Моделирование программ: системы переходов, абстракция

Занятие №6

Простота — вот единственная почва, на которой мы можем строить здание наших обобщений. Но если эта простота — только кажущаяся, то будет ли достаточно надежной и сама почва? Вопрос этот заслуживает исследования.

А. Пуанкаре.
Наука и гипотеза

Александр Сергеевич Камкин

kamkin@ispras.ru

Вспомним, что такое структура Крипке

- AP – множество элементарных высказываний (атомов)
- $\langle S, S_0, R, L \rangle$ – *структура Крипке* над множеством AP
  - S — множество состояний
  - $S_0 \subseteq S$ — множество начальных состояний
  - $R \subseteq S \times S$ — отношение переходов
  - $L: S \to 2^{AP}$ — функция разметки

Структура Крипке как модель программы

- AP – утверждения о состоянии программы
  - Ограничения на значения переменных (данные)
  - Ограничения на точки исполнения (управление)
- $\langle S, S_0, R, L \rangle$ – система переходов, моделирующая программу
  - S — множество *(обобщенных) состояний* программы
  - $S_0 \subseteq S$ — множество начальных состояний
  - $R \subseteq S \times S$ — множество *(обобщенных) переходов* программы
  - $L: S \to 2^{AP}$ — функция разметки

Обобщенные состояния (абстракция данных)

$\langle \hat{S}, \hat{S}_0, \hat{R}, \hat{L} \rangle$ – *абстрактная система переходов,* порожденная *функцией абстракции* $f: S \to \hat{S}$

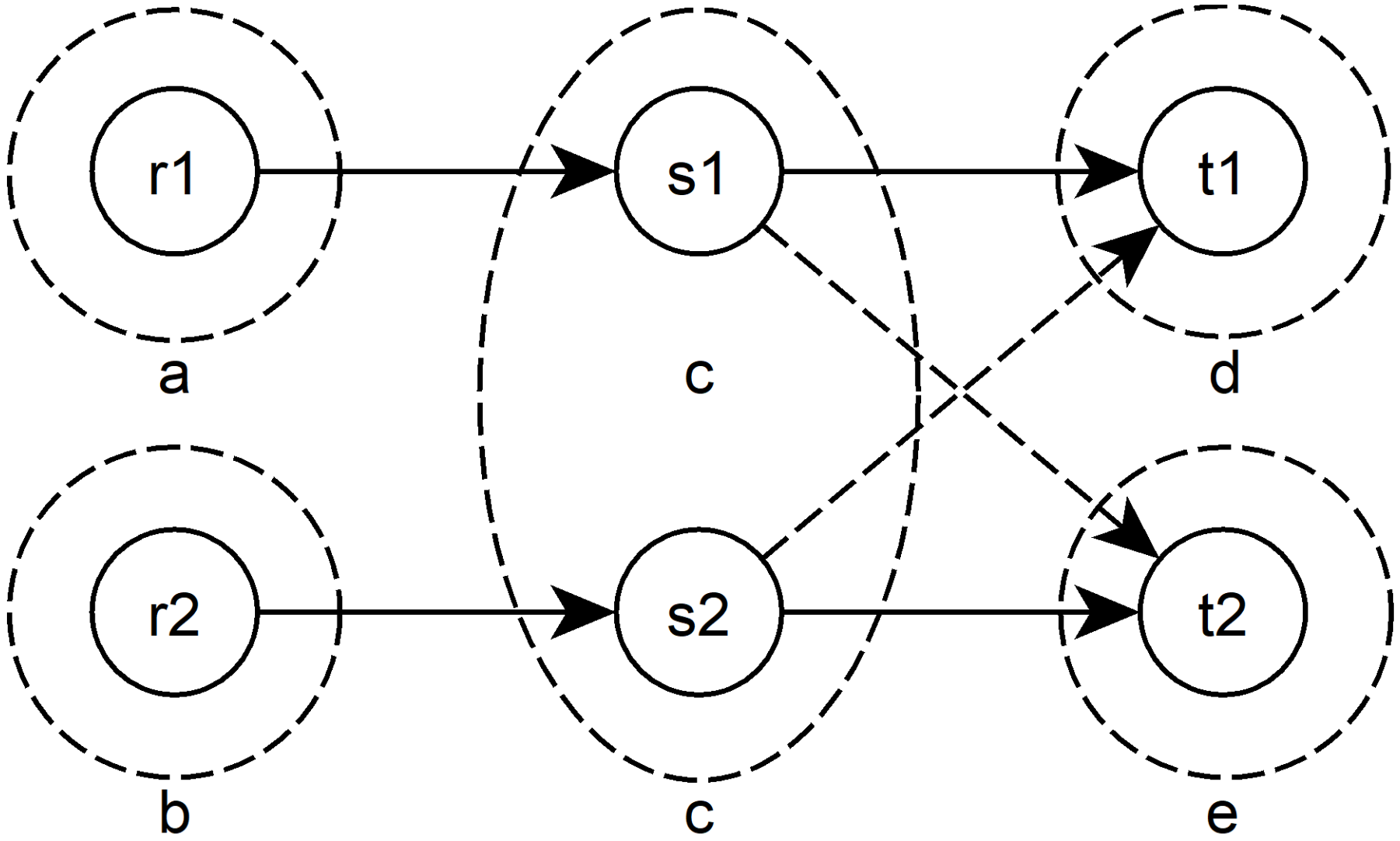
- $\hat{S}$ – множество обобщенных состояний
- $\hat{S}_0 = \{f(s) \mid s \in S_0\}$
- $\hat{R} = \{(f(s), f(s')) \mid (s, s') \in R\}$
- $\hat{L}\big(f(s)\big) = L(s)$ для всех $s \in S$
  - Предполагается, что $\big(f(s) = f(s')\big) \to \big(L(s) = L(s')\big)$

