修士論文

ナローイングによる完全集合の自動計算

2410171　本藤　大翔

主指導教員　廣川　直
審査委員主査　廣川　直
審査委員　石井　大輔
緒方　和博
髙木　翼

北陸先端科学技術大学院大学
先端科学技術研究科
(情報科学)

令和 8 年 2 月

Abstract

In this thesis, we discuss how to solve equational unification problems automatically. Equational unification is the problem to compute equational unifiers of a given equation $s = t$ with respect to a given equational system $\mathcal{E}$. A complete set is a set of general solutions of an equation. We show examples for the equational problems and complete sets.

Examples Let us see the equation $x + y \approx \mathsf{s}(0)$ and the equational system $\mathcal{E}_0$

$$0 + x \approx x \qquad\qquad \mathsf{s}(x) + y \approx \mathsf{s}(x + y).$$

The set $\{\{x \mapsto \mathsf{s}(0), y \mapsto 0\}, \{x \mapsto 0, y \mapsto \mathsf{s}(0)\}\}$ is the set of equational unifiers of $x + y \approx \mathsf{s}(0)$ with respect to the equational system $\mathcal{E}_0$ and this set is the complete set. This set corresponds to the set of general solutions of the equation $x + y = 1$ in arithmetic. We also consider equational systems related to functional programming. Let $\mathcal{F} = \{[\,]^{(0)}, 0^{(0)}, 1^{(0)}, \mathsf{isort}^{(1)}, :^{(2)}, \mathsf{insert}^{(2)}\}$. The function symbol : is infix and right-associative. Let us consider the equation $\mathsf{isort}(xs) \approx 0 : 0 : 1 : [\,]$ and the equational system $\mathcal{E}_1$

$$\begin{aligned} \mathsf{insert}(x, [\,]) &\approx x : [\,] & \mathsf{isort}([\,]) &\approx [\,] \\ \mathsf{insert}(0, xs) &\approx 0 : xs & \mathsf{isort}(x : xs) &\approx \mathsf{insert}(x, \mathsf{isort}(xs)) \\ \mathsf{insert}(1, x : xs) &\approx x : 1 : xs. \end{aligned}$$

The set

$$\{\{xs \mapsto 0 : 0 : 1 : [\,]\}, \{xs \mapsto 1 : 0 : 0 : [\,]\}, \{xs \mapsto 0 : 1 : 0 : [\,]\}\}$$

is the set of equational unifiers of $\mathsf{isort}(xs) \approx 0 : 0 : 1 : [\,]$ with respect to the equational system $\mathcal{E}_1$ and this set is the complete set.

Contribution Computing complete sets has been studied in the equational unification theory [2, 9]. Narrowing [14] has been studied by Fay [5] and Hullot [10], and elements in a complete set can be computed by narrowing. However, narrowing cannot recognize that it has computed all elements in a complete set. In this thesis, combining narrowing and ununifiability analysis, we can compute a complete set. Ununifiablity problems can be converted to unreachability problems. Tree automata are used for unreachabilyty analysis [6]. In this thesis, we refine the tree automata technique for unreachability analysis.

目次

第 1 章

序論

項の方程式（等式）の解を求める問題は等式単一化問題として研究されている [2]。本論文では等式単一化問題の自動解法について議論する。

1.1 問題設定

等式単一化は等式系と方程式 (等式) を入力としその解としての代入 (等式単一化子) を出力する問題である。方程式（等式）の解は複数存在しうるが解の一般解の集合を完全集合という。等式単一化問題とそれの答えに対応する完全集合の例を示す。

例 1.1. 自然数上の足し算を表している等式系 $\mathcal{E}$

$$0 + x \approx x \qquad\qquad \mathsf{s}(x) + y \approx \mathsf{s}(x + y)$$

と等式 $x + y \approx \mathsf{s}(0)$ を考える。ただし、自然数をペアノ数で表す。ここで、

$$\mathsf{s}(0) + 0 \approx_{\mathcal{E}} \mathsf{s}(0 + 0) \approx_{\mathcal{E}} \mathsf{s}(0) \qquad\qquad 0 + \mathsf{s}(0) \approx_{\mathcal{E}} \mathsf{s}(0)$$

