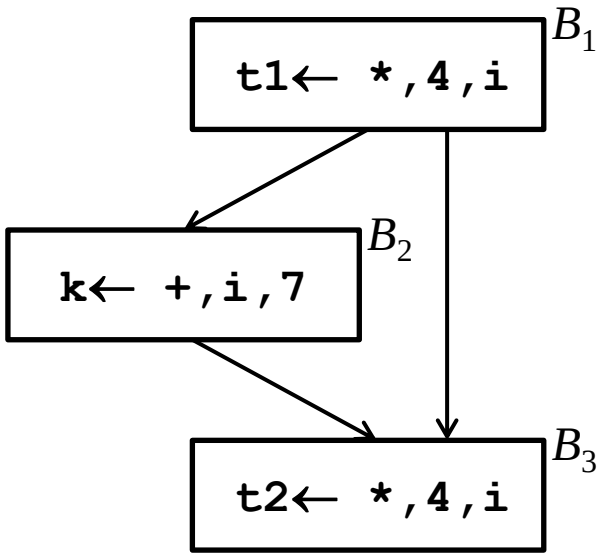
3. Анализ потока данных

3.1 Доступные выражения

3.1.1 Определение

◊ Выражение e *доступно* в точке p, если e вычисляется на любом пути от *Entry* до p причем переменные, входящие в состав e не переопределяются между последним таким вычислением и точкой p.



Пусть точка p – вход в блок B_3.

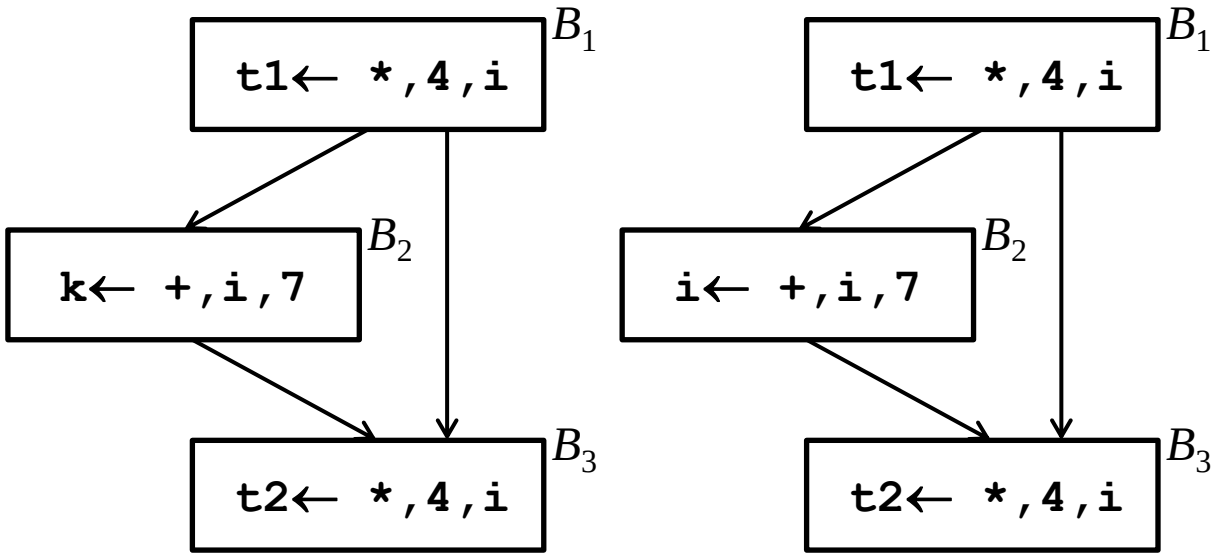
На рисунке присваивание *не убивает* выражение **`*,4,i`**.

Поэтому **`t2←*,4,i`** можно заменить на **`t2← t1`**

3.1 Доступные выражения

3.1.1 Определение

◊ Выражение e *доступно* в точке p, если e вычисляется на любом пути от $Entry$ до p причем переменные, входящие в состав e не переопределяются между последним таким вычислением и точкой p.



