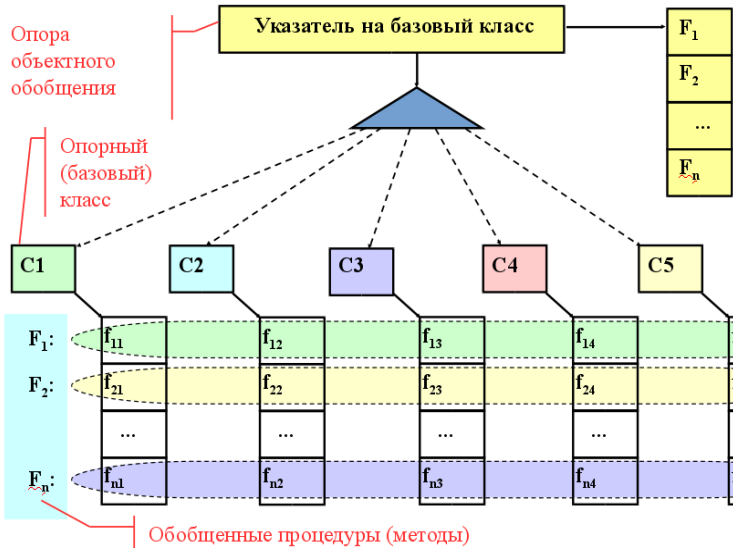
Общая Специфика эволюционного расширения