が成り立つので、等式の解として代入 $\{x \mapsto \mathsf{s}(0), y \mapsto 0\}, \{x \mapsto 0, y \mapsto \mathsf{s}(0)\}$ が求まる。ただし、$\approx_{\mathcal{E}}$ は等式系から導出される等号である。また、

$$\{\{x \mapsto \mathsf{s}(0), y \mapsto 0\}, \{x \mapsto 0, y \mapsto \mathsf{s}(0)\}\}$$

は等式 $x + y \approx \mathsf{s}(0)$ の完全集合であり、これは算術における方程式 $x + y = 1$ の一般解の集合に対応する。この集合が完全集合であることは 2 章で説明する。

例 1.1 では算術に関する方程式 (等式) の完全集合を求めたが部分的な関数型プログラ

ムに関する方程式の解も求めることができる。次の等式系は 0 と 1 を要素にもつリストに関する挿入ソートを表す。

例 1.2. シグネチャ $\mathcal{F}$ を $\{[\,]^{(0)}, 0^{(0)}, 1^{(0)}, \mathsf{isort}^{(1)}, :^{(2)}, \mathsf{insert}^{(2)}\}$ とする。ただし、関数記号 : は右連結の中置記法である。等式系 $\mathcal{E}$

$$\begin{aligned} \mathsf{insert}(x, [\,]) &\approx x : [\,] & \mathsf{isort}([\,]) &\approx [\,] \\ \mathsf{insert}(0, xs) &\approx 0 : xs & \mathsf{isort}(x : xs) &\approx \mathsf{insert}(x, \mathsf{isort}(xs)) \\ \mathsf{insert}(1, x : xs) &\approx x : 1 : xs \end{aligned}$$

と等式 $\mathsf{isort}(xs) \approx 0 : 0 : 1 : [\,]$ を考える。ここで、

$$\begin{aligned} \mathsf{isort}(0 : 0 : 1 : [\,]) &\approx_{\mathcal{E}} 0 : 0 : 1 : [\,] \\ \mathsf{isort}(0 : 1 : 0 : [\,]) &\approx_{\mathcal{E}} 0 : 0 : 1 : [\,] \\ \mathsf{isort}(1 : 0 : 0 : [\,]) &\approx_{\mathcal{E}} 0 : 0 : 1 : [\,] \end{aligned}$$

が成り立つので、代入 $\{xs \mapsto 0 : 0 : 1 : [\,]\}, \{xs \mapsto 1 : 0 : 0 : [\,]\}, \{xs \mapsto 0 : 1 : 0 : [\,]\}$ が等式の解として得られる。また、等式の解はこれ以上存在しないので集合

$$\{\{xs \mapsto 0 : 0 : 1 : [\,]\}, \{xs \mapsto 1 : 0 : 0 : [\,]\}, \{xs \mapsto 0 : 1 : 0 : [\,]\}\}$$

は完全集合である。

次の例は例 1.2 の等式系の $\mathsf{insert}(0, xs) \approx 0 : xs$ と $\mathsf{insert}(1, x : xs) \approx x : 1 : xs$ の $0, 1$ を入れ替えた等式系で挿入ソートを表す。

例 1.3. シグネチャ $\mathcal{F}$ を $\{[\,]^{(0)}, 0^{(0)}, 1^{(0)}, \mathsf{isort}^{(1)}, :^{(2)}, \mathsf{insert}^{(2)}\}$ とする。ただし、関数記号 : は右連結の中置記法である。等式系 $\mathcal{E}$

$$\begin{aligned} \mathsf{insert}(x, [\,]) &\approx x : [\,] & \mathsf{isort}([\,]) &\approx [\,] \\ \mathsf{insert}(1, xs) &\approx 1 : xs & \mathsf{isort}(x : xs) &\approx \mathsf{insert}(x, \mathsf{isort}(xs)) \\ \mathsf{insert}(0, x : xs) &\approx x : 0 : xs \end{aligned}$$

と等式 $\mathsf{isort}(xs) \approx 1 : 0 : [\,]$ を考える。ここで、

$$\begin{aligned} \mathsf{isort}(0 : 1 : [\,]) &\approx_{\mathcal{E}} 1 : 0 : [\,] \\ \mathsf{isort}(1 : 0 : [\,]) &\approx_{\mathcal{E}} 1 : 0 : [\,] \end{aligned}$$

