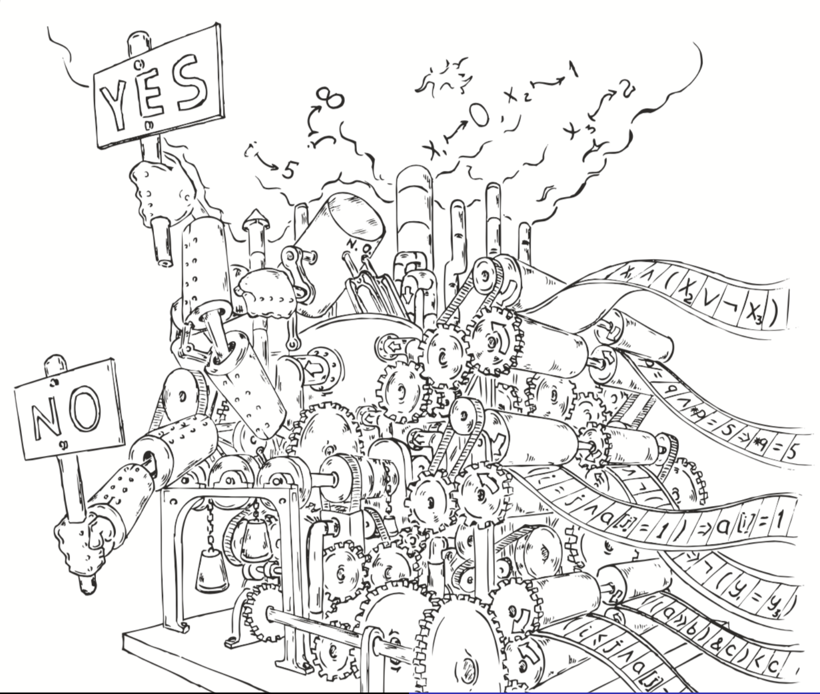

YES
NO
N.O.
$x_1 \to 0$
$x_2 \to 1$
$i \to 5$

Задачи разрешимости логических формул и приложения
Лекция 5. Логики равенства и неинтерпритируемых функций

Роман Холин

Московский государственный университет

Москва, 2022

Логика равенства (EQ)

$$formula : formula \vee formula | \neg formula | (formula) | atom$$
$$atom : term = term$$
$$term : identifier | constant$$

Логика равенства (EQ)

$formula : formula \vee formula | \neg formula | (formula) | atom$
$atom : term = term$
$term : identifier | constant$
Какова сложность задачи разрешимости формулы логики равенства?

Логика равенства (EQ)

$formula : formula \vee formula | \neg formula | (formula) | atom$
$atom : term = term$
$term : identifier | constant$
Какова сложность задачи разрешимости формулы логики равенства?
Зачем изучать и пропозиционную логику и логику равенства?

Удаление констант

$\phi = \phi'$
В ϕ' заменить все c_i на C_{c_i}
Для всех $i \neq j$ добавить в ϕ' формулы $C_{c_i} \neq C_{c_j}$

Удаление констант

$\phi' = \phi$
В ϕ' заменить все c_i на C_{c_i}
Для всех $i \neq j$ добавить в ϕ' формулы $C_{c_i} \neq C_{c_j}$
Далее будем считать, что в логике нет констант

Логика неинтерпретируемых функций (UF)

$$F(x) = F(G(y)) \lor x + 1 = y$$

Логика неинтерпретируемых функций (UF)

$$F(x) = F(G(y)) \vee x + 1 = y$$
$$formula : formula \vee formula | \neg formula | (formula) | atom$$
$$atom : term = term | predicate\_symbol(list\_of\_terms)$$
$$term : identifier | function\_symbol(list\_of\_terms)$$

Логика неинтерпретируемых функций (UF)

$F(x) = F(G(y)) \vee x + 1 = y$
$formula : formula \vee formula | \neg formula | (formula) | atom$
$atom : term = term | predicate\_symbol(list\_of\_terms)$
$term : identifier | function\_symbol(list\_of\_terms)$
Если predicate_symbol состоит только из $\{=\}$, то логику называют EUF

Логика неинтерпретируемых функций(UF)

Как используется?