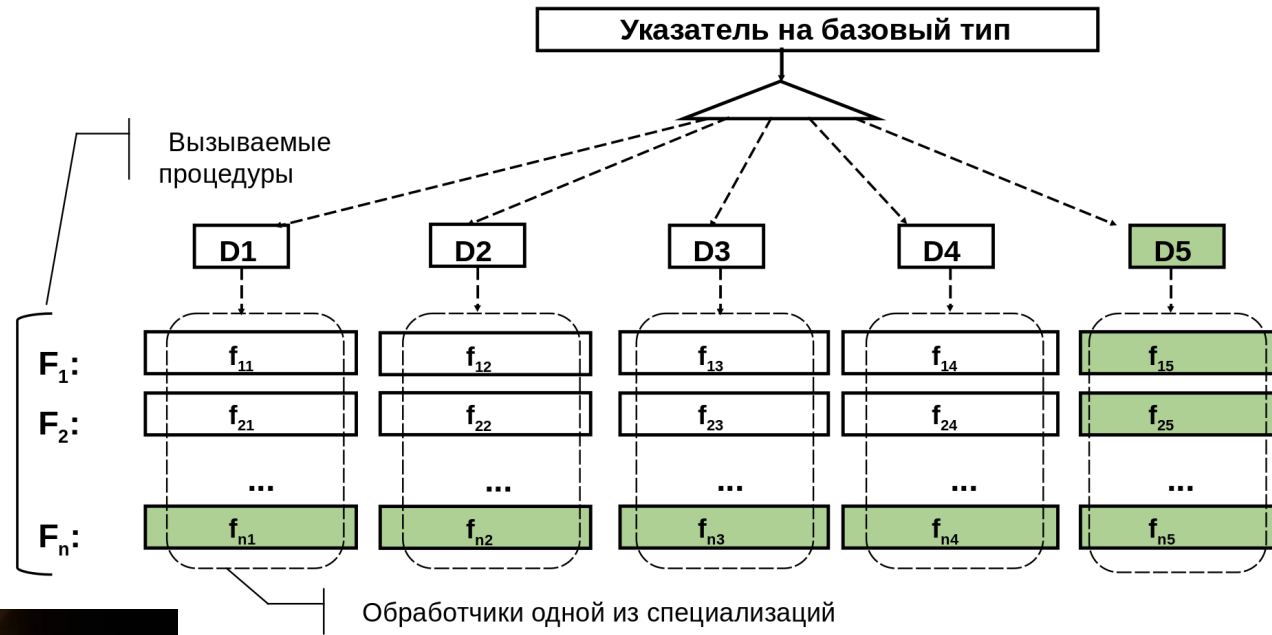


а) Изменение существующего кода



б) Эволюционное расширение





Процедурно-параметрическая парадигма

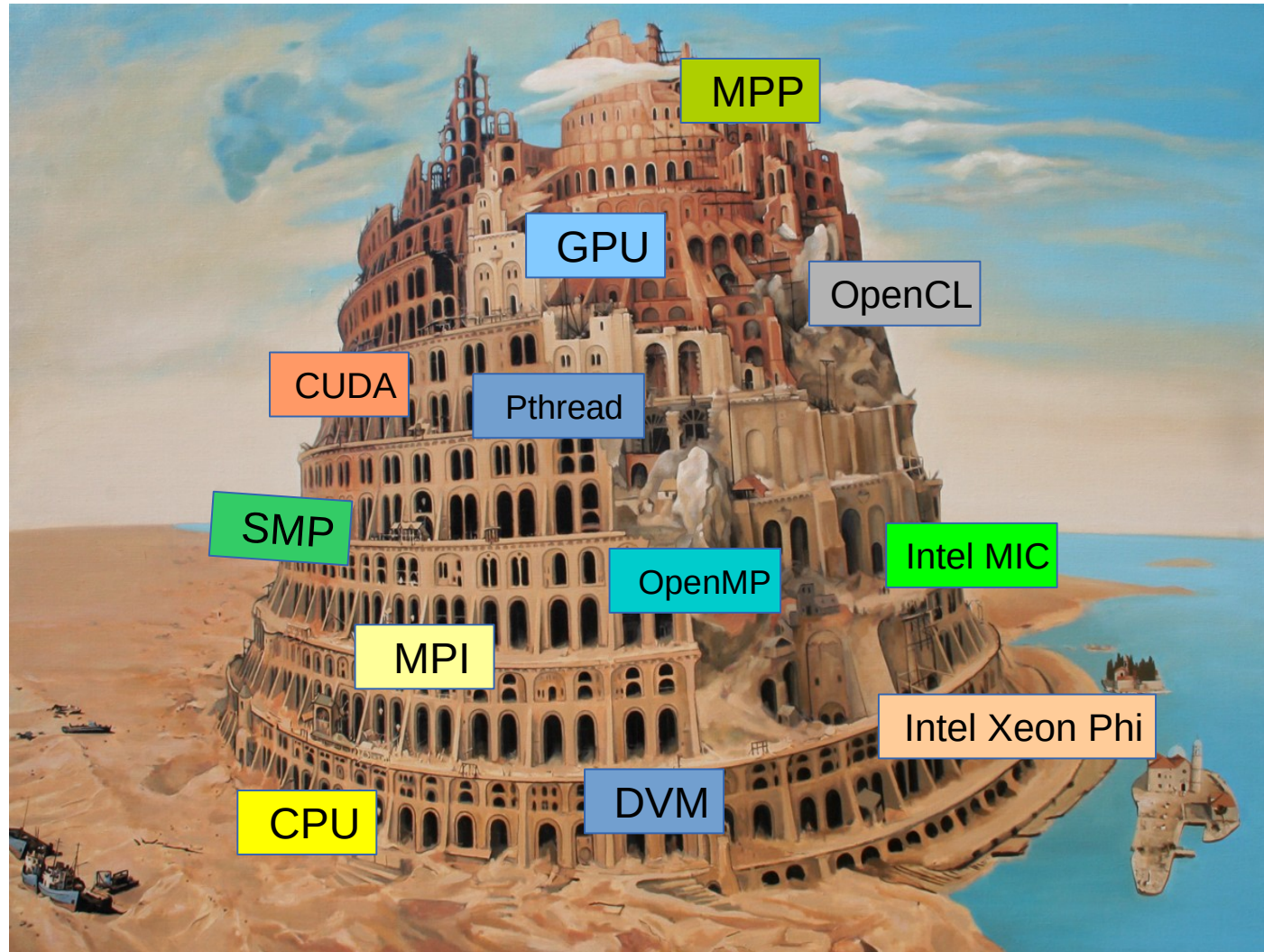
В отличие от концепции базового типа, расширяемого за счет добавления в производных типах, использование параметрического обобщения предполагает **сохранение исходного типа**, а альтернативные специализации вводятся как его **уточнения**.

Внешне все специализации имеют **единый тип**, а их разнообразное толкование используется только внутри него. Это позволяет **убрать глобальную идентификацию типов**, применяемую при расширении записей или наследовании, и обеспечивает поддержку концепции строгой типизации.