が成り立つので、等式の解として代入 $\{xs \mapsto 0 : 1 : [\,]\}, \{xs \mapsto 1 : 0 : [\,]\}$ が得られる。等式の解はこれ以上存在しないので等式の集合

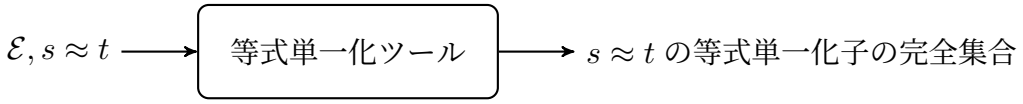
$$\{\{xs \mapsto 0 : 1 : [\,]\}, \{xs \mapsto 1 : 0 : [\,]\}\}$$

は完全集合である。等式系が正しく挿入ソートを表しているなら完全集合は空集合になるはずなので、この等式系は正しく挿入ソートを表していないことが分かる。

例 1.3 のように完全集合を求めることでプログラムの入出力を解析することができ、プログラムが正しく記述されているのか確認することができる。このことから、完全集合の自動計算を行う技術をプログラムの振る舞いの分析に応用できることが期待できる。

1.2 貢献と本論文の構成

完全集合の計算自動化は等式単一化理論分野において中心的な問題として研究されてきた [2, 9]。特に、等式単一化を行う手法としてナローイング [14] が研究され、Fay [5] と Hullot [10] によってナローイングを用いることで完全集合の要素を列挙することが可能になった。ただし、ここでいう列挙とは一般に停止しない手続きであるので完全集合の要素を全て列挙できたか判定できない。そこで、本論文ではナローイングと等式単一化不能性解析を組み合わせることで完全集合の全ての要素を列挙できたのか判定する手法を提案する。単一化不能性解析は到達不能性解析に帰着でき、到達不能性解析は木オートマトンを用いて行うことができる。本論文では木オートマトンの技術を洗練させて単一化不能性解析を行う。本研究では以下の図で示すような入力を等式系 $\mathcal{E}$、等式 $s \approx t$ として完全集合を自動的に出力する等式単一化ツールを実現する。



本論文の構成は次の通りである。2 章では項書換えに関する基本的な定義について述べる。3 章ではベーシックナローイングに基づいて完全集合を求める方法を述べる。4 章では木オートマトンによる到達不能性解析について述べる。5 章では 3 章 と 4 章で述べた手法の評価を行う。6 章では本論文のまとめと今後の課題について述べる。

第 2 章

準備

この章では以降の議論で必要な定義を [1] と [14] を基に記述する。

2.1 抽象書換え系

この節で抽象書換え系を定義する。

定義 2.1. 集合 A 上の**恒等関係**は二項関係 $\mathsf{id}_A = \{(a, a) \mid a \in A\}$ である。

定義 2.2. R, S を A 上の二項関係とする。R と S の合成 $R \cdot S$ は A 上の二項関係 $\{(x, z) \in A \times A \mid (x, y) \in R \text{ and } (y, z) \in S \text{ for some } y \in A\}$ である。

定義 2.3. 集合 A と A 上の二項関係 $\to$ からなる組 $\mathcal{A} = (A, \to)$ を**抽象書換え系**という。また、$a \to b$ で $(a, b) \in \to$ を表し、$a \not\to b$ で $(a, b) \notin \to$ を表す。

定義 2.4. $\mathcal{A} = (A, \to)$ を抽象書換え系とする。$a \in A$ が**簡約可能**であるとは $a \to b$ となる $b \in A$ が存在することをいう。a が簡約可能でないとき a を**正規形**という。

定義 2.5. $\mathcal{A} = (A, \to)$ を抽象書換え系とする。関係 $\to$ についての記法を以下の通り定義する。

- $\to^0 = \{(x, x) \mid x \in A\}$
- $\to^{n+1} = \to^n \cdot \to$
- $\to^+ = \bigcup_{n>0} \to^n$
- $\to^* = \to^+ \cup \to^0$
- $\leftarrow = \{(y, x) \mid x \to y\}$

