

Суперкомпиляция как основа для построения алгоритмов решения уравнений в словах

А. Н. Непейвода

Институт Программных систем им. А.К. Айламазяна РАН

*IV совместное рабочее совещание
по функциональному языку Рефал
8 июня 2021*

Постановка задачи

Дана система уравнений в словах $\mathcal{E}qs$. Существует ли последовательность подстановок σ , порождающая решение $\mathcal{E}qs$?

Уравнения в словах

Определение

Даны алфавит констант Σ и переменных \mathcal{V} . *Уравнение в словах* — выражение $\Phi = \Psi$, где $\Phi, \Psi \in \{\Sigma \cup \mathcal{V}\}^*$.

Решение уравнения в словах — подстановка $\sigma : \mathcal{V} \rightarrow \Sigma^*$ такая, что $\Phi\sigma$ буквально совпадает с $\Psi\sigma$.

$E = xAB = BAx$, где $A, B \in \Sigma$, $x \in \mathcal{V}$. Рассмотрим $\sigma_1 : x \rightarrow Bx$, $\sigma_2 : x \rightarrow \varepsilon$. Тогда $\sigma_2 \circ \sigma_1 : x \rightarrow B$ — решение E : $(xAB)\sigma_1\sigma_2 = BAB = (BAx)\sigma_1\sigma_2$.

Краткая история

Теория:

- Решение квадратичных уравнений ($xAy = yAx$) и уравнений от одной переменной (Матиясевич, 1965)
- Решение уравнений от трех переменных (Хмелевский, 1971)
- Алгоритм решения уравнений в общем случае (Маканин, 1977)
- Более быстрые (но также гиперэкспоненциальные) алгоритмы (Plandowski, 2006, Jez, 2016)

Краткая история

Практика:

- эффективные алгоритмы решения линейных ($xxx = yAz$) уравнений (Rümmer et al., 2014–...)
- алгоритмы решения квадратичных уравнений (Le et al., Lin et al., 2018)
- алгоритмы решения уравнений с ограниченной длиной решений (Bjørner, 2009–..., Day, 2019)

Язык уравнений в словах

Определение

Множество закодированных уравнений:

$$\text{Eqs} ::= \text{Eq Eqs} \mid \varepsilon$$
$$\text{Eq} ::= (\text{Side}, \text{Side})$$
$$\text{Side} ::= \text{Char Side} \mid \text{Var Side} \mid \varepsilon$$

$\text{Var} \in \mathcal{V}$, $\text{Char} \in \Sigma$, ε — пустое слово.

Синтаксический сахар:

$$(\text{LHS}, \text{RHS}) \longrightarrow \text{LHS} = \text{RHS};$$
$$(\text{LHS}_1, \text{RHS}_1) \dots (\text{LHS}_n, \text{RHS}_n) \longrightarrow \langle \text{LHS}_i = \text{RHS}_i \rangle_{i=1}^n.$$

Простой логический язык \mathcal{L}

Определение

Определим последовательность сужений Narrs .

$$\text{Narrs} ::= (\text{Narr}) \text{ Narrs} \mid \varepsilon$$

$$\text{Narr} ::= ' \text{Var} \rightarrow \text{Char Var}' \mid ' \text{Var} \rightarrow \text{Var}_1 \text{Var}' \mid ' \text{Var} \rightarrow \varepsilon '$$

$$\text{Var}, \text{Var}_1 \in \mathcal{V}, \text{Char} \in \Sigma, \text{Var} \neq \text{Var}_1.$$

Любая последовательность из Narrs определяет подстановку $\sigma : \mathcal{V} \rightarrow (\mathcal{V} \cup \Sigma)^*$. Пусть $x \in \mathcal{V}$, тогда σ — это $x \rightarrow \Phi$ или $x \rightarrow \Phi x$, где Φ не содержит x .

Множество последовательностей из Narrs рассматриваем как простой логический язык \mathcal{L} над Eqs .