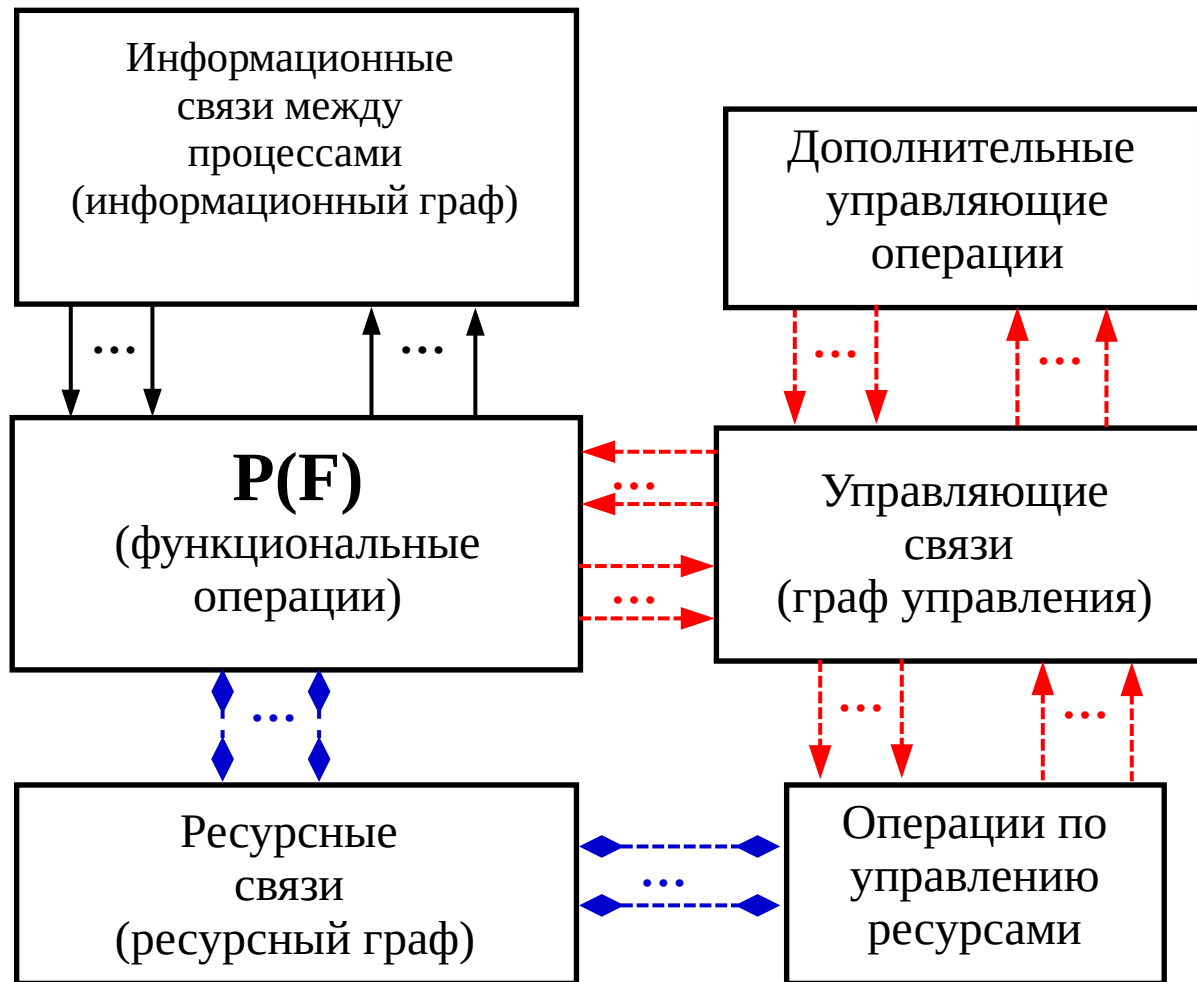
==> Возможность табличных обращений

Косвенная адресация --- Индексация

Мир параллельного программирования



Описание поведения процессов в ресурсах вычислительной системы



Стратегии управления в вычислительных системах

Условия готовности вычислений

Условие готовности данных

(Information, I-условие).

Перед запуском процесса на его информационных входах должны присутствовать все необходимые аргументы и функции, составляющие операционный пакет. Отсутствие каких-либо данных к моменту запуска процесса ведет к получению неправильного результата.