- $\leftrightarrow = \rightarrow \cup \leftarrow$

定義 2.6. $\mathcal{A} = (A, \rightarrow)$ を抽象書換え系とする。

- $a_n \rightarrow a_{n+1}$ を満たす無限列 $(a_n)_{n\in\mathbb{N}}$ が存在しないとき $\rightarrow$ は**停止性**を持つという。
- 任意の $a, b, c \in A$ に対して、$a \rightarrow^* b$ かつ $a \rightarrow^* c$ ならば $b \rightarrow^* d$ かつ $c \rightarrow^* d$ が成り立つ $d \in A$ が存在するとき、$\rightarrow$ は**合流性**を持つという。
- $\rightarrow$ が停止性と合流性を両方もつとき、$\rightarrow$ は**完備**であるという。

2.2 項書換え系

この節で項書換え系を定義する。

定義 2.7. シグネチャ $\mathcal{F}$ は関数記号 $f^{(n)}$ の集合である。ここで、自然数 n は f の**項数**といい、$f^{(0)}$ は**定数記号**という。また、$\mathcal{T}(\mathcal{F}, \mathcal{V})$ は $\mathcal{F} \cap \mathcal{V} = \varnothing$ を満たす $\mathcal{F}$ と**変数**の集合 $\mathcal{V}$ から構成される**項**全体の集合である。項を次のように定義する。

- 変数 $x \in \mathcal{V}$ は項である。
- $f^{(n)} \in \mathcal{F}$ かつ $t_1, \ldots, t_n$ が項ならば $f(t_1, \ldots, t_n)$ は項である。

定義 2.8. 項 t の**位置**全体の集合 $\mathcal{P}\mathsf{os}(t)$ は以下の通り定義される。

$$\mathcal{P}\mathsf{os}(t) = \begin{cases} \{\epsilon\} & \text{if } t \in \mathcal{V} \\ \{\epsilon\} \cup \{i \cdot p \mid 1 \leqslant i \leqslant n \text{ and } p \in \mathcal{P}\mathsf{os}(t_i)\} & \text{if } t = f(t_1, \ldots, t_n) \end{cases}$$

ただし ϵ は空列であり $p \cdot q$ は p と q の連接である。p を t の位置とするとき p における t の**部分項** $t|_p$ は以下の通り定義される。

$$t|_p = \begin{cases} t & \text{if } p = \epsilon \\ (t_i)|_p & \text{if } t = f(t_1, \ldots, t_n) \text{ and } p = i \cdot q \end{cases}$$

また、$\mathcal{P}\mathsf{os}(t)$ の部分集合 $\mathcal{P}\mathsf{os}_{\mathcal{F}}(t)$ は集合 $\{p \in \mathcal{P}\mathsf{os}(t) \mid t|_p \notin \mathcal{V}\}$ である。t の位置 p, q について $p \cdot r = q$ となるような r が存在するとき $p \leqslant q$ で表す。

例 2.9. 項 $x + \mathsf{s}(\mathsf{0})$ の位置の集合 $\mathcal{P}\mathsf{os}(x + \mathsf{s}(\mathsf{0}))$ は $\{\epsilon, 1, 2, 21\}$ である。

定義 2.10. 項 $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ について、t に現れる変数の集合 $\mathcal{V}\mathsf{ar}(t)$ を以下のように定義する。

$$\mathcal{V}\mathsf{ar}(t) = \{t|_p \mid p \in \mathcal{P}\mathsf{os}(t) \setminus \mathcal{P}\mathsf{os}_{\mathcal{F}}(t)\}$$

また、$\mathcal{V}\mathsf{ar}(t) = \varnothing$ を満たす項 $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ を**基底項**という。基底項全体の集合を $\mathcal{T}(\mathcal{F})$ と表す。

例 2.11. $\mathcal{V}\mathsf{ar}(\mathsf{s}(x) + y) = \{x, y\}$ である。また、項 $\mathsf{s}(0)$ について $\mathcal{V}\mathsf{ar}(\mathsf{s}(0)) = \varnothing$ なので $\mathsf{s}(0)$ は基底項である。

定義 2.12. t, s を項、p を t の位置とする。このとき t の部分項 $t|_p$ を s で置き換えた項 $t[s]_p$ を以下の通り定義する。

$$t[s]_p = \begin{cases} s & \text{if } p = \epsilon \\ f(t_1, \ldots, t_i[s]_q, \ldots, t_n) & \text{if } t = f(t_1, \ldots, t_n) \text{ and } p = i \cdot q \end{cases}$$

