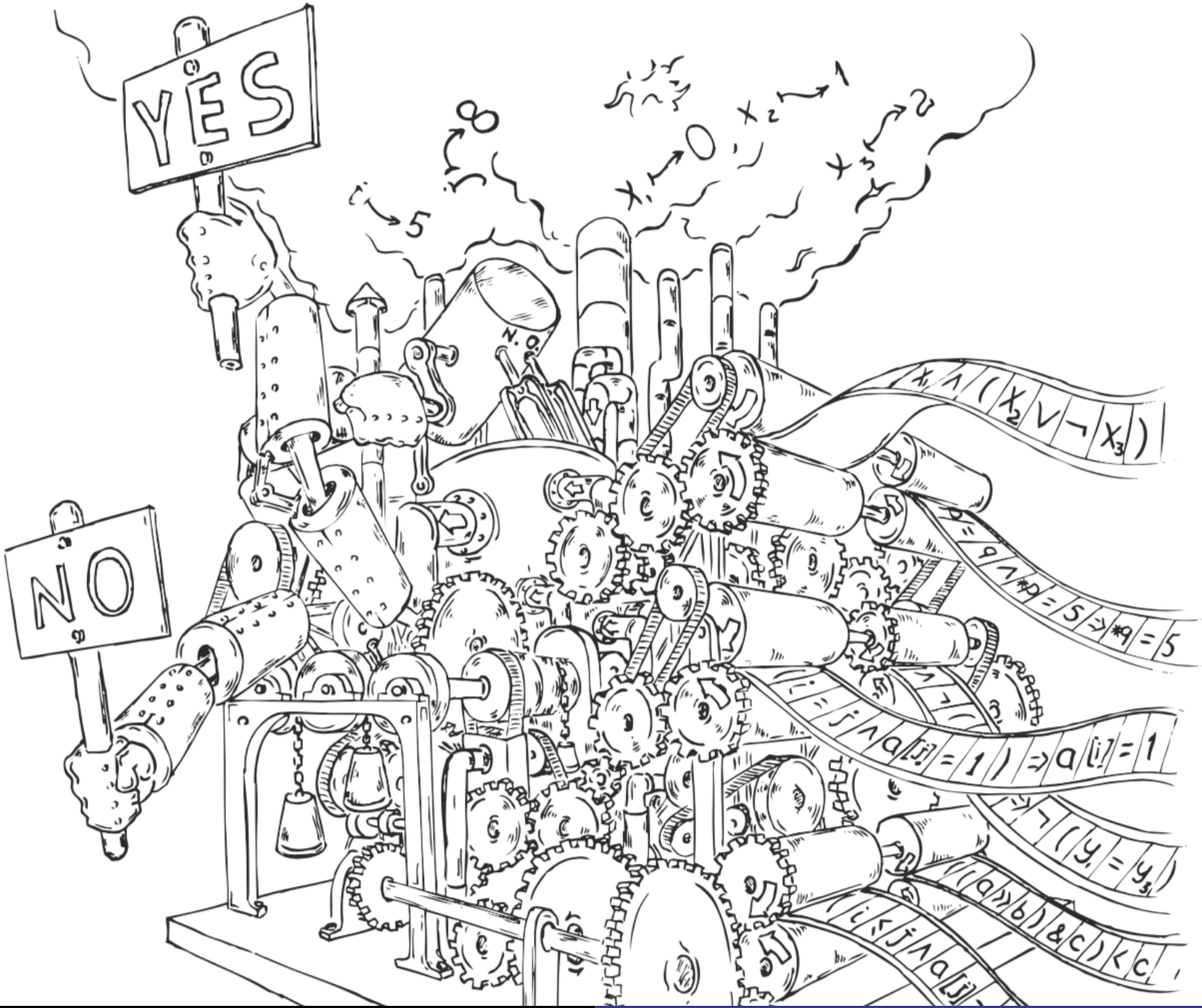

YES
NO

Задачи разрешимости логических формул и приложения
Лекция 4. Введение в SMT

Роман Холин

Московский государственный университет

Москва, 2021

Определения

- $at(\phi)$ - множество атомов в формуле ϕ
- $at_i(\phi)$ - некоторый заданный порядок этих атомов
- Атом a сопоставим с булевой переменной $e(a)$ (такую процедуру будем называть булевское кодирование)
- $e(t)$ - булева формула, полученную булевым кодированием каждого атома формулы t (такую формулу будем называть пропозиционным скилетом формулы t)