Условие выделения ресурсов

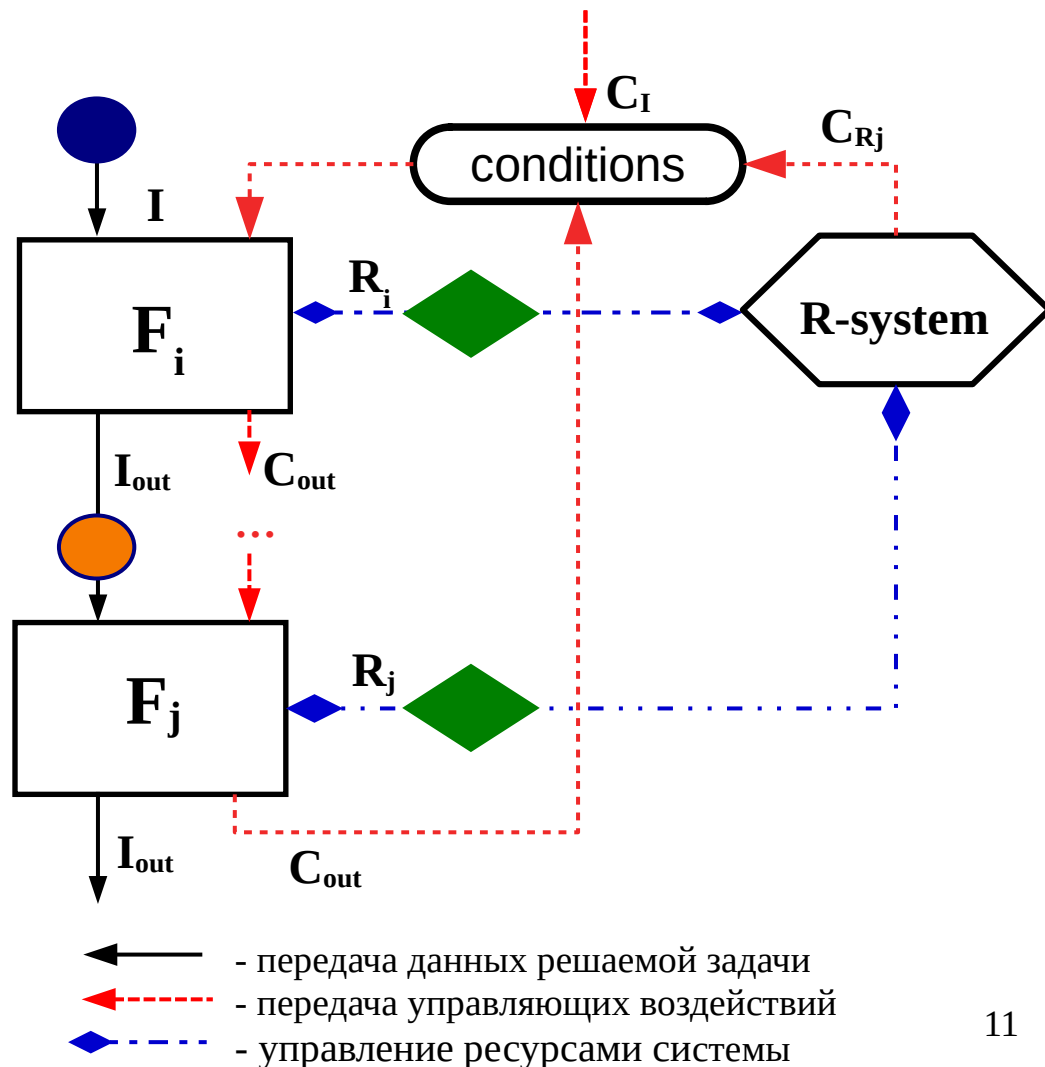
(Resource, R-условие).

Выполнение процесса протекает в соответствующих ресурсах, поэтому, перед его запуском, эти ресурсы должны быть определены и предоставлены.

Условие подтверждения

(Acknowledge, A-условие).

Ресурсы, занимаемые процессом, могут освобождаться или повторно использоваться только после того, как результаты вычислений будут использованы всеми процессами, являющихся приемниками этих результатов в качестве аргументов.



Стратегии управления в вычислительных системах

Способы задания управления

Явный (субъективный) (Subjective)

В ходе написания программы разработчик сам создает логику порождения и проверки условий готовности

Неявный

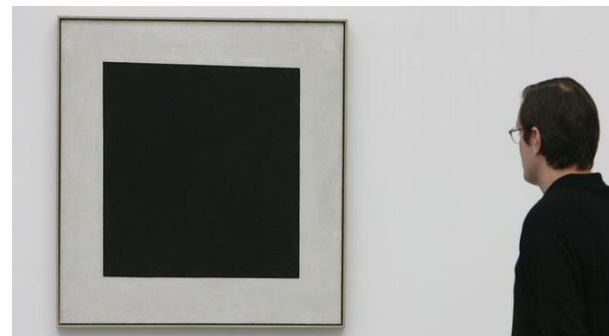
Управление тем или иным условием готовности не задается из расчета, что процессы и так будут выполняться корректно.