Простой логический язык \mathcal{L}

Определение

Определим последовательность сужений Narrs .

$$\text{Narrs} ::= (\text{Narr}) \text{ Narrs} \mid \varepsilon$$

$$\text{Narr} ::= ' \text{Var} \rightarrow \text{Char Var}' \mid ' \text{Var} \rightarrow \text{Var}_1 \text{Var}' \mid ' \text{Var} \rightarrow \varepsilon '$$

Совместимость сужений с $\langle \Phi_1 = \Psi_1, \dots, \Phi_n = \Psi_n \rangle$:

$'x \rightarrow \varepsilon'$	$x\Phi_1 = \Psi_1$ или $\Phi_1 = x\Psi_1$	$'x \rightarrow tx'$	$x\Phi_1 = t\Psi_1$ или $t\Phi_1 = x\Psi_1$
$'x \rightarrow x_1x'$		$x\Phi_1 = x_1\Psi_1$ или $x_1\Phi_1 = x\Psi_1$	

Множество последовательностей из Narrs рассматриваем как простой логический язык \mathcal{L} над Eqs .

Семантика \mathcal{L}

Интерпретатор \mathcal{L} $Wl_{\mathcal{L}}$ принимает последовательность $(\sigma_1)(\sigma_2)\dots(\sigma_n)$ и систему $\langle \Phi_i = \Psi_i \rangle_{i=1}^m$.

Вызов $Wl_{\mathcal{L}}((\sigma_1)(\sigma_2)\dots(\sigma_n), \langle \Phi_i = \Psi_i \rangle_{i=1}^m)$ возвращает T , если $\forall i, 1 \leq i \leq m (\Phi_i \sigma_1 \dots \sigma_n = \Psi_i \sigma_1 \dots \sigma_n)$, и F иначе.

Свойства интерпретатора $Wl_{\mathcal{L}}$:

- совершает не больше n шагов (всегда завершается);
- для всех $\langle \Phi_i = \Psi_i \rangle_{i=1}^m$ возвращает либо T , либо F .

Суперкомпиляция интерпретаторов языка \mathcal{L}

В вызове $Wl_{\mathcal{L}}(P, \langle \Phi_i = \Psi_i \rangle_{i=1}^n)$ заменяем последовательность P на параметр \mathcal{P} .

Получаем следующую задачу суперкомпиляции:

$$Wl_{\mathcal{L}}(\mathcal{P}, \langle \Phi_i = \Psi_i \rangle_{i=1}^n)$$

Ее развертка порождает дерево (возможно, бесконечное): описание путей вычисления всех возможных \mathcal{L} -программ на $\langle \Phi_i = \Psi_i \rangle_{i=1}^n$.

Задача верификации

Пусть дана система уравнений $\mathcal{E}qs$. Скажем, что ее верификация успешна, если $\mathcal{E}qs$ если остаточная программа, порожденная суперкомпиляцией $Wl_{\mathcal{L}}(\mathcal{P}, \mathcal{E}qs)$, содержит функцию, возвращающую \top , лишь в том случае, если $\mathcal{E}qs$ имеет решения.

Суперкомпиляция может уходить в бесконечную развертку для некоторых $\mathcal{E}qs$.

Синтаксис остаточных программ

Определение

$\text{Program} ::= \text{Rule}; \text{Program} \mid \varepsilon$

$\text{Rule} ::= \text{Name}(\text{Pattern}) = \text{Expression}$

$\text{Pattern} ::= (\text{Narr}) \mid (\text{Narr}) ++ \text{Pattern} \mid p \mid \varepsilon$

$\text{Expression} ::= T \mid F \mid \text{Name}(p)$

p пробегает Narrs, Name — имя функции.

Все функции — от одного аргумента. В левых и правых частях определений — максимум по одному вхождению переменной p .

Пример

Для уравнения $Ax = xA$ суперкомпилятор порождает следующую программу со входной точкой $F(p)$.

```
F(( 'x → ε' )) = T
F(( 'x → Ax' ) ++ p) = F(p)
F(p) = F
```

Для уравнения $Ax = xB$ остаточная программа будет такой: (входная точка — $G(p)$).

```
G(( 'x → ε' )) = F
G(( 'x → Ax' ) ++ p) = G(p)
G(p) = F
```

Общая структура интерпретаторов

Основной цикл интерпретации

Main

1. Выбрать подстановку

Subst

2. Применить ее

Smpl

3. Упростить результат

- Smpl получает на вход статические данные (систему уравнений) и не меняет их семантику.
- Smpl всегда завершается.