Пример абстракции данных

Структура конкретных сообщений	Структура абстрактных сообщений

typedef ConcreteMessage {
  /* *порядковый номер* */
  unsigned seq:4;
  /* *контрольная сумма* */
  unsigned checksum:16;
  /* *передаваемые данные* */
  byte data[32];
}
	typedef AbstractMessage {
	  /* *порядковый номер* */
	  unsigned seq:4;
	  /* *признак ошибки* */
	  bool error;
	  /* *передаваемые данные* */
	  /* *не моделируются* */
	}

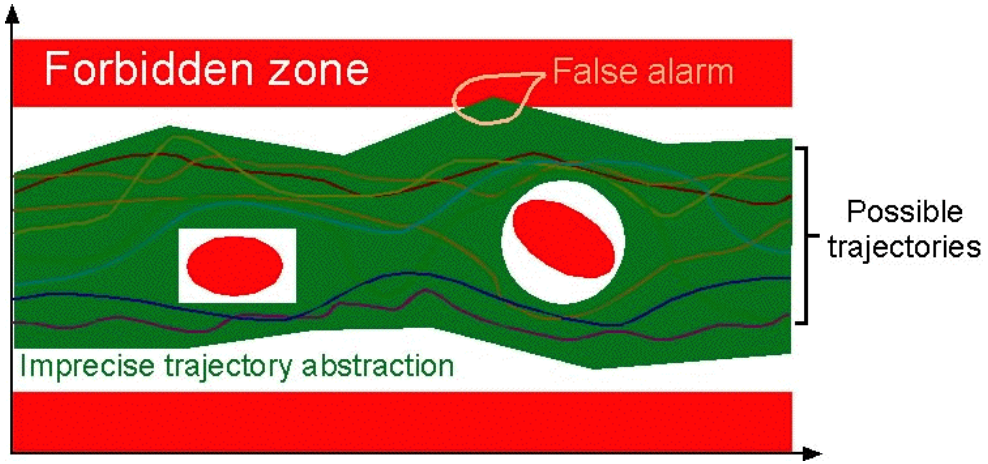
Проблема появления лишних траекторий



Пример появления лишних траекторий

Исходная модель	Результат абстракции
if :: φ(x) -> y = 1 :: **else** -> y = 0 **fi;** **if** :: φ(x) -> **assert**(y == 1) :: **else** -> **assert**(y == 0) **fi**	**if** *// игнорируем x* :: y = 1 :: y = 0 **fi;** **if** :: **assert**(y == 1) :: **assert**(y == 0) **fi**