- Автоматический (Automatic)

Ресурсы ВС организованы таким образом, что всегда поддерживают выполнение заданного условия.

- Пустой (Empty)

Описание взаимодействия процессов может быть сделано так, что, несмотря на отсутствие в ВС механизма автоматической поддержки, условие готовности всегда истинно, поэтому его и не нужно проверять.



Стратегии управления в вычислительных системах

Множество условий готовности:

$$X = (\text{Information, Resource, Acknowledge})$$

Множество способов управления, обеспечивающих соблюдение условий готовности:

$$Y = \{\text{Subjective, Automatic, Empty}\}$$

$S_1 = \{\text{Information} \times Y\}$ – множество методов управления данными;

$S_2 = \{\text{Resource} \times Y\}$ – множество методов управления ресурсами;

$S_3 = \{\text{Acknowledge} \times Y\}$ – множество методов управления подтверждениями.

Множество стратегий управления в ВС:

$$S = \{S_1 \times S_2 \times S_3\} = 27$$

Стратегии управления в языках программирования

Множество условий готовности:

$$X = (\text{Information, Resource, Acknowledge})$$

Множество способов управления, обеспечивающих соблюдение условий готовности:

$$\begin{aligned} \text{Automatic} + \text{Empty} &= \text{Unevident} - \text{неявный} \\ Z &= \{\text{Subjective, Unevident}\} \end{aligned}$$

$S_1 = \{\text{Information} \times Z\}$ – множество методов управления данными;

$S_2 = \{\text{Resource} \times Z\}$ – множество методов управления ресурсами;

$S_3 = \{\text{Acknowledge} \times Z\}$ – множество методов управления подтверждениями.

Множество стратегий управления в ВС:

$$S = \{S_1 \times S_2 \times S_3\} = 8$$

Архитектурно-независимое параллельное программирование

Общая стратегия управления: Unevident Information, Unevident Resource, Unevident Acknowledge



*(Empty Information,
Empty Resource,
Empty Acknowledge)-машина*

Виртуальный программируемый вычислитель должен:

- обладать неограниченными вычислительными ресурсами;
- обеспечивать поддержку автоматического управления данными и неограниченными ресурсами (при этом предполагается, что ресурсы могут только предоставляться, но не забираться обратно);
- не поддерживать управление подтверждениями, обеспечивая, вместо этого, предоставление новых ресурсов каждому независимому набору данных.

То есть, поддерживать стратегию для систем типа:

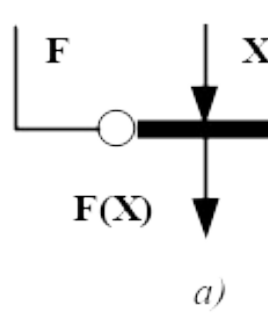
Automatic Information, Automatic Resource, Empty Acknowledge

Не искажать информационный граф программы!

Модель функционально-потокowych параллельных вычислений

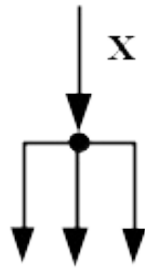
- Ориентация на бесконечные вычислительные ресурсы (принцип единственного использования вычислительных ресурсов).
- Запуск операции по готовности данных. Определяется в соответствии с аксиоматикой языка и его алгеброй преобразований (неявное управление вычислениями).
- Программа не имеет циклов, а значит и механизмов описания повторного использования ресурсов (все итеративные вычисления задаются через рекурсию).
- Для повышения эффективности при описании параллелизма используются специальные программно-формирующие структуры данных, определяемые как списки различного вида.
- Параллелизм на уровне элементарных операций.
- Программа – информационный граф.
- Выбор операций и аксиом, определяющих примитивы языка, ориентирован на наглядное текстовое представление информационного графа программы.