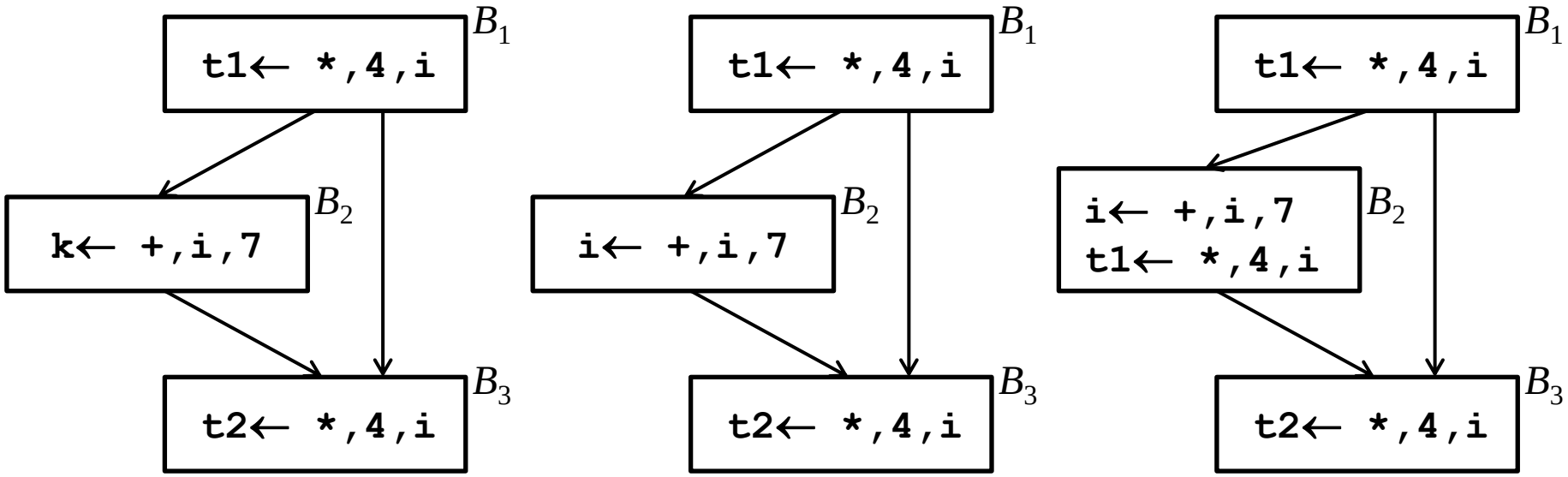
Точка p – вход в блок B_3.

На втором рисунке присваивание *убивает* выражение **`*,4,i`**, так как меняет значение **`i`**. Так что в этом случае замена **`t2←*,4,i`** на **`t2← t1`** будет некорректной.

3.1 Доступные выражения

3.1.1 Определение

◊ Выражение e *доступно* в точке p, если e вычисляется на любом пути от $Entry$ до p причем переменные, входящие в состав e не переопределяются между последним таким вычислением и точкой p.



Точка p – вход в блок B_3.

Если в блоке перевычислить выражение **`*,4,i`** после изменения **`i`**, замена **`t2←*,4,i`** на **`t2← t1`** после снова станет возможной.

3.1 Доступные выражения

3.1.1 Определение

◊ Для каждого базового блока B определим

- ◈ множество e_kill_B выражений, *убиваемых* в блоке B (пример – присваивание **`i← +,i,7`** в блоке B_2 убивает выражение **`*,4,i`**),
- ◈ множество e_gen_B выражений, *порождаемых* в блоке B (пример – присваивание **`t1← *,4,i`** в блоке B_2 порождает выражение **`*,4,i`**).

3.1 Доступные выражения

3.1.2 Уравнения потока данных

◊ Для того, чтобы найти доступные выражения, можно использовать метод, напоминающий метод вычисления достигающих определений.

- U – множество всех выражений программы.
- $In[B]$ – множество выражений из U, доступных на входе в B,
- $Out[B]$ – множество выражений из U, доступных на выходе из B.