Ошибки первого и второго рода



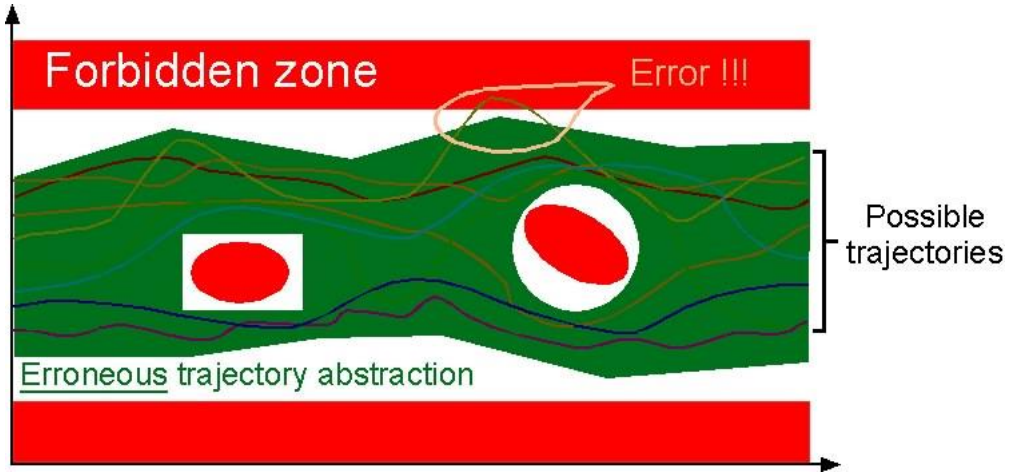
Patrick Cousot, *Abstract Interpretation in a Nutshell*, 2008

Ложная тревога (*false positive*)

- Сообщение выдается
- Ошибки в модели нет
- Неточная абстракция

Пропуск ошибки (*false negative*)

- Сообщение не выдается
- Ошибка в модели есть
- Некорректная абстракция



Адекватность модели программы

- Не допускаются ошибки второго рода (*false negatives*)
- Возможны ошибки первого рода (*false positives*)
- Для анализа модели требуются приемлемые ресурсы
  - Включая разбор ошибок первого рода
- *При абстракции на основе факторизации состояний, ошибки второго рода исключены*

Обобщенные переходы (гранулярность)

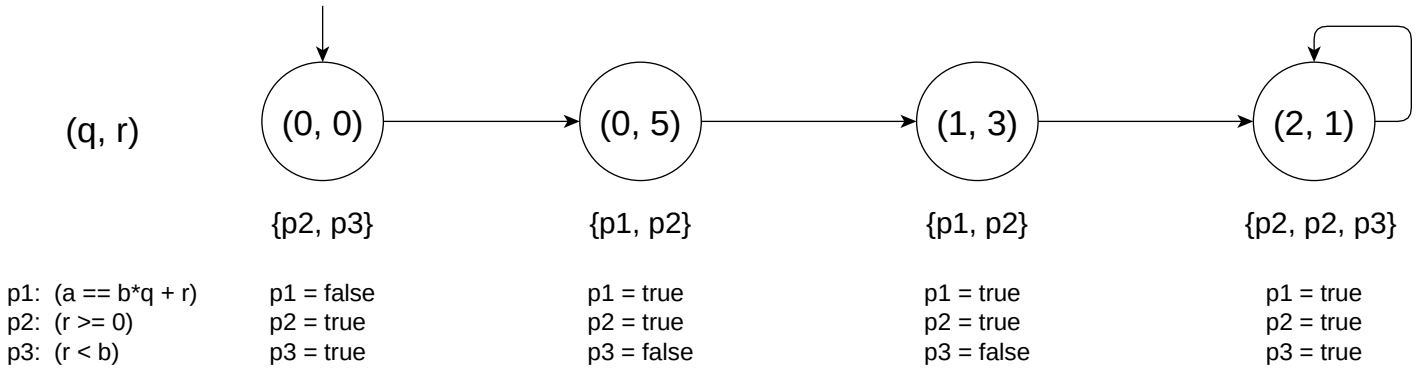
$$x := x + 1 \;||\; \ldots \;||\; x := x + 1$$

$P_1: t_1 := x$	$t_1 = 0$		$P_1: t_1 := x$	$t_1 = 0$
другие процессы			$P_1: x := t_1 + 1$	$x = 0 + 1 = 1$
могут выполняться	…	…	…	…
в произвольном			$P_n: t_n := x$	$t_n = n - 1$
порядке				
$P_1: x := t_1 + 1$	$x = 0 + 1 = 1$		$P_n: x := t_n + 1$	$x = (n-1) + 1 = n$

Пример: целочисленное деление

```
proctype DIV(int a, b) {
  int q, r;
  assert(a >= 0 && b > 0);
  atomic {
    q = 0;
    r = a
  }
  do
  :: atomic {
       r >= b -> q = q + 1;
                 r = r - b
     }
  :: else  -> break
  od;
  assert(a == b*q + r && 0 <= r && r < b)
}
```