Operators



$X:F$ or $F^X \equiv F(X)$

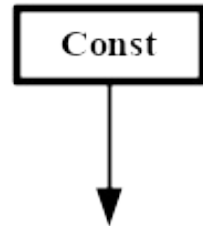
interpretation operator



value $\gg$ name, or
name $\ll$ value

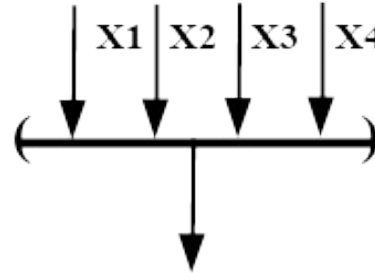
a)

*data
copying*



b)

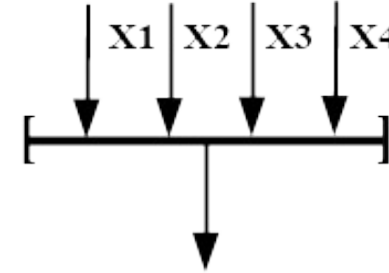
*constant
assignment*



$(X1, X2, X3, X4)$
 $(a,b):+ \equiv a+b$

c)

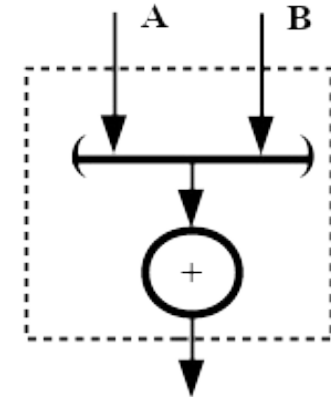
*grouping in
a data list*



$[X1, X2, X3, X4]$
 $[x,y]:\sin \equiv$
 $[x:\sin, y:\sin]$

d)