Пример

- $\phi := x = y \vee x = z$
- $e(x = y) = b_1$
- $e(x = z) = b_2$
- $e(\phi) = b_1 \vee b_2$

Определения

- α - некоторая (возможно, частичная) оценка формулы $e(\phi)$
- $\alpha = \{e(at_1) \to FALSE, e(at_2) \to TRUE\}$
- $Th(at_i, \alpha) = at_i$, если $\alpha(at_i) = TRUE$, $\neg at_i$ иначе
- $Th(\alpha) = \{Th(at_i, \alpha) | e(at_i), \alpha\}$
- $\overline{Th(\alpha)}$ - конъюнкция всех элементов их $Th(\alpha)$

Ленивый алгоритм решение SMT задач

Пусть нам данн алгоритм (назовём его *Deduction*), который может решить $\overline{Th(\alpha)}$

- Вычислим $B = e(\phi)$

Ленивый алгоритм решение SMT задач

Пусть нам данн алгоритм (назовём его *Deduction*), который может решить $\overline{Th(\alpha)}$

- Вычислим $B = e(\phi)$
- Отправим B SAT-решателю

Ленивый алгоритм решение SMT задач

Пусть нам данн алгоритм (назовём его *Deduction*), который может решить $\overline{Th(\alpha)}$

- Вычислим $B = e(\phi)$
- Отправим B SAT-решателю
- Если получили "UNSAT то возвращаем "UNSAT"

Ленивый алгоритм решение SMT задач

Пусть нам данн алгоритм (назовём его *Deduction*), который может решить $\overline{Th(\alpha)}$

- Вычислим $B = e(\phi)$
- Отправим B SAT-решателю
- Если получили "UNSAT то возвращаем "UNSAT"
- Если получили "UNKNOWN то возвращаем "UNKNOWN"

Ленивый алгоритм решение SMT задач

Пусть нам данн алгоритм (назовём его *Deduction*), который может решить $\overline{Th(\alpha)}$

- Вычислим $B = e(\phi)$
- Отправим B SAT-решателю
- Если получили "UNSAT то возвращаем "UNSAT"
- Если получили "UNKNOWN то возвращаем "UNKNOWN"
- Если получили некоторую α, то вычисляем *Deduction* от $\overline{Th(\alpha)}$

Ленивый алгоритм решение SMT задач

Пусть нам данн алгоритм (назовём его *Deduction*), который может решить $\overline{Th(\alpha)}$

- Вычислим $B = e(\phi)$
- Отправим B SAT-решателю
- Если получили "UNSAT то возвращаем "UNSAT"
- Если получили "UNKNOWN то возвращаем "UNKNOWN"
- Если получили некоторую α, то вычисляем *Deduction* от $\overline{Th(\alpha)}$
- Если получили "SAT то возвращаем "SAT"

Ленивый алгоритм решение SMT задач

Пусть нам данн алгоритм (назовём его *Deduction*), который может решить $\overline{Th(\alpha)}$

- Вычислим $B = e(\phi)$
- Отправим B SAT-решателю
- Если получили "UNSAT то возвращаем "UNSAT"
- Если получили "UNKNOWN то возвращаем "UNKNOWN"
- Если получили некоторую α, то вычисляем *Deduction* от $\overline{Th(\alpha)}$
- Если получили "SAT то возвращаем "SAT"
- Если получили "UNKNOWN то возвращаем "UNKNOWN"