$DIV(5, 2)$

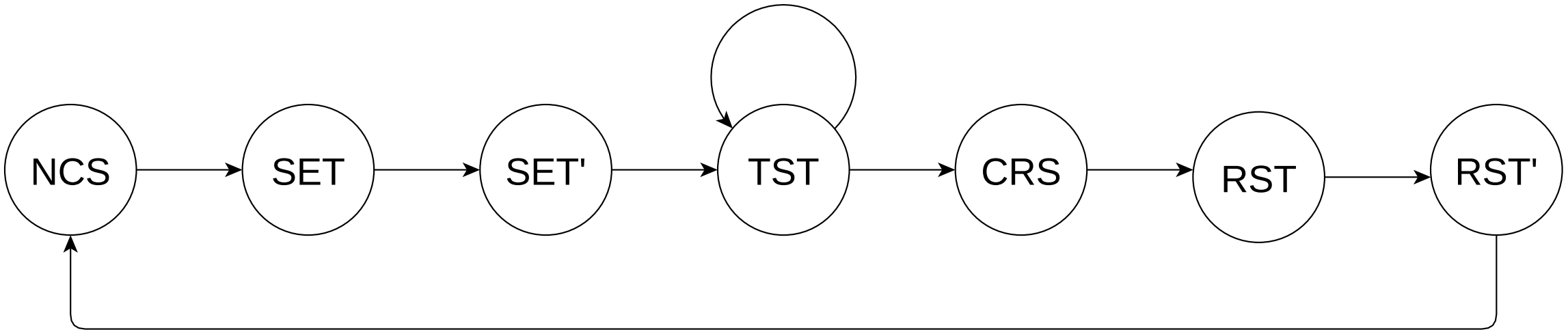


- $\pi \vDash \mathbf{F}\{p_1 \wedge p_2 \wedge p_3\}$
- $\pi \vDash \mathbf{FG}\{p_1 \wedge p_2 \wedge p_3\}$
- $\pi \vDash \mathbf{XG}\{p_1 \wedge p_2\}$

Пример: алгоритм Петерсона (1)

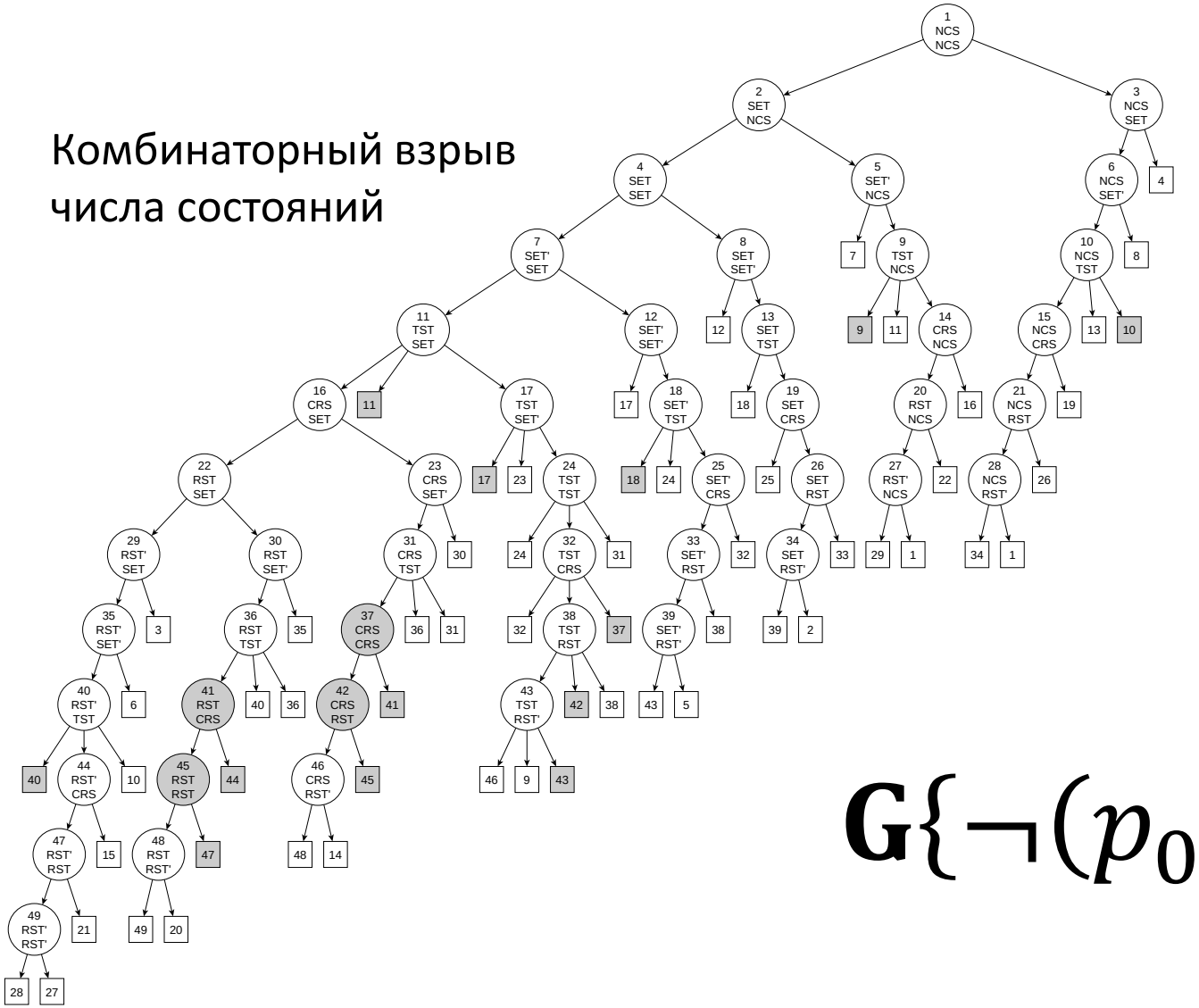
```
active [2] proctype P() {
  bit i = _pid;
  NCS: /* некритическая секция */
  SET: flag[i] = 1; turn = i;
  TST: !(flag[1 - i] == 1 && turn == i);
  CRS: /* критическая секция */
  RST: flag[i] = 0; goto NCS
}
```