*grouping in
a parallel list*

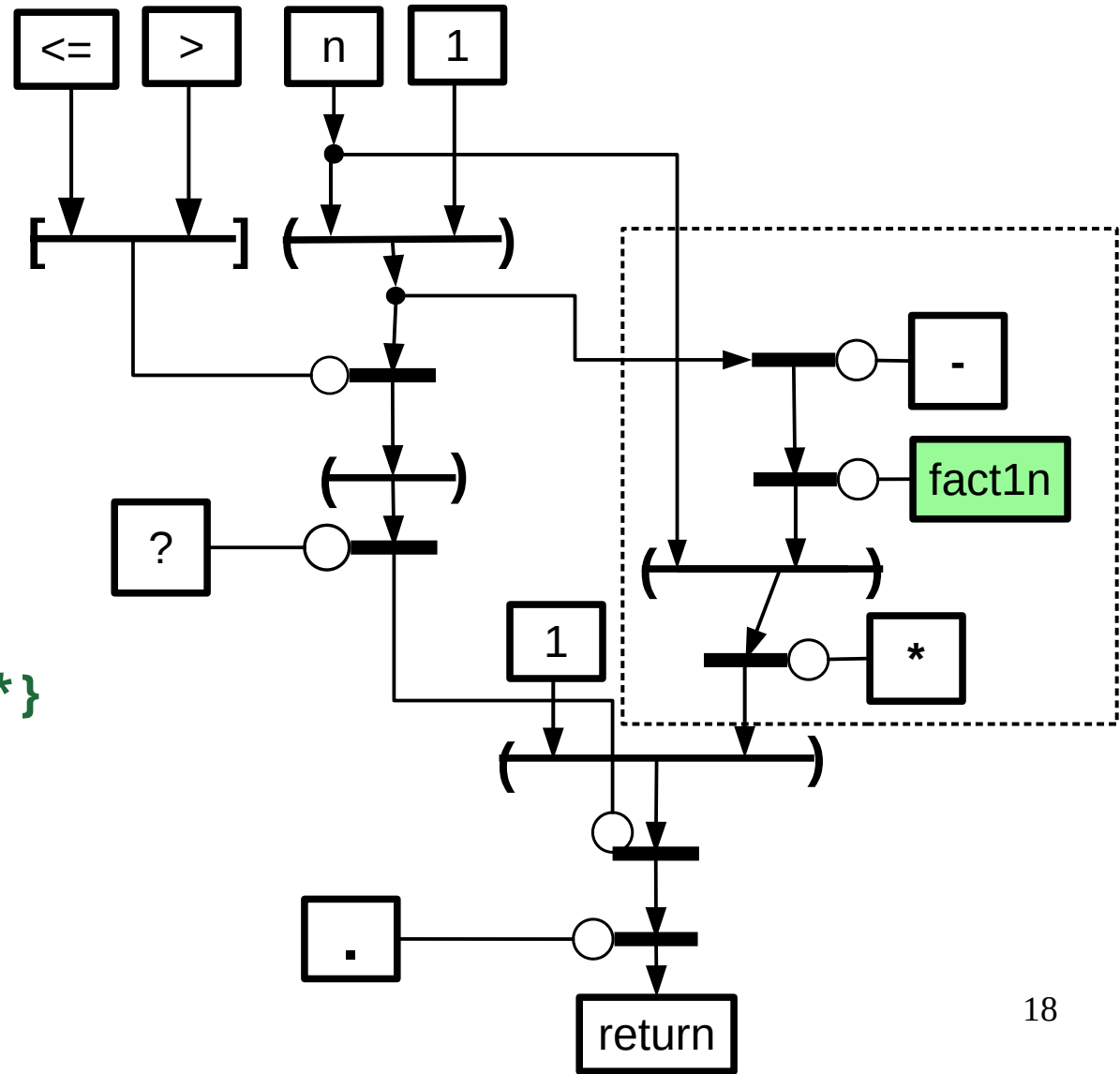


$\{ (A, B):+ \}$

e)

*delay list
creation*

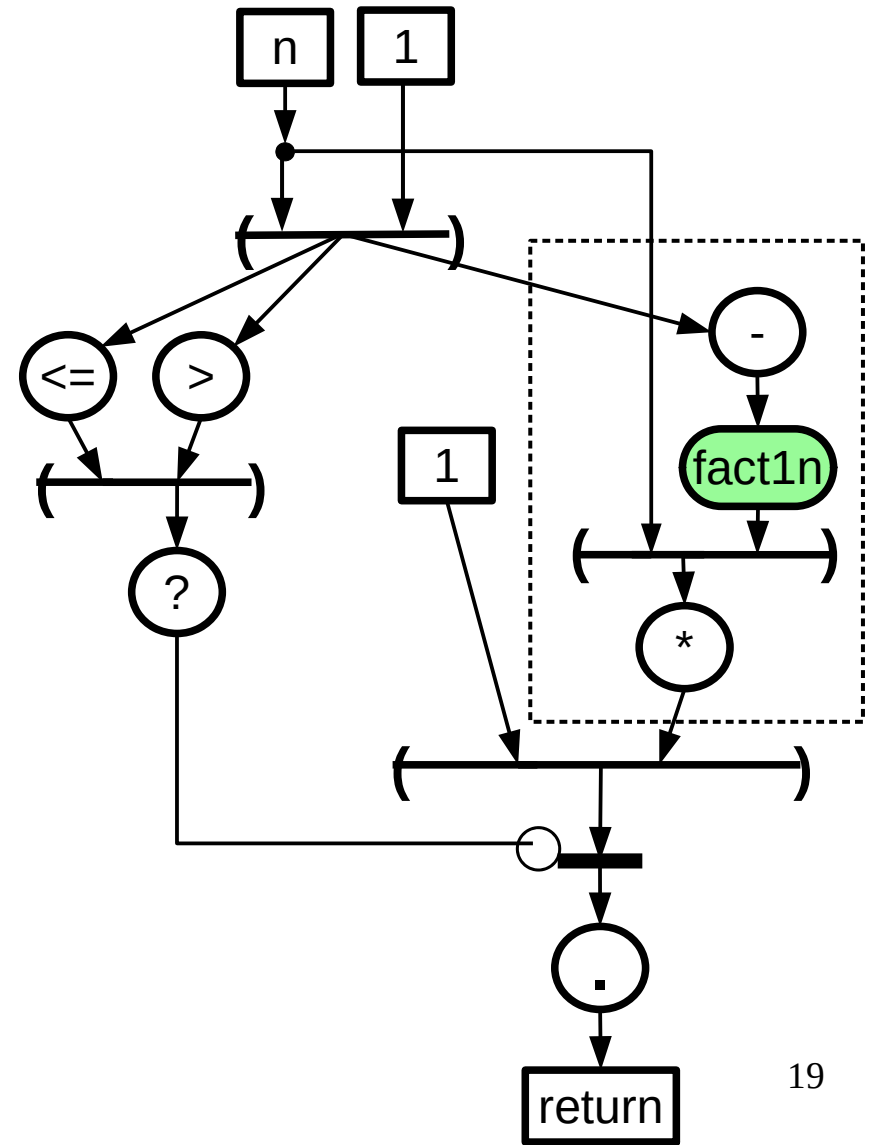
```
fact1n << funcdef n {
  n1<< (n,1);
  [(n1:[<=,>]):?]^
  (
    1,
    {(n, n1:-:fact1n):*}
  ):.
>>return
}
```