Граничное условие:

$$Out[Entry] = \varnothing$$

Система уравнений:

$$Out[B] = e\_gen_B \cup (In[B] - e\_kill_B)$$

$$In[B] = \bigcap_{P \in Pred(B)} Out[P]$$

3.1 Доступные выражения

3.1.3 Итеративный алгоритм вычисления доступных выражений

◊ ***Алгоритм «Доступные выражения»***

- **Вход**: граф потока, в котором для каждого блока B вычислены e_genB и e_killB
- **Выход**: множества выражений, доступных на входе ($In[B]$) и выходе ($Out[B]$) каждого базового блока B графа потока. .
- **Метод**: выполнить следующую программу

$Out[Entry] = \varnothing;$

for (каждый базовый блок B, отличный от $Entry$) $Out[B] = U;$

while (внесены изменения в Out)

for (каждый базовый блок B, отличный от $Entry$) {

$$Out[B] = e\_gen_B \cup (In[B] - e\_kill_B)$$

$$In[B] = \bigcap_{P \in Pred(B)} Out[P]$$

}

3.2 Полурешетки

3.2.1 Анализ потока данных

◊ При анализе потока данных рассматриваются множества переменных для описания состояния и такие операции как *объединение* ($\cup$) и *пересечение* ($\cap$) множеств.

3.2 Полурешетки

3.2.1 Анализ потока данных

◊ При анализе потока данных рассматриваются множества переменных для описания состояния и такие операции как *объединение* ($\cup$) и *пересечение* ($\cap$) множеств.

◊ Свойства операций $\cup$ и $\cap$:

$A \cup A = A$	$A \cap A = A$	*(идемпотентность)*
$A \cup B = B \cup A$	$A \cap B = B \cap A$	*(коммутативность)*
$A \cup (B \cup C) = (A \cup B) \cup C)$	$A \cap (B \cap C) = (A \cap B) \cap C$	*(ассоциативность)*
Если $A \cup B = A$, то $B \subseteq A$	Если $A \cap B = A$, то $B \supseteq A$	отношение частичного порядка, связанное с операцией

3.2 Полурешетки

3.2.1 Анализ потока данных

◊ При анализе потока данных рассматриваются множества переменных для описания состояния и такие операции как *объединение* ($\cup$) и *пересечение* ($\cap$) множеств.

◊ Свойства операций $\cup$ и $\cap$:

$A \cup A = A$	$A \cap A = A$	*(идемпотентность)*
$A \cup B = B \cup A$	$A \cap B = B \cap A$	*(коммутативность)*
$A \cup (B \cup C) = (A \cup B) \cup C)$	$A \cap (B \cap C) = (A \cap B) \cap C$	*(ассоциативность)*
Если $A \cup B = A$, то $B \subseteq A$	Если $A \cap B = A$, то $B \supseteq A$	отношение частичного порядка, связанное с операцией

◊ Если множество U содержит все элементы, то любое множество $A \subseteq U$ и

$$A \cup U = U \cup A = U$$

◊ Для пересечения ($\cap$) роль U играет $\varnothing$: любое множество $A \supseteq \varnothing$ и

$$A \cap \varnothing = \varnothing \cap A = \varnothing$$

3.2 Полурешетки

3.2.2 Определение полурешетки

◊ Полурешетка – это абстрактная алгебраическая структура, над элементами которой определена абстрактная операция $\wedge$ (мы будем называть ее «сбор»), обладающая свойствами операций $\cup$ и $\cap$.

◊ **Определение.** *Полурешетка* представляет собой множество L, на котором определена бинарная операция «*сбор*» $\wedge$, такая, что для всех x, y и $z \in L$:

$$x \wedge x = x \qquad (\textit{идемпотентность})$$

$$x \wedge y = y \wedge x \qquad (\textit{коммутативность})$$

$$x \wedge (y \wedge z) = (x \wedge y) \wedge z \qquad (\textit{ассоциативность})$$

Полурешетка имеет *верхний элемент* (или *верх*) $\mathsf{T} \in L$ такой, что для всех $x \in L$ выполняется $\mathsf{T} \wedge x = x$

3.2 Полурешетки

3.2.3 Решеточное отношение частичного порядка $\leq$

◊ Для всех пар $x, y \in L$ определим отношение $\leq$:

$x \leq y$ тогда и только тогда, когда $x \wedge y = x$

◊ Отношение $\leq$ является отношением частичного порядка.