Базовый интерпретатор $WIBase_{\mathcal{L}}$

Структура функции $Smp1$

Удалить общий префикс \rightarrow Удалить общий суффикс

Эту операцию далее называем нормированием.

Входной формат

p — последовательности правил;

$\mathcal{E}qs$ — (системы) уравнений.

$$Go(p, \mathcal{E}qs) = Main(p, Smp1(\varepsilon, \mathcal{E}qs));$$

- Суперкомпиляция задачи $WIBase_{\mathcal{L}}(\mathcal{P}, \Phi = \Psi)$ успешно решает все квадратичные уравнения $\Phi = \Psi$.

Интерпретатор с расщеплением

$\text{WISplit}_{\mathcal{L}}$

Структура функции Smp1

Нормировать \rightarrow Расщепить слева \rightarrow Нормировать

Входной формат

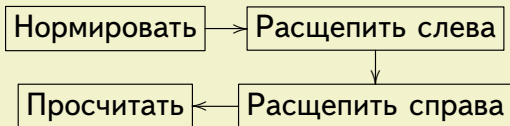
$$\text{Go}(p, \mathcal{E}qs) = \text{Main}(p, \text{Smp1}(0, \varepsilon, \mathcal{E}qs));$$

- Первый аргумент Smp1 добавлен для сохранения оптимальности (см. ниже).
- Суперкомпиляция задачи $\text{WISplit}_{\mathcal{L}}(\mathcal{P}, \langle \Phi = \Psi \rangle)$ решает регулярно-упорядоченные уравнения $\Phi = \Psi$.

Интерпретатор с подсчетом $WlCount_{\mathcal{L}}$

Ищет противоречия путем сравнения мультимножеств термов с левой и правой стороны уравнения.

Структура функции $Smp1$



Входной формат

$$Go(p, \mathcal{E}qs) = Main(p, Smp1(0, \varepsilon, \mathcal{E}qs));$$

- Суперкомпиляция $WlCount_{\mathcal{L}}(\mathcal{P}, \langle \Phi = \Psi \rangle)$ решает все уравнения $\Phi = \Psi$ от одной переменной.

Рекламная пауза

Классы уравнений, которые в общем не решаются CVC4 и Z3Str3, но решаются у нас:

- квадратичные уравнения, не имеющие решений $(x_1 x_2 x_3 ABABAB = AAABBB x_2 x_3 x_1)$;
- регулярно-упорядоченные уравнения, не имеющие решений $(ABxxuy = xxuyBA)$.

Уравнения от одной переменной, не решаемые CVC4 и Z3Str3 — те же регулярно-упорядоченные без решений.

Таблица тестирования

Тестовый набор	Тесты	Не завершились		
		CVC4	z3str3	WCount _{\mathcal{L}}
Track 1 (Woorpje)	200	8	13	21
Track 5 (Woorpje)	200	4	14	19
Наш набор	50	21	28	10

Среднее время работы WCount _{\mathcal{L}} : 3,5 минуты на **один тест**.

Время работы CVC4 и z3str3 — меньше 2 минут на все тесты.

Почему такая разница по времени?

Мы решаем разные задачи:

- солверы ищут одно решение;
- мы пытаемся построить описание всех решений.

Следствия

- солверы всегда решают безкоэффициентные уравнения (у них есть пустое решение);
- наш метод работает очень медленно на линейных уравнениях.

Другие интерпретаторы

- Интерпретатор, вычисляющий возможные длины переменных с последующим порождением подстановок вида $x \rightarrow s_1 \dots s_n$ (s_i — символьные переменные).
- Интерпретатор, позволяющий в некоторых случаях строить подстановки с конца уравнения.
- Интерпретатор, решающий систему линейных диофантовых уравнений на длины переменных в уравнениях (Олег Шатнюк).