Как используется?
Если можем доказать в этой логике, то можем доказать и в общей фомруле

Логика неинтерпретируемых функций(UF)

Как используется?
Если можем доказать в этой логике, то можем доказать и в общей фомруле
Главное, чтобы выполнялось условие функциональной совместимости: значение функции на одних и тех же аргументах одинаково

Логика неинтерпретируемых функций(UF)

Как используется?
Если можем доказать в этой логике, то можем доказать и в общей фомруле
Главное, чтобы выполнялось условие функциональной совместимости: значение функции на одних и тех же аргументах одинаково
Обратное не верно: если формула не тавтология в EUF, то не факт, что это не тавтология в EUF

Эквивалентность программ

```
int power3(int in)
{
  int i, out_a;
  out_a = in;
  for (i = 0; i < 2; i++)
    out_a = out_a * in;
  return out_a;
}
```

(a)

```
int power3_new(int in)
{
  int out_b;

  out_b = (in * in) * in;

  return out_b;
}
```

(b)

Эквивалентность программ

```
int power3(int in)
{
  int i, out_a;
  out_a = in;
  for (i = 0; i < 2; i++)
    out_a = out_a * in;
  return out_a;
}
```

(a)

```
int power3_new(int in)
{
  int out_b;

  out_b = (in * in) * in;

  return out_b;
}
```

(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:

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

Эквивалентность программ

```
int power3(int in)
{
  int i, out_a;
  out_a = in;
  for (i = 0; i < 2; i++)
    out_a = out_a * in;
  return out_a;
}
```

(a)

```
int power3_new(int in)
{
  int out_b;

  out_b = (in * in) * in;

  return out_b;
}
```

(b)

Перепишем формулу в EUF. Будем считать, что * - это некоторый двуместный функциональный символ G(,)

Эквивалентность программ

```
int power3(int in)
{
  int i, out_a;
  out_a = in;
  for (i = 0; i < 2; i++)
    out_a = out_a * in;
  return out_a;
}
```

(a)

```
int power3_new(int in)
{
  int out_b;

  out_b = (in * in) * in;

  return out_b;
}
```

(b)

Перепишем формулу в EUF. Будем считать, что * - это некоторый двуместный функциональный символ G(,)

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

$\phi_b := out\_b_0 = G(G(in\_b_0, in\_b_0), in\_b_0)$

Чтобы программы были эквивалентны, должно выполняться: $(in\_a_0 = in\_b_0) \wedge \phi_a \wedge \phi_b \to out\_a_2 = out\_b_0$, но теперь нам не нужно знать природу G

Эквивалентность программ

```
int mul3(struct list *in)
{
 int i, out_a;
 struct list *a;
 a = in;
 out_a = in -> data;
 for (i = 0; i < 2; i++) {
   a = a -> n;
   out_a= out_a * a -> data;
 }
 return out_a;
}
```

(a)

```
int mul3_new(struct list *in)
{
  int out_b;

  out_b =
    in -> data *
    in -> n -> data *
    in -> n -> n -> data;

  return out_b;
}
```

(b)

Эквивалентность программ

```
int mul3(struct list *in)
{
 int i, out_a;
 struct list *a;
 a = in;
 out_a = in -> data;
 for (i = 0; i < 2; i++) {
   a = a -> n;
   out_a= out_a * a -> data;
 }
 return out_a;
}
```

(a)

```
int mul3_new(struct list *in)
{
  int out_b;

  out_b =
    in -> data *
    in -> n -> data *
    in -> n -> n -> data;

  return out_b;
}
```

(b)

$$\begin{aligned}
&a0_a = in0_a && \wedge\\
&out0_a = list_data(in0_a) && \wedge\\
&a1_a = list_n(a0_a) && \wedge\\
&out1_a = G(out0_a, list_data(a1_a)) && \wedge\\
&a2_a = list_n(a1_a) && \wedge\\
&out2_a = G(out1_a, list_data(a2_a))
\end{aligned}$$

$$\begin{aligned}
&out0_b = G(G(list_data(in0_b),\\
&list_data(list_n(in0_b)),\\
&list_data(list_n(list_n(in0_b)))))
\end{aligned}$$