(1) *Рефлексивность* $\leq$ следует из идемпотентности $\wedge$:

$x \leq x \Leftrightarrow x \wedge x = x$

(2) *Антисимметричность* $\leq$ следует из коммутативности $\wedge$:

пусть $x \leq y$ и $y \leq x$; тогда $x = x \wedge y = y \wedge x = y$

(3) *Транзитивность* $\leq$ следует из ассоциативности $\wedge$:

пусть $x \leq y$ и $y \leq z$; тогда по определению $\leq$

$x \wedge y = x$ и $y \wedge z = y$;

$(x \wedge z) = ((x \wedge y) \wedge z) = (x \wedge (y \wedge z)) - (x \wedge y) = x$,

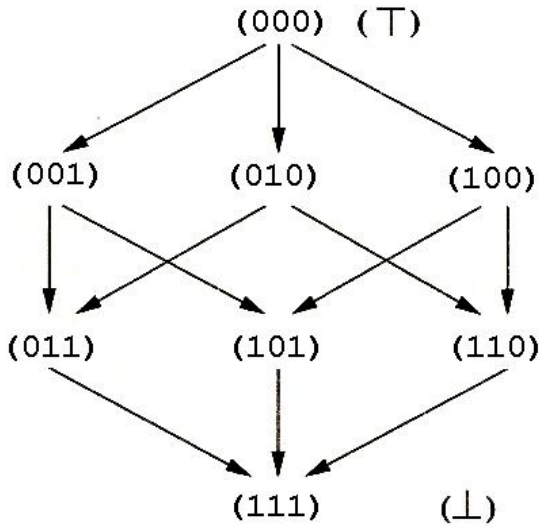
$(x \wedge z) = x \Leftrightarrow x \leq z$.

3.2 Полурешетки

3.2.4. Диаграммы полурешеток

◊ *Диаграмма полурешетки* $\langle L, \wedge \rangle$ представляет собой граф, узлами которого являются элементы L, а ребра направлены от x к y, если $y \leq x$.

◊ **Пример**. Диаграмма полурешетки $\langle U, \cup \rangle$, $|U| = 8$:
элемент множества U представляется битовым 3-вектором.



3.3 Структура потока данных

3.3.1 Терминология

◊ **Определение**. *Структурой потока данных* называется четверка

$$\langle D, F, L, \wedge \rangle ,$$

где

- D – направление анализа (*Forward* или *Backward*),
- F – семейство передаточных функций,
- L – поток данных,
- $\wedge$ - операция сбора.

◊ **Определение**. Семейство передаточных функций F называется *замкнутым*, если:

- F содержит тождественную функцию I: $\forall x \in L: \; I(x) = x$.
- F замкнуто относительно композиции:

$$\forall\, f, g \in F \Rightarrow h(x) = g(f(x)) \in F.$$

3.3.2 Замкнутость

◊ **Утверждение 1**. Семейство передаточных функций F, используемое при анализе достигающих определений (передаточные функции вида gen-$kill$), является замкнутым.

3.3.2 Замкнутость

◊ **Утверждение 1**. Семейство передаточных функций F, используемое при анализе достигающих определений (передаточные функции вида $gen\text{-}kill$), является замкнутым.

1) Замкнутость относительно композиции уже установлена.

2) Тождественная функция $I(x) = x$ является функцией вида $gen\text{-}kill$ с $gen = kill = \varnothing$.

3.3.2 Замкнутость

◊ **Утверждение 1**. Семейство передаточных функций F, используемое при анализе достигающих определений (передаточные функции вида gen-$kill$), является замкнутым.

1) Замкнутость относительно композиции уже установлена.

2) Тождественная функция $I(x) = x$ является функцией вида gen-$kill$ с $gen = kill = \varnothing$.

◊ **Утверждение 2**. Семейство передаточных функций, используемое при анализе живых переменных является замкнутым.

◊ **Утверждение 3**. Семейство передаточных функций, используемое при анализе доступных выражений является замкнутым.

3.3 Структура потока данных

3.3.3 Монотонные структуры

◊ **Определение 1**. Структура потока данных $\langle D, F, L, \wedge \rangle$ называется *монотонной,* если $\forall x, y \in L, \forall f \in F \; (x \le y) \Rightarrow f(x) \le f(y)$.