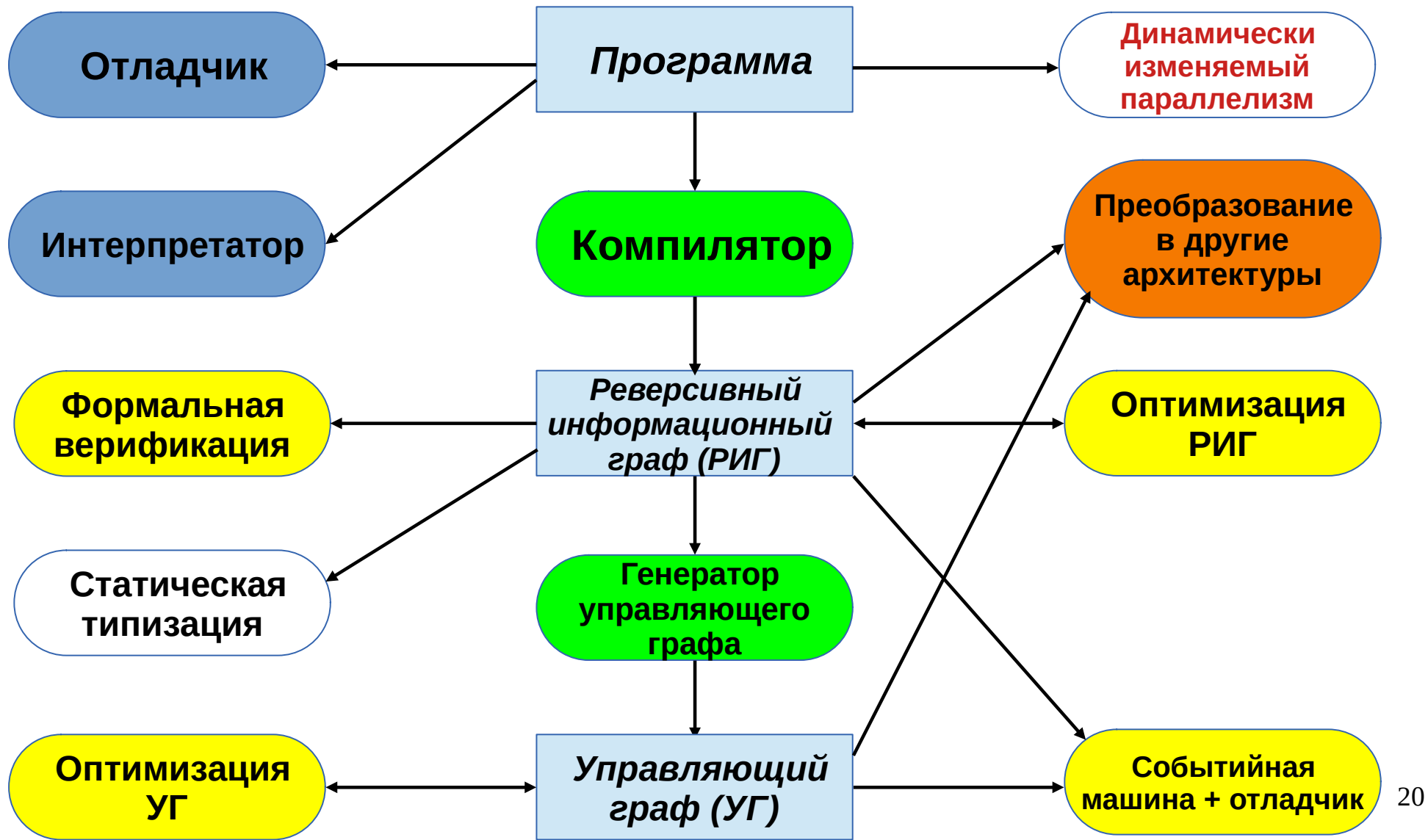


```

fact1n << funcdef n {
  n1<< (n,1);
  [(n1:[<=,>]):?]^
  (
    1,
    {(n, n1:-:fact1n):*}
  ):.
}>>return
}

```





Ссылки по теме

Стратегии управления вычислениями

<http://softcraft.ru/parallel/strat/>

**Различные аспекты функционально-потокowego
параллельного программирования:**

<http://softcraft.ru/fppp/>

Благодарю за внимание!