例 2.13. 項 $x + \mathsf{s}(0)$ の位置 1 を項 $\mathsf{s}(0)$ で置き換えた項 $(x + \mathsf{s}(0))[\mathsf{s}(0)]_1$ は $\mathsf{s}(0) + \mathsf{s}(0)$ である。

定義 2.14. $\square \notin \mathcal{F}$ となる定数 $\square$ を**穴**と呼ぶ。また、穴が現れる位置がただ一つだけ存在する項を**文脈**と呼ぶ。文脈 C の穴を項 t で置き換えた項 $C[t]$ は以下の通り定義される。

$$C[t] = \begin{cases} t & \text{if } C = \square \\ f(t_1, \ldots, C'[t], \ldots, t_n) & \text{if } C = f(t_1, \ldots, C', \ldots, t_n) \end{cases}$$

定義 2.15. 関数 $\sigma \colon \mathcal{V} \to \mathcal{T}(\mathcal{F}, \mathcal{V})$ について、集合 $\mathcal{D}\mathsf{om}(\sigma) = \{x \in \mathcal{V} \mid \sigma(x) \neq x\}$ が有限集合になるとき関数 σ を**代入**といい代入 σ の 項 t への適用 $t\sigma$ を以下の通り定義する。

$$t\sigma = \begin{cases} \sigma(t) & \text{if } t \in \mathcal{V} \\ f(t_1\sigma, \ldots, t_n\sigma) & \text{if } t = f(t_1, \ldots, t_n) \end{cases}$$

代入 σ, τ の合成 $\sigma\tau$ を以下の通り定義する。

$$\sigma\tau = \{x_1 \mapsto s_1\tau, \ldots, x_n \mapsto s_n\tau\} \cup \{y_1 \mapsto t_1, \ldots, y_m \mapsto t_m\}$$

ただし、任意の $i \in \{1, \ldots, n\}, j \in \{1, \ldots, m\}$ に対して $s_i = \sigma(x_i), t_j = \tau(y_j)$ かつ $y_j \notin \{x_1, \ldots, x_n\}$ である。代入 σ, σ' に対して $\sigma\tau = \sigma'$ を満たす代入 τ が存在するとき $\sigma \leqslant \sigma'$ と書く。

例 2.16. 代入 $\sigma = \{x \mapsto \mathsf{s}(x), y \mapsto x\}$ と $\sigma' = \{x \mapsto \mathsf{s}(0), y \mapsto 0\}$ について、

$$\sigma\{x \mapsto 0\} = \{x \mapsto \mathsf{s}(x), y \mapsto x\}\{x \mapsto 0\} = \{x \mapsto \mathsf{s}(0), y \mapsto 0\} = \sigma'$$

が成り立つ。よって $\sigma \leqslant \sigma'$ が成り立つ。

定義 2.17. σ を代入、V を変数の集合とする。このとき、σ の V への制限 $\sigma{\restriction_V}$ を以下の通り定義する。

$$\sigma{\restriction_V}(x) = \begin{cases} \sigma(x) & \text{if } x \in V \\ x & \text{otherwise} \end{cases}$$

また、$\sigma{\restriction_V} = \tau{\restriction_V}$ が成り立つとき $\sigma = \tau\,[V]$ と表し、$\sigma\sigma' = \tau\,[V]$ となる代入 σ' が存在するとき $\sigma \leqslant \tau\,[V]$ と表す。

定義 2.18. 項の対 (ℓ, r) を**等式**といい $\ell \approx r$ で表す。等式の集合を**等式系**という。$\mathcal{E}$ を等式系と項 s, t について、$s = C[\ell\sigma]$ かつ $t = C[r\sigma]$ となる文脈 C と代入 σ が存在するとき $s \to_{\mathcal{E}} t$ と書く。また、$s \leftrightarrow_{\mathcal{E}}^* t$ を $s \approx_{\mathcal{E}} t$ で表す。