◊ **Определение 2**. Структура потока данных $\langle D, F, L, \wedge \rangle$ называется *монотонной,* если $\forall x, y \in L, \forall f \in F \;\; f(x \wedge y) \le f(x) \wedge f(y)$.

3.3 Структура потока данных

3.3.3 Монотонные структуры

◊ **Определение 1**. Структура потока данных $\langle D, F, L, \wedge \rangle$ называется *монотонной,* если $\forall x, y \in L, \forall f \in F\ (x \le y) \Rightarrow f(x) \le f(y)$.

◊ **Определение 2**. Структура потока данных $\langle D, F, L, \wedge \rangle$ называется *монотонной,* если $\forall x, y \in L, \forall f \in F\ \ f(x \wedge y) \le f(x) \wedge f(y)$.

◊ **Утверждение.** Определения 3.3.3.1 и 3.3.3.2 эквивалентны.

3.3.3 Монотонные структуры

◊ **Определение 1**. Структура потока данных $\langle D, F, L, \wedge \rangle$ называется *монотонной,* если $\forall x, y \in L, \forall f \in F\ (x \le y) \Rightarrow f(x) \le f(y).$

◊ **Определение 2**. Структура потока данных $\langle D, F, L, \wedge \rangle$ называется *монотонной,* если $\forall x, y \in L, \forall f \in F\ \ f(x \wedge y) \le f(x) \wedge f(y).$

◊ **Утверждение.** Определения 3.2.3.1 и 3.2.3.2 эквивалентны.

⧫ Из определения 1 следует определение 2:

$x \wedge y = inf(x, y) \Rightarrow x \wedge y \le x$ и $x \wedge y \le y \Rightarrow (1) \Rightarrow$

$f(x \wedge y) \le f(x)$ *и* $f(x \wedge y) \le f(y) \Rightarrow$

$f(x \wedge y) = inf(f(x), f(y)) = f(x) \wedge f(y)$

3.3.3 Монотонные структуры

◊ **Определение 1**. Структура потока данных $\langle D, F, L, \wedge \rangle$ называется *монотонной,* если $\forall x, y \in L, \forall f \in F\ (x \le y) \Rightarrow f(x) \le f(y).$

◊ **Определение 2**. Структура потока данных $\langle D, F, L, \wedge \rangle$ называется *монотонной,* если $\forall x, y \in L, \forall f \in F\ \ f(x \wedge y) \le f(x) \wedge f(y).$

◊ **Утверждение.** Определения 3.2.3.1 и 3.2.3.2 эквивалентны.

◈ Из определения 2 следует определение 1:

$$x \le y \Rightarrow x \wedge y = x \Rightarrow_{(\text{опр2})} f(x) \le f(x) \wedge f(y),$$

$$f(x) \wedge f(y) = inf(f(x), f(y)) \le f(y); \Rightarrow f(x) \le f(y)$$

3.3.4 Дистрибутивные структуры

◊ **Определение**. Структура потока данных $\langle D, F, L, \wedge \rangle$ называется *дистрибутивной,* если

$$\forall x, y \in L, \forall f \in F: \quad f(x \wedge y) = f(x) \wedge f(y).$$

3.3.4 Дистрибутивные структуры

◊ **Определение**. Структура потока данных $\langle D, F, L, \wedge \rangle$ называется *дистрибутивной,* если

$$\forall x, y \in L, \forall f \in F: \quad f(x \wedge y) = f(x) \wedge f(y).$$

◊ **Утверждение.** Если структура потока данных $\langle D, F, L, \wedge \rangle$ дистрибутивна, то она монотонна.

$$a = b \Rightarrow_{\text{идемпотентность}} a \wedge b = a \Rightarrow a \leq b$$

$$f(x \wedge y) = f(x) \wedge f(y) \Rightarrow \; f(x \wedge y) \leq f(x) \wedge f(y)$$

◊ Обратное утверждение неверно.

В качестве доказательства можно привести пример монотонной структуры потока данных, которая не дистрибутивна.