Ленивый алгоритм решение SMT задач

Пусть нам данн алгоритм (назовём его *Deduction*), который может решить $\overline{Th(\alpha)}$

- Вычислим $B = e(\phi)$
- Отправим B SAT-решателю
- Если получили "UNSAT то возвращаем "UNSAT"
- Если получили "UNKNOWN то возвращаем "UNKNOWN"
- Если получили некоторую α, то вычисляем *Deduction* от $\overline{Th(\alpha)}$
- Если получили "SAT то возвращаем "SAT"
- Если получили "UNKNOWN то возвращаем "UNKNOWN"
- Если получили "UNSAT то $B = B \wedge$ "блокирующий дизюнкт"t и начинаем со второго пункта

Блокирующий дизюнкт

Так же называют леммой
Свойства:

- t - тавтология в T
- t - состоит только из атомов из ϕ
- t - "блокирует"α, т.е. при отправлении B SAT-решателю мы не сможем снова получить α

Блокирующий дизюнкт

Так же называют леммой

Свойства:

- t - тавтология в T
- t - состоит только из атомов из ϕ
- t - "блокирует"α, т.е. при отправлении B SAT-решателю мы не сможем снова получить α

Пример:

- Пусть $\alpha = \{e(at_1) \to FALSE, e(at_2) \to TRUE\}$
- $t := e(at_1) \vee \neg e(at_2)$

Есть и другие способы найти блокирующий дизюнкт

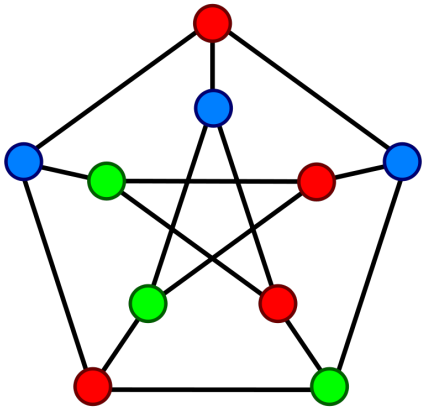
Ленивый алгоритм решение SMT задач

- Есть более эффективные алгоритмы решения SMT задач, но для этого нам нужно изучить, как работает SAT решатель
- В процессе работы ленивого алгоритма может возникнуть много блокирующих дизюнктов
- Многие теории можно свести к SAT задаче, но обычно существуют более эффективные алгоритмы решения теорий

Пример

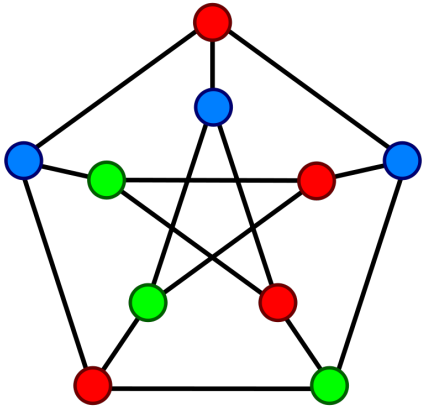
$$x = y \wedge ((y = z \wedge \neg(x = z)) \vee x = z)$$

Раскраска графа



Имеется граф $G = (V, E)$. Можно ли его вершини расскрасить в k цветов так, чтобы никакие соседние вершины не были раскрашены в один и тот же цвет?