Лемма об оптимальности

Лемма

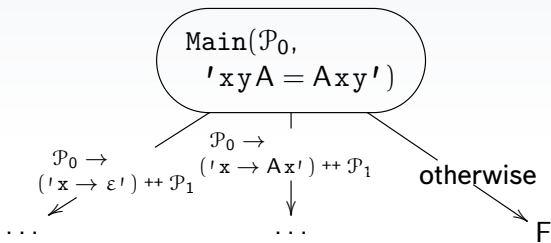
Все операции свертки в графе развертки $WI_{\mathcal{L}}(\mathcal{P}, \langle \Phi_i = \Psi_i \rangle_{i=1}^n)$ совершаются только над узлами, помеченными конфигурациями вида:

$$\text{Main}(\mathcal{P}_j, \langle \Phi_i^j = \Psi_i^j \rangle_{i=1}^{n_j})$$

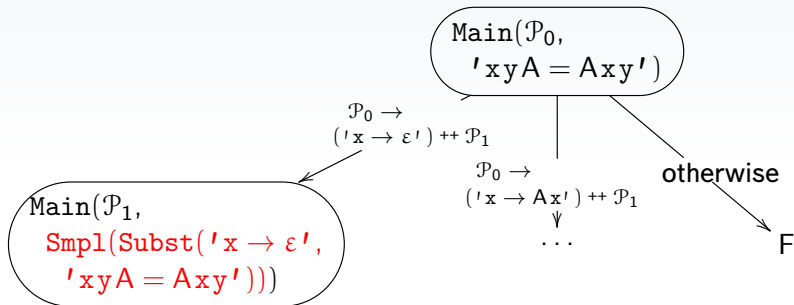
где \mathcal{P}_j , и других вхождений параметров нет.

Эта лемма определяет соответствие между графом развертки $WI_{\mathcal{L}}(\mathcal{P}, \langle \Phi_i = \Psi_i \rangle_{i=1}^n)$ и графом решения системы $\langle \Phi_i = \Psi_i \rangle_{i=1}^n$.