- $p_0 \equiv P[0]@CRS$
- $p_1 \equiv P[1]@CRS$



Пример: алгоритм Петерсона (2)

Комбинаторный взрыв числа состояний



Асинхронная композиция систем переходов процессов

Исследование пространства состояний композиции

$$\mathbf{G}\{\neg(p_0 \wedge p_1)\}$$

Исследование пространства состояний

Поиск в ширину

```
S := {s0};
queue.push(s0);
while queue ≠ ∅ do
  s := queue.pop();
  // действие над s
  for s' in (succ(s) \ S) do
    queue.push(s')
  end;
  S := S ∪ succ(s)
end
```

Поиск в глубину

```
S := {s0};
stack.push(s0);
while stack ≠ ∅ do
  s := stack.top();
  if (succ(s) \ S) = ∅ then
    // действие над s
    stack.pop()
  else
    s' := choose(succ(s) \ S);
    stack.push(s');
    S := S ∪ {s'}
  end
end
```

Предикатная абстракция и CEGAR

- Задача – проверить достижимость заданной точки последовательной программы ($error$)
  - Программа сложная и содержит циклы (путей ∞ много)
  - Строится абстрактная модель программы специального вида
    - Если абстрактная модель корректна ($error$ недостижима)
      - Корректна и исходная программа
    - В противном случае строится контрпример (путь в $error$)
      - Если путь реализуем в исходной программе → *ошибка*
      - Иначе абстрактная модель уточняется (CEGAR)

Предикаты и регионы на множестве состояний

- Пусть p – предикат на множестве S состояний данных
  - $[\![p]\!] = \{s \in S \mid s \vDash p\}$
  - $[\![false]\!] = \emptyset$
  - $[\![true]\!] = S$
- Пусть $\{p_1, \ldots, p_n\}$ – конечное множество предикатов
  - *Регион* (обобщенное состояние данных)
    - $false$
    - $\bigwedge_{k=1}^{m} p_{i_k}$ $\quad (true$, если $m = 0)$

Разминка: построение регионов

Постройте все возможные регионы для предикатов $(x \geq 0)$ и $(x \leq 0)$

- $r_1 = \cdots ?$
- $r_2 = \cdots ?$
- $r_3 = \cdots ?$
- $r_4 = \cdots ?$
- $r_5 = \cdots ?$

Абстрактная система переходов

- Для заданного множества предикатов $\{p_1, \ldots, p_n\}$
- Вместо конфигураций $\langle l, s \rangle$, где s – состояние данных, рассматриваются $\langle l, r \rangle$, где r – регион
- *Вопрос*: в какой регион перейдет регион r при исполнении оператора (фрагмента кода) op?