Замыкание классов эквивалентности

- Построить замыкание классов эквивалентности:
  Поместить в один класс эквивалентности термы, если $t_1 = t_2$
  Слить все в один класс эквивалентности формулы $F(t_i)$, такие что $F(t_i) = F(t_j)$ и t_i и t_j в одном классе эквивалентности
  Повторять, пока есть что сливать
- Если какие-то ϕ_i и ϕ_j в одном классе эквивалентности и есть неравенство $\phi_i \neq \phi_j$, то вернуть результат «Unsatisfiable». Иначе вернуть «Satisfiable»

Пример

$$(x_1 = x_2) \wedge (x_2 = x_3) \wedge (x_4 = x_5) \wedge (x_5 \neq x_1) \wedge (F(x_1) \neq F(x_3))$$

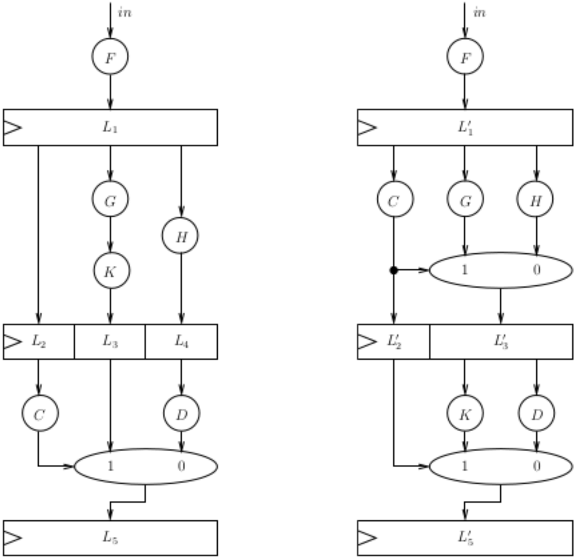
Недостаточность функциональной совместимости

$$(x_1 = y_2) \wedge (x_2 = y_1) \to (x_1 + y_1) = (x_2 + y_2)$$

Алгоритм уточнения

1. $\phi' = t(\phi)$ (t - абстрагирование исходной формулы)
2. если ϕ' - тавтология, то исходная так же таводогия
3. если $\phi' = \phi$, то не тавтология
4. добавить "уточнение"и вернуться на шаг 2

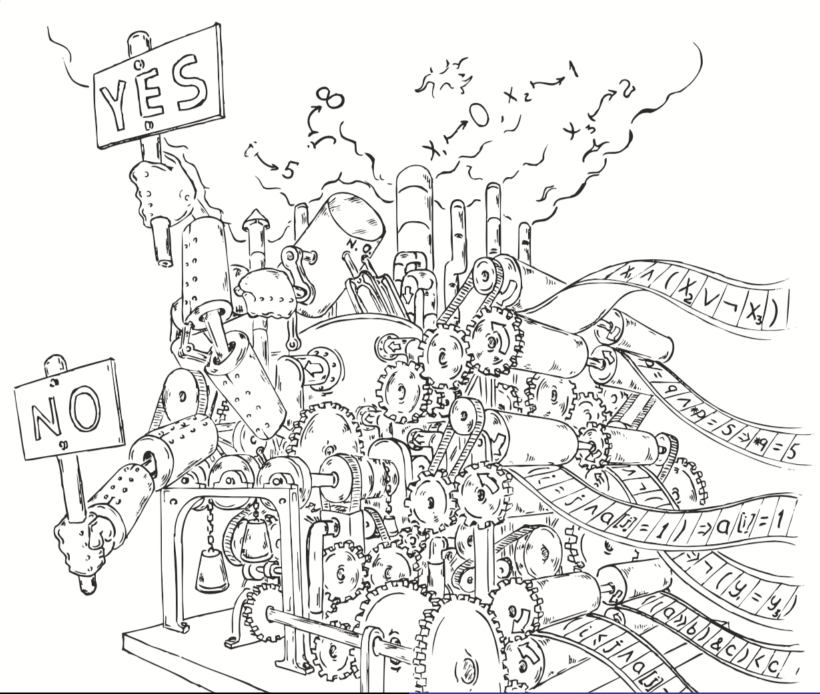
Пример



Пример

$$z = (x_1 + y_1) * (x_2 + y_2)$$

$$u_1 = x_1 + y_1; u_2 = x_2 + y_2; z = u_1 * u_2$$


YES
NO