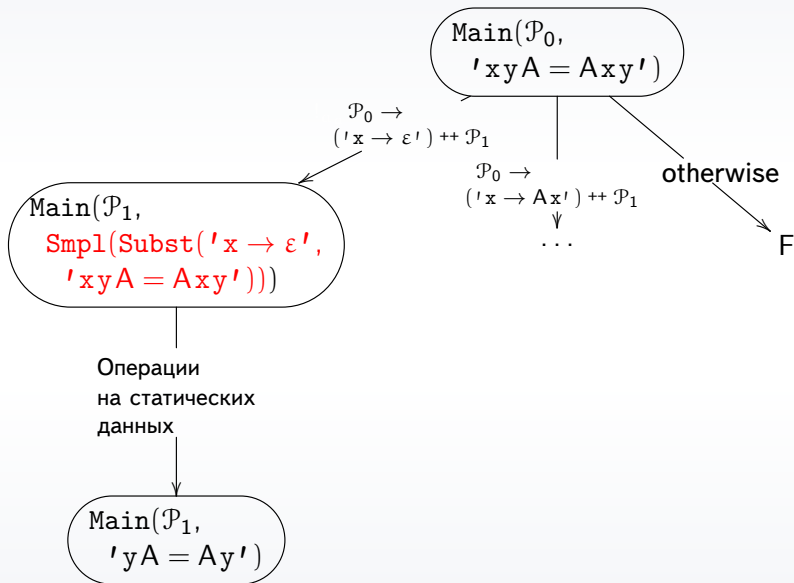
Порождение сужений



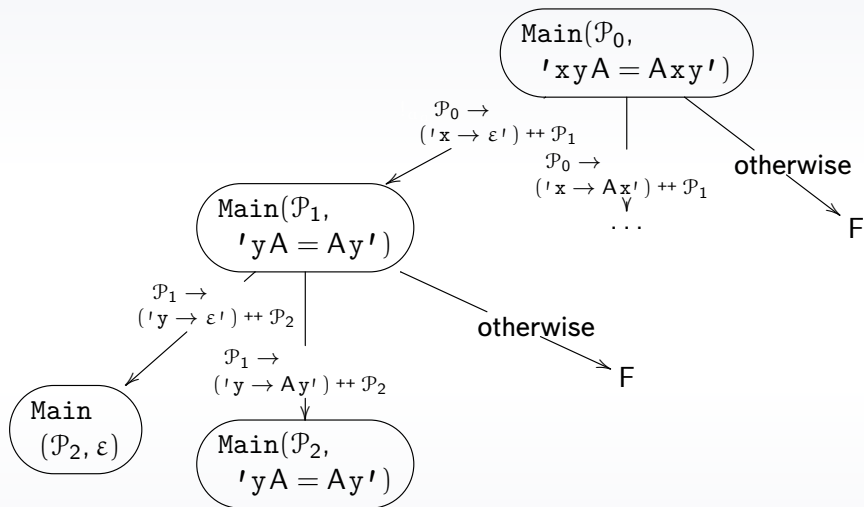
Порождение нового узла



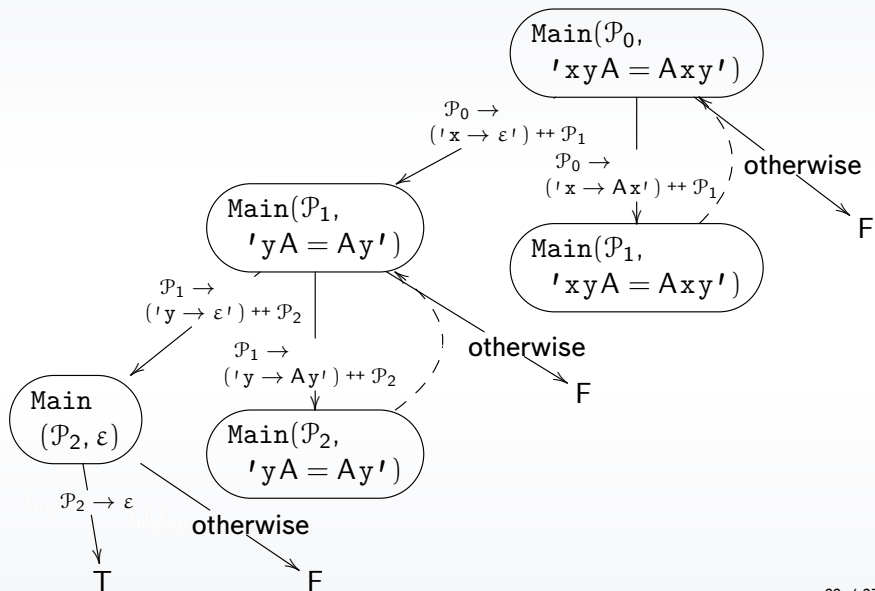
Транзитные операции



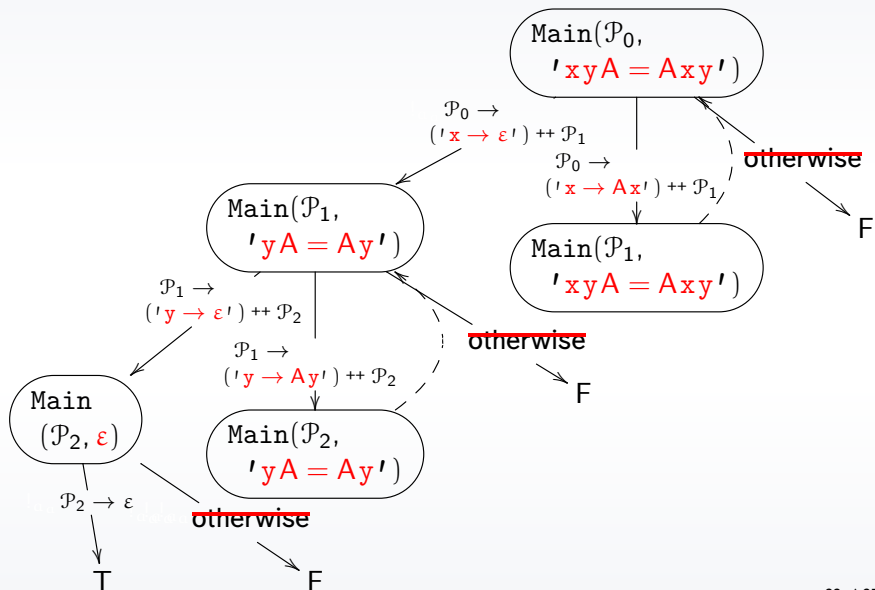
Новый шаг развертки



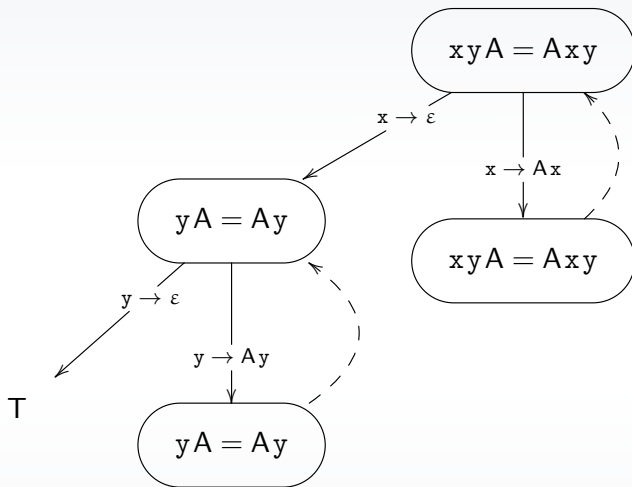
Свертка

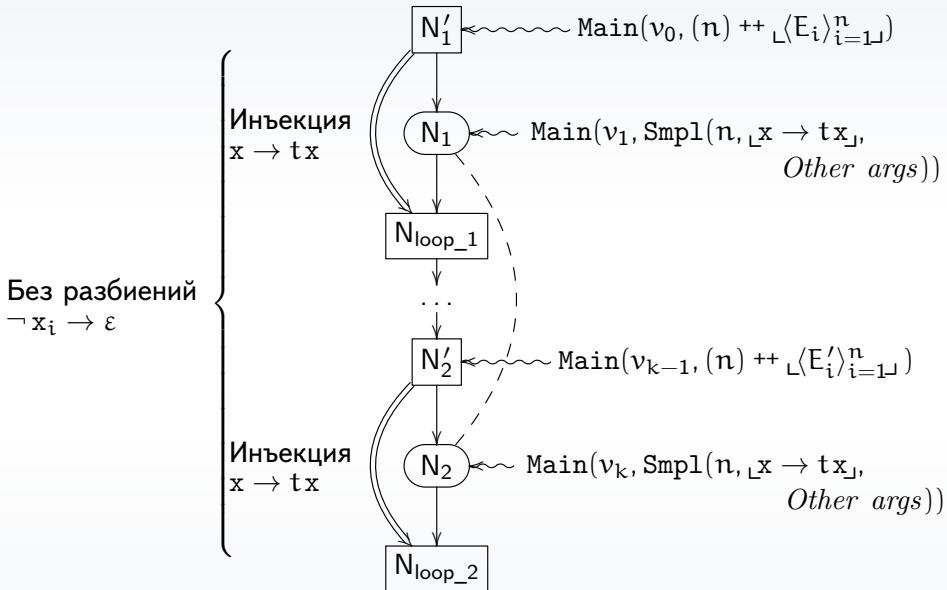


Удаление данных интерпретатора



Граф решений





О дополнительной разметке

- В интерпретаторе $WIBase_{\mathcal{L}}$ помним только применяемую последней подстановку;
- В интерпретаторах $WISplit_{\mathcal{L}}$ и $WICount_{\mathcal{L}}$ — еще и количество уравнений в последней базовой конфигурации.

Структура интерпретаторов

Если преобразования не инъективны, понадобится (очередная) дополнительная разметка.

Пример

Допустим, мы умеем не только расщеплять уравнения, но и удалять дубли. Пусть после шага $x \rightarrow Ax$ получилась система с двумя уравнениями:

$$\begin{aligned} xAAy &= yAxA \\ Ax &= xA \end{aligned}$$

Что можно сказать о системе, предшествующей ей, если известно, что в ней было два уравнения?

Основные особенности решения

- Для свертки было использовано отношение переименовки, чтобы избежать привязки к особенностям суперкомпилятора.
- Структура интерпретаторов подобрана таким образом, чтобы можно было использовать списочные стуктуры вместо рефальных.

Рекламная пауза

А если мы используем SCP4, можно просто написать в заголовке так:

```
*$Transient No;  
(или не писать ничего!)
```

...и никакой дополнительной разметки не понадобится!

Спасибо!