次に項書換え系を定義する。

定義 2.19. $\mathcal{V}\mathsf{ar}(r) \subseteq \mathcal{V}\mathsf{ar}(\ell)$ かつ $\ell \notin \mathcal{V}$ を満たす項の対 (ℓ, r) を**書換え規則**といい $\ell \to r$ で表す。書換え規則の集合を**項書換え系**という。項書換え系 $\mathcal{R}$ と項 s, t について、$s|_p = \ell\sigma$ かつ $t = s[r\sigma]_p$ となる位置 $p \in \mathcal{P}\mathsf{os}(s)$、代入 σ、規則 $\ell \to r \in \mathcal{R}$ が存在するとき $s \to_{\mathcal{R}} t$ または $s \to_{[p,\ell\to r]} t$ と書く。また、s の部分項 $\ell\sigma$ を**可約項**という。

代入の正規形を定義する。

定義 2.20. $\mathcal{R}$ を項書換え系、σ を代入とする。任意の $x \in \mathcal{D}\mathsf{om}(\sigma)$ について $\sigma(x)$ が正規形であるとき σ は**正規化**されているという。また、代入 σ が代入 τ の**正規形**であるとは、σ が正規化されており、任意の変数 $x \in \mathcal{V}$ に対して $\tau(x) \to_{\mathcal{R}}^* \sigma(x)$ が成り立つことをいう。

特別な性質を持つ項書換え系を定義する。

定義 2.21. 同一の変数が二回以上出現しない項を**線形**という。項書換え系 $\mathcal{R}$ において規則 $\ell \to r \in \mathcal{R}$ が**線形**であるとは ℓ および r が線形であることをいう。項書換え系 $\mathcal{R}$ の全ての規則が線形であるとき $\mathcal{R}$ は**線形**であるという。

定義 2.22. 全ての規則 $\ell \to r \in \mathcal{R}$ に対して $\mathcal{V}\mathsf{ar}(\ell) = \mathcal{V}\mathsf{ar}(r)$ が成り立つとき項書換え系 $\mathcal{R}$ は**変数非消去**であるという。また、$\mathcal{V}\mathsf{ar}(\ell) \neq \mathcal{V}\mathsf{ar}(r)$ が成り立つ規則 $\ell \to r \in \mathcal{R}$ が存在するとき項書換え系 $\mathcal{R}$ は**変数消去**であるという。

2.3 等式単一化

この節で等式単一化と完全集合を定義する。等式単一化は方程式の解を求めることに対応しており、完全集合は方程式の解集合に対応している。

定義 2.23. 項 s,t について $s\sigma = t\sigma$ が成り立つ代入 σ が存在するとき項 s,t は**単一化可能**であるといい $s \sim t$ で表す。この代入 σ を $s \approx t$ の**単一化子**であるという。この σ が任意の $s \approx t$ の単一化子 τ について $\sigma \leqslant \tau$ が成り立つときこの σ を**最汎単一化子**であるという。

例 2.24. 等式 $\mathsf{s}(x) \approx \mathsf{s}(\mathsf{s}(y))$ について、代入 $\sigma = \{x \mapsto \mathsf{s}(y)\}$ を考える。このとき、

$$\mathsf{s}(x)\sigma = \mathsf{s}(\mathsf{s}(y)) = \mathsf{s}(\mathsf{s}(y))\sigma$$

が成り立つ。よってこの代入 σ は $\mathsf{s}(x) \approx \mathsf{s}(\mathsf{s}(y))$ の単一化子である。また、σ は最汎単一化子である。

定義 2.25. 等式系 $\mathcal{E}$ が与えられたとき、$s\sigma \approx_{\mathcal{E}} t\sigma$ となる代入 σ を $s \approx t$ の $\mathcal{E}$ **単一化子**または**等式単一化子**という。

例 2.26. 等式系 $\mathcal{E}$

$$\begin{array}{cc} 0 + x \approx x & 0 \times x \approx 0 \\ \mathsf{s}(x) + y \approx \mathsf{s}(x + y) & \mathsf{s}(x) \times y \approx x \times y + y \end{array}$$

と等式 $x \times y \approx \mathsf{s}(0)$ と代入 $\sigma = \{x \mapsto \mathsf{s}(0), y \mapsto \mathsf{s}(0)\}$ に対して

$$(x \times y)\sigma = \mathsf{s}(0) \times \mathsf{s}(0) \approx_{\mathcal{E}} \mathsf{s}(0)$$