3.3.5. Дистрибутивность структуры достигающих определений

◊ **Утверждение.** Структура достигающих определений $RD = \langle Forward, Gen\text{-}kill, \varnothing, \cup \rangle$ дистрибутивна

◊ $y, z \in RD, f(x) = G \cup (x - K) \in Gen\text{-}kill.$

Докажем, что

$$G \cup ((y \cup z) - K) = (G \cup (y - K)) \cup (G \cup (z - K))$$

(1) $y, z \in G$: равенство выполняется, так как G входит

и в левую, и в правую его части.

(2) $y, z \notin G$: G можно исключить из равенства:

$$(y \cup z) - K = (y - K)) \cup (z - K)$$

Это равенство проверяется при помощи диаграмм Венна.

3.3.5. Дистрибутивность структуры достигающих определений

◊ **Утверждение.** Структура достигающих определений $RD = \langle Forward, Gen\text{-}kill, \varnothing, \cup \rangle$ дистрибутивна

◊ $y, z \in RD, f(x) = G \cup (x - K) \in Gen\text{-}kill.$

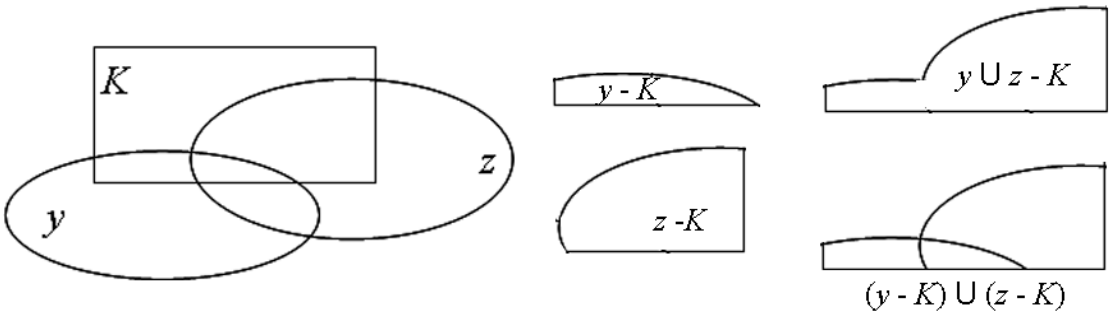
Докажем, что

$$G \cup ((y \cup z) - K) = (G \cup (y - K)) \cup (G \cup (z - K))$$

(2) $y, z \notin G$: G можно исключить из равенства:

$$(y \cup z) - K = (y - K)) \cup (z - K)$$

Это равенство проверяется при помощи диаграмм Венна.



3.3 Структура потока данных

3.3.6. Дистрибутивность структур живых переменных и доступных выражений

◊ **Следствие 1.** Структура живых переменных

$$LV = \langle Backward, Def\text{-}use, \varnothing, \cup \rangle$$

дистрибутивна

◊ $f(x) \in Def\text{-}use$ алгебраически подобна функции класса $Gen\text{-}kill$.

◊ **Следствие 2.** Структура доступных выражений

$$AE = \langle Forward, Gen\text{-}kill, U, \cap \rangle$$

дистрибутивна

3.4 Обобщенный итеративный алгоритм

3.4.1 Описание алгоритма

◊ **Алгоритм.** Итеративное решение задачи анализа потока данных

- **Вход**: граф потока управления,
  структура потока данных $\langle D, F, L, \wedge \rangle$,
  передаточная функция $f_B \in F$
  константа из $l \in L$ для граничного условия
- **Выход**: значения из L для $In[B]$ и $Out[B]$ для каждого блока B в графе потока.
- **Метод**: если $D = Forward$ выполнить программу (3.4.2);
  если $D = Backward$ выполнить программу (3.4.3).

3.4 Обобщенный итеративный алгоритм

3.4.2 Решение задачи потока данных ($D = Forward$)

$Out[Entry] = l;$

for ($each\ B \neq Entry$) $Out[B] = \top;$

while (внесены изменения в Out)

for ($each\ B \neq Entry$) {

$$In[B] = \bigwedge_{P \in Pred(B)} Out[P]$$

$$Out[B] = f_B(In[B])$$

}

3.4 Обобщенный итеративный алгоритм

3.4.3 Решение задачи потока данных ($D = Backward$)

$In[Exit] = l;$

for ($each\ B \neq Exit$) $In[B] = \top;$

while (внесены изменения в In)

for ($each\ B \neq Exit$) {

$$Out[B] = \bigwedge_{S \in Succ(B)} In[S]$$

$$In[B] = f_B(Out[B])$$

}

3.5 Свойства итеративного алгоритма

3.5.1 Сходимость к решению

◊ **Утверждение.** Если обобщенный итеративный алгоритм сходится, то получающийся результат является решением уравнений потоков данных.