Раскраска графа



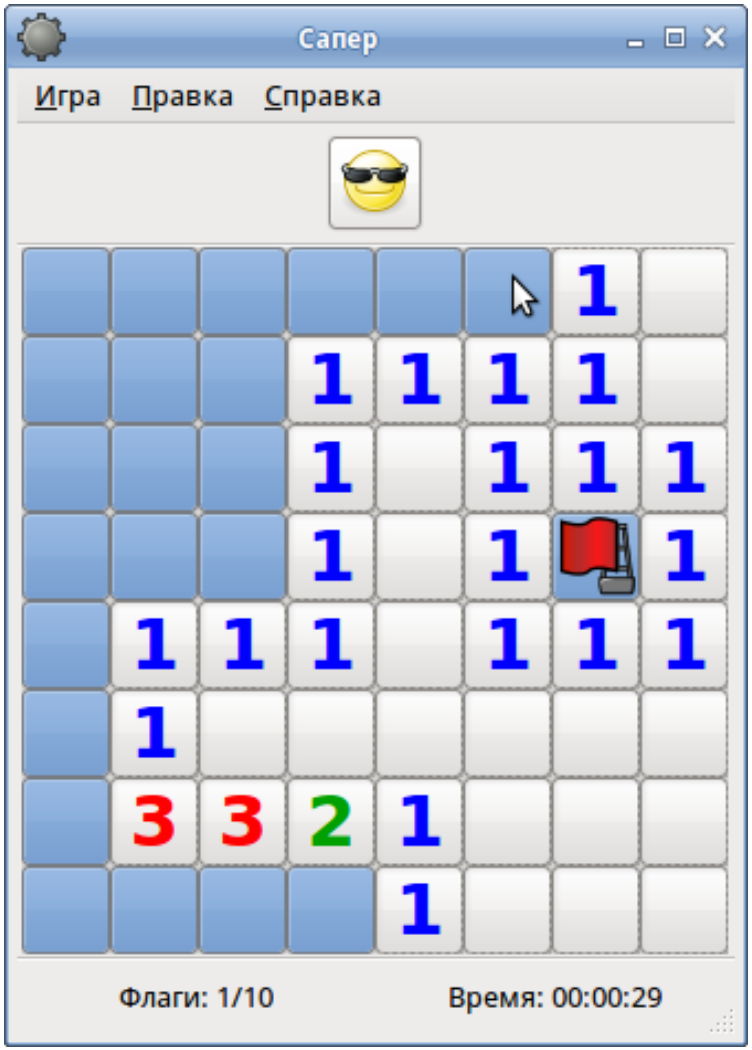
Имеется граф $G = (V, E)$. Можно ли его вершини расскрасить в k цветов так, чтобы никакие соседние вершины не были раскрашены в один и тот же цвет?

- x_i - целые
- $1 \leq x_i \leq c$
- $x_i \neq x_j$ для $(x_i, x_j) \in E$

Судоку

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Сапер



Эквивалентность программ

Программа a:

```
int i, out_a = in;
for (int i = 0; i < 2; ++i) out_a = out_a * inn;
return out_a;
```

Программа b:

```
int out_b = (in * in) * in;
return out_b;
```

Эквивалентность программ

Программа a:

```
int i, out_a = in;
for (int i = 0; i < 2; ++i) out_a = out_a * in;
return out_a;
```

Программа b:

```
int out_b = (in * in) * in;
return out_b;
```

$\phi_a := (out\_a_0 = in\_a_0) \wedge (out\_a_1 = out\_a_0 * in\_a_0) \wedge (out\_a_2 = out\_a_1 * in\_a_0)$

$\phi_b := out\_b_0 = (in\_b_0 * in\_b_0) * in\_b_0$

Эквивалентность программ

Программа a:

```
int i, out_a = in;
for (int i = 0; i < 2; ++i) out_a = out_a * in;
return out_a;
```

Программа b:

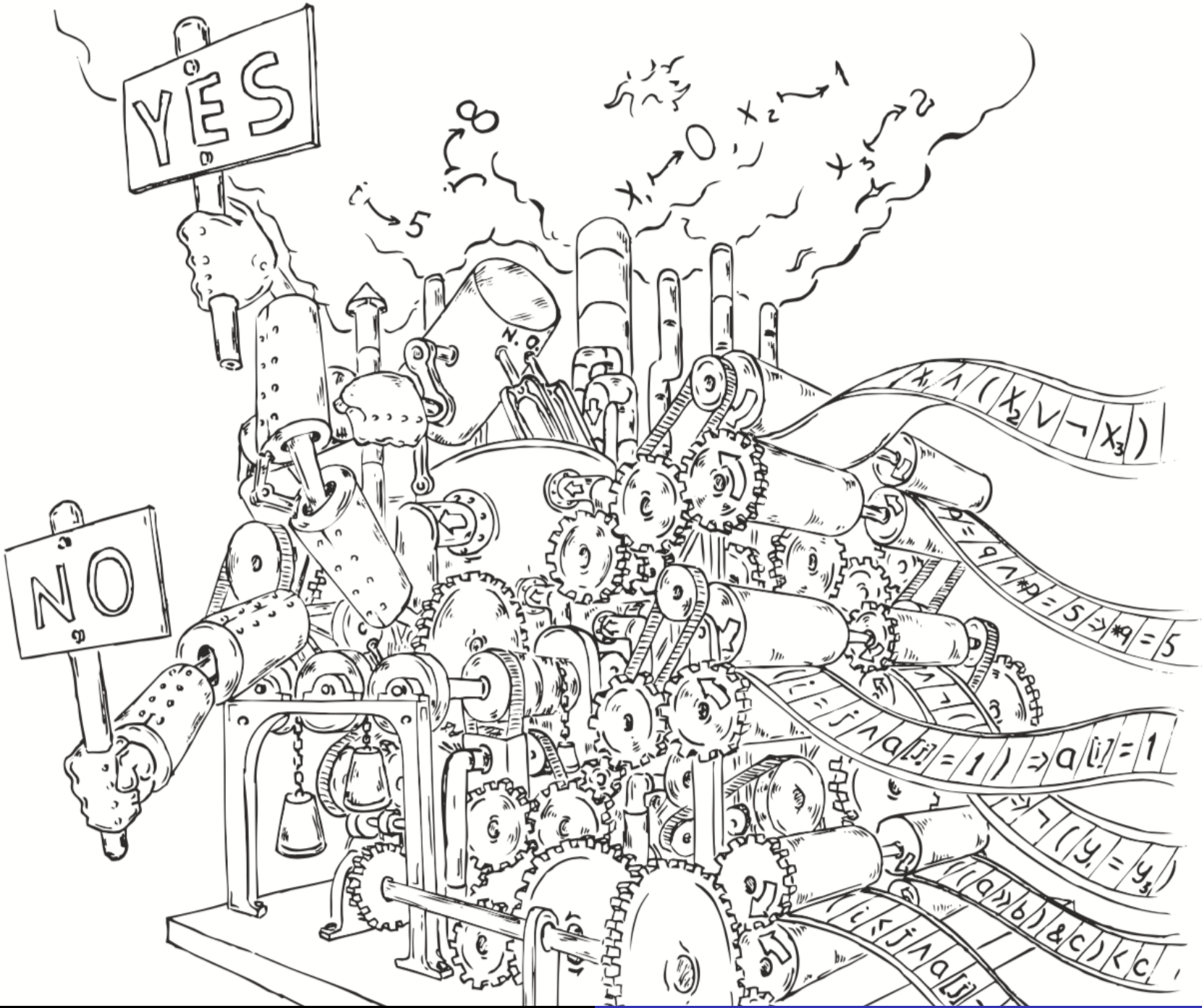
```
int out_b = (in * in) * in;
return out_b;
```

$\phi_a := (out\_a_0 = in\_a_0) \wedge (out\_a_1 = out\_a_0 * in\_a_0) \wedge (out\_a_2 = out\_a_1 * in\_a_0)$

$\phi_b := out\_b_0 = (in\_b_0 * in\_b_0) * in\_b_0$

Чтобы программы были эквивалентны, должно выполняться:

$(in\_a_0 = in\_b_0) \wedge \phi_a \wedge \phi_b \rightarrow out\_a_2 = out\_b_0$


YES
NO