が成り立つので、代入 σ は $\mathcal{E}$ 単一化子である。

定義 2.27. $\mathcal{E}$ を等式系、V を変数の集合、σ, τ を代入とする。ある代入 τ' が存在して $(x\tau)\tau' \approx_{\mathcal{E}} x\sigma$ が任意の $x \in V$ で成り立つとき $\tau \leqslant_{\mathcal{E}} \sigma\,[V]$ と表す。また、$\tau(x) \approx_{\mathcal{E}} \sigma(x)$ が任意の $x \in V$ で成り立つとき $\tau =_{\mathcal{E}} \sigma\,[V]$ と表す。

定義 2.28. $\mathcal{E}$ を等式系、$\mathcal{C}$ を $s \approx t$ の $\mathcal{E}$ 単一化子の集合、$V = \mathcal{V}\mathsf{ar}(s) \cup \mathcal{V}\mathsf{ar}(t)$ とする。任意の $s \approx t$ の $\mathcal{E}$ 単一化子 σ に対してある $\tau \in \mathcal{C}$ が存在して $\tau \leqslant_{\mathcal{E}} \sigma\,[V]$ が成り立つとき $\mathcal{C}$ を $s \approx t$ の $\mathcal{E}$ 単一化子の**完全集合**という。

例 2.29. 等式系 $\mathcal{E}$

$$\mathsf{0} + x \approx x \qquad\qquad \mathsf{s}(x) + y \approx \mathsf{s}(x + y)$$

について考える。このとき、等式 $x + y \approx \mathsf{s}(\mathsf{0})$ の $\mathcal{E}$ 単一化子の集合

$$\{\{x \mapsto \mathsf{s}(\mathsf{0}), y \mapsto \mathsf{0}\}, \{x \mapsto \mathsf{0}, y \mapsto \mathsf{s}(\mathsf{0})\}\}$$

は $x + y \approx \mathsf{s}(\mathsf{0})$ の $\mathcal{E}$ 単一化子の完全集合である。

例 2.29 の代入の集合がなぜ完全集合になるのかは次の章で述べる。

第 3 章

完全集合の自動計算

この章ではナローイングに基づいて完全集合を計算する方法を述べる。3.1 節ではナローイングを定義してナローイングを用いて完全集合の要素を列挙できることを示す。3.2 節ではナローイングを行う位置を制限したベーシックナローイングを定義してナローイング同様に完全集合の要素を列挙できることを示し、さらにベーシックナローイングの不変量を議論するために導出規則を定義する。3.3 節では 3.2 節で定義した導出規則の問題点を解消するために新たな導出規則を定義する。

3.1 ナローイング

この節でナローイングを定義してナローイングで完全集合の要素が列挙できることを示す。

定義 3.1. $\mathcal{R}$ を項書換え系、$\ell \to r$ を項 t, t' と変数が共有されないように改名した $\mathcal{R}$ の規則とする。$t|_p$ と ℓ の最汎単一化子 σ が存在して、$t' = (t[r]_p)\sigma$ が成り立つとき、$t \rightsquigarrow t', t \rightsquigarrow_{[p,\ell\to r,\sigma]} t'$ または $t \rightsquigarrow_\sigma t'$ で表す。関係 $\rightsquigarrow$ を**ナローイング**といい、ナローイングの系列

$$t_0 \rightsquigarrow_{\sigma_1} t_1 \rightsquigarrow_{\sigma_2} \cdots \rightsquigarrow_{\sigma_n} t_n$$

が存在するとき、$t_0 \rightsquigarrow^* t_n$ または $t_0 \rightsquigarrow^*_\sigma t_n$ と書く。ただし、$\sigma = \sigma_1\sigma_2\cdots\sigma_n$ であり、$n = 0$ のときは $\sigma = \mathsf{id}_\mathcal{V}$ である。また、項 u と t_n の最汎単一化子 τ が存在するとき $t_0 \rightsquigarrow^\star u$ または $t_0 \rightsquigarrow^\star_\sigma u$ と書く。ただし、$\sigma = \sigma_1\sigma_2\cdots\sigma_n\tau$ であり、$n = 0$ のときは $\sigma = \tau$ である。