◈ $D = Forward:$

Если после очередной итерации цикла **`while`** хотя бы для одного B уравнение $Out[B] = f_B(In[B])$ не удовлетворяется, то для этого B

$$OutNew[B] \neq Out[B].$$

Следовательно, в множество $Out[B]$ будет внесено изменение и $change$ не позволит выйти из цикла.

3.5 Свойства итеративного алгоритма

3.5.1 Сходимость к решению

◊ **Утверждение.** Если обобщенный итеративный алгоритм сходится, то получающийся результат является решением уравнений потоков данных.

◈ $D = Backward:$

Аналогичные рассуждения, только вместо множества $Out[B]$ рассматривается множество $In[B]$.

3.5 Свойства итеративного алгоритма

3.5.2 Максимальная фиксированная точка

◊ **Определение**. *Максимальная фиксированная точка* системы уравнений

$$In[B] = \bigwedge_{P \in Pred(B)} Out[P]$$

$$Out[B] = f_B(In[B])$$

представляет собой решение $\{In[B_i]^{max}, Out[B_i]^{max}\}$ этой системы, обладающее тем свойством, что для любого другого решения $\{In[B_i], Out[B_i]\}$ выполняются условия

$$In\ [B_i] \le In[B_i]^{max} \text{ и } Out[B_i] \le Out[B_i]^{max}$$

где $\le$ – полурешеточное отношение частичного порядка.

3.5 Свойства итеративного алгоритма

3.5.3 Монотонность итераций

◊ **Утверждение 1.** Пусть $In[B]^i$ и $Out[B]^i$ – значения $In[B]$ и $Out[B]$ после i-ой итерации. Если структура потока данных монотонна, то

$$In[B]^{i+1} \leq In[B]^i \text{ и } Out[B]^{i+1} \leq Out[B]^i$$

◈ Индукция по i.

3.5 Свойства итеративного алгоритма

3.5.3 Монотонность итераций

◊ **Утверждение 1.** Пусть $In[B]^i$ и $Out[B]^i$ – значения $In[B]$ и $Out[B]$ после i-ой итерации. Если структура потока данных монотонна, то

$$In[B]^{i+1} \le In[B]^i \text{ и } Out[B]^{i+1} \le Out[B]^i$$

◈ Индукция по i.

◈ ***Основание***:

$$In[B]^1 \le In[B]^0 \quad \text{и}$$

$$Out[B]^1 \le Out[B]^0,$$

так как $\forall B \ne Entry$: $In[B]^0 = \top$ и $Out[B]^0 = \top$.

3.5 Свойства итеративного алгоритма

3.5.3 Монотонность итераций

◊ **Утверждение 1.** Пусть $In[B]^i$ и $Out[B]^i$ – значения $In[B]$ и $Out[B]$ после i-ой итерации. Если структура потока данных монотонна, то

$$In[B]^{i+1} \leq In[B]^i \text{ и } Out[B]^{i+1} \leq Out[B]^i$$

- ◈ Индукция по i.
- ◈ ***Шаг***: Пусть $In[B]^k \leq In[B]^{k-1}$ и $Out[B]^k \leq Out[B]^{k-1}$

3.5 Свойства итеративного алгоритма

3.5.3 Монотонность итераций

◊ **Утверждение 1.** Пусть $In[B]^i$ и $Out[B]^i$ – значения $In[B]$ и $Out[B]$ после i-ой итерации. Если структура потока данных монотонна, то

$$In[B]^{i+1} \le In[B]^i \text{ и } Out[B]^{i+1} \le Out[B]^i$$

⧫ Индукция по i.