$$r \xrightarrow{op} ?$$

Переходы между регионами (1)

Наша цель – найти *наиболее точный* регион r', включающий множество возможных состояний

$$\forall s \in S, \forall s' \in M[\![op]\!](s): s \in [\![r]\!] \Rightarrow s' \in [\![r']\!]$$

или

$$sp(op, r) \rightarrow r'$$

Переходы между регионами (2)

$$\left(p \to \bigwedge_{i=1}^{m} p_{i_k}\right) \equiv \left(\neg p \vee \bigwedge_{i=1}^{m} p_{i_k}\right) \equiv \left(\bigwedge_{i=1}^{m} (\neg p \vee p_{i_k})\right)$$

$$\neg\left(p \to \bigwedge_{i=1}^{m} p_{i_k}\right) \equiv \left(\bigvee_{i=1}^{m} (p \wedge \neg p_{i_k})\right)$$

Переходы между регионами (3)

- Если формула $sp(op, r)$ невыполнима, то $r' = false$
- В противном случае r' имеет вид конъюнкции (возможно, пустой) и строится так
  - Для всех $i \in \{1, \ldots, n\}$
    - Если формула $sp(op, r) \wedge \neg p_i$ невыполнима
      - Предикат p_i добавляется в конъюнкцию

Адаптивное построение модели (CEGAR)

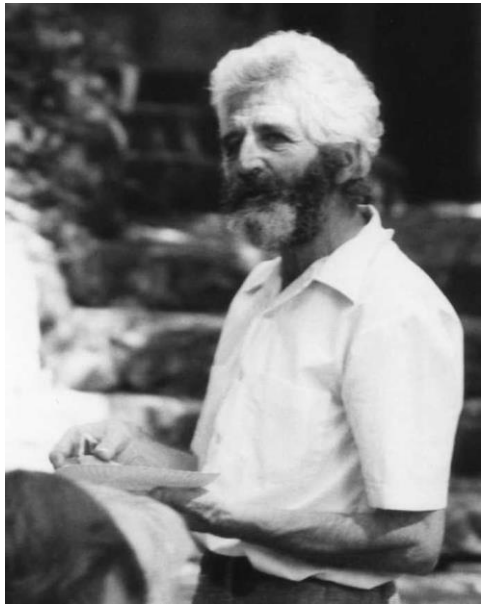
- $P \coloneqq \emptyset$ (имеем два региона: $true$ и $false$)
- **`while`** `true`
  - Строится абстрактная модель для P
  - Проверяется недостижимость точки $error$:
    - Если модель корректна, корректна и исходная программа
  - Строится π — вычисление из начальной конфигурации в конфигурацию вида $\langle error, r \rangle$, где r — регион, отличный от $false$
  - Проверяется выполнимость $sp(op(\pi), true)$:
    - Если SAT, исходная программа некорректна (π — контрпример)
  - В P добавляются предикаты, построенные по $op(\pi)$

Расширение множества предикатов (1)

- **Интерполяционная теорема Крейга (1957)**

  Если общезначима импликация $\varphi \to \psi$,
  то существует формула ρ, все свободные переменные которой (как и неинтерпретируемые функциональные и предикатные символы) являются общими для φ и ψ, такая что общезначимы $\varphi \to \rho$ и $\rho \to \psi$.

- Формула ρ называется *интерполянтом Крейга*



William Craig
(1918-2016)

Расширение множества предикатов (2)

- Пусть π — ложный контрпример и $op(\pi) = \{op_i\}_{i=1}^{n}$
- Формула $sp(op(\pi), true) = (\psi_1 \wedge \cdots \wedge \psi_n)$ невыполнима (SSA-представление)
- Формула $\psi_1 \rightarrow \neg(\psi_2 \wedge \cdots \wedge \psi_n)$ общезначима
- По теореме Крейга, существует интерполянт ρ_1 (по одной версии на переменную)
- Берем предикат p_1', такой что $\rho_1 = p_1'\theta_1$ (оригиналы := версии)
- Если $p_1' \in P$, то p_1' входит в регион r_1' (после исполнения op_1)
- Формулы $sp\left(op\left(\pi^{(2)}\right), r_1'\right)$, … тоже невыполнимы
- Применяем тот же подход: получаем предикаты p_2', …
- Процесс завершается, когда $r_k' = false$
- Для расширенного множества P вычисление π не есть контрпример