補題 3.2 (Lemma 3.4. [14])**.** $\mathcal{R}$ を項書換え形、s, t を項、代入 θ を正規形、V を

$\mathcal{V}\mathsf{ar}(s) \cup \mathcal{D}\mathsf{om}(\theta) \subseteq V$ が成り立つ変数の集合、$t = s\theta$ であるとする。このとき、$t \to_{\mathcal{R}}^{*} t'$ が成り立つならば $s \rightsquigarrow_{\sigma}^{*} s', t' = s'\theta', \theta = \sigma\theta' \, [V]$ かつ θ' が正規化された代入であるような項 s'、代入 θ', σ が存在する。

以下は [14] の定理 3.8 から直接得られる系である。

系 3.3. $\mathcal{R}$ を完備な項書換え系、s を項、t を正規形かつ基底項であるとする。このとき、$s\sigma \approx_{\mathcal{R}} t$ ならば $\tau \leqslant_{\mathcal{R}} \sigma \, [\mathcal{V}\mathsf{ar}(s)]$ となるナローイングの系列 $s \rightsquigarrow_{\tau}^{\star} t$ が存在する。

系 3.4. $\mathcal{R}$ を完備な項書換え系、s を項、t を基底項かつ正規形とする。このとき、集合 $\{\sigma|_{\mathcal{V}\mathsf{ar}(s)} \mid s \rightsquigarrow_{\sigma}^{\star} t\}$ は $s \approx t$ の $\mathcal{R}$ 単一化子の完全集合である。

系 3.4 では右辺が正規形かつ基底項の等式の等式単一化子の完全集合を求めているが、一般の等式の等式単一化子の完全集合も同様に系 3.4 から求めることができることを説明する。完備な項書換え系 $\mathcal{R}$ と等式 $s \approx t$ を考える。等式の右辺 t が正規形かつ基底項ではないとき、規則 $x \approx x \to \mathsf{true}$ を $\mathcal{R}$ に追加して $\mathcal{R}' = \mathcal{R} \cup \{x \approx x \to \mathsf{true}\}$ とする。このとき、$\mathcal{R}'$ は再び完備な項書換え系となっていることが知られている [14]。したがって、系 3.4 より $s \approx t$ の $\mathcal{R}$ 単一化子の完全集合は $\{\sigma|_{\mathcal{V}\mathsf{ar}(s \approx t)} \mid s \approx t \rightsquigarrow_{\sigma}^{\star} \mathsf{true}\}$ を計算すれば求まる。

3.2 ベーシックナローイング

この節でベーシックナローイングを定義して、ナローイングと同様に完全集合の要素を列挙できることを示す。また、ベーシックナローイングの不変量を議論するために導出規則を定義する。

定義 3.5. ナローイングの系列

$$t_0 \rightsquigarrow_{[p_1, \ell_1 \to r_1, \sigma_1]} t_1 \rightsquigarrow_{[p_2, \ell_2 \to r_2, \sigma_2]} \cdots \rightsquigarrow_{[p_n, \ell_n \to r_n, \sigma_n]} t_n$$

を考える。ここで、集合 $B_1, \ldots, B_n$ を以下の通り定義する。

$$B_1 = \mathcal{P}\mathsf{os}_{\mathcal{F}}(t_1)$$
$$B_{n+1} = \mathcal{B}(B_n, p_n, r_n)$$

ただし、$\mathcal{B}(B, p, r) = (B \setminus \{q \in B \mid p \leqslant q\}) \cup \{p \cdot q \mid q \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(r)\}$ である。このとき、各ナローイングステップでの位置 p_n が $p_n \in B_n$ であるときこのナローイングの系列は**ベーシック**であるという。

定義 3.6. 項書換え系 $\mathcal{R}$ と書換えの系列

$$t_0 \to_{[p_1,\ell_1\to r_1]} t_1 \to_{[p_2,\ell_2\to r_2]} \cdots \to_{[p_n,\ell_n\to r_n]} t_n$$

を考える。任意の $i \in \{0,\ldots,n\}$ に対して $p_i \in B_i$ が成り立つとき、この書換えの系列は位置の集合 $B_0 \subseteq \mathcal{P}\mathsf{os}_{\mathcal{F}}(t_0)$ に**基づく**という。ただし、$B_1,\ldots,B_n$ は定義 3.5 の関数 $\mathcal{B}$