⧫ ***Шаг***: Пусть $In[B]^k \le In[B]^{k-1}$ и $Out[B]^k \le Out[B]^{k-1}$

$$In[B]^{k+1} = \bigwedge_{P \in Pred(B)} Out[P]^k \le \bigwedge_{P \in Pred(B)} Out[P]^{k-1} = In[B]^k$$

так как операция сбора монотонна.

3.5 Свойства итеративного алгоритма

3.5.3 Монотонность итераций

◊ **Утверждение 1.** Пусть $In[B]^i$ и $Out[B]^i$ – значения $In[B]$ и $Out[B]$ после i-ой итерации. Если структура потока данных монотонна, то

$$In[B]^{i+1} \le In[B]^i \text{ и } Out[B]^{i+1} \le Out[B]^i$$

⧫ Индукция по i.

⧫ ***Шаг***: Пусть $In[B]^k \le In[B]^{k-1}$ и $Out[B]^k \le Out[B]^{k-1}$

$$In[B]^{k+1} = \bigwedge_{P \in Pred(B)} Out[P]^k \le \bigwedge_{P \in Pred(B)} Out[P]^{k-1} = In[B]^k$$

так как операция сбора монотонна.

$$Out[B]^{k+1} = f_B(In[B]^{k+1}) \le f_B(In[B]^k) = Out[B]^k,$$

так как передаточная функция $f_B(x)$ монотонна.

3.5 Свойства итеративного алгоритма

3.5.3 Монотонность итераций

◊ **Утверждение 1.** Пусть $In[B]^i$ и $Out[B]^i$ – значения $In[B]$ и $Out[B]$ после i-ой итерации. Если структура потока данных монотонна, то

$$In[B]^{i+1} \leq In[B]^i \text{ и } Out[B]^{i+1} \leq Out[B]^i$$

◊ **Следствие**. Для любого i

$$In[B] \leq In[B]^i$$

$$Out[B] \leq Out[B]^i$$

3.5 Свойства итеративного алгоритма

3.5.3 Монотонность итераций

◊ **Утверждение 2.** Если структура потока данных монотонна, то решение системы уравнений (1), найденное с помощью итеративного алгоритма, является максимальной фиксированной точкой этой системы.

◊ Требуется доказать, что для всех j

$$In[B_j] \le In[B_j]^{IA},\ Out[B_j] \le Out[B_j]^{IA},$$

где $\{In[B_j]^{IA}, Out[B_j]^{IA}\}$ – решение системы (1), найденное с помощью итеративного алгоритма, $\{In[B_j], Out[B_j]\}$ – любое другое решение этой системы.

Аналогично доказательству предыдущего утверждения.

3.5 Свойства итеративного алгоритма

3.5.4. Сходимость итеративного алгоритма

◊ **Определение 1.** *Восходящей цепочкой* в частично упорядоченном множестве $(L, \leq)$ называется последовательность его элементов, в которой $x_1 < x_2 < \ldots < x_n$.

◊ **Определение 2.** *Высотой* полурешетки называется наибольшее количество отношений $<$ в восходящих цепочках.

◊ **Утверждение.** Если полурешетка структуры монотонна и имеет конечную высоту, то итеративный алгоритм гарантированно сходится после количества итераций, не превышающего произведения высоты полурешетки на количество базовых блоков.

3.5 Свойства итеративного алгоритма

3.5.5 Решение, получаемое итеративным алгоритмом (максимальная фиксированная точка)

◊ Итеративный алгоритм

(1) посещает базовые блоки не в порядке их выполнения, а в порядке обхода ГПУ (на каждой итерации каждый узел посещается только один раз)

(2) в каждой точке сбора применяет операцию сбора к значениям потока данных, полученным к этому моменту

(3) иногда в пределах итерации базовый блок B посещается до посещения его предшественников (прямой обход)

3.5.5 Решение, получаемое итеративным алгоритмом (максимальная фиксированная точка)

◊ (4) для процесса итерации *необходимо граничное условие*, так как к блоку *Entry* передаточная функция неприменима.

(5) в качестве «нулевой итерации» все $Out[B]$ инициализируются значением $\top$, которое, по определению, «не меньше» всех значений потока, и, следовательно, того значения, которое оно заменяет; при этом монотонность передаточных функций обеспечивает получение результата, «не меньшего», чем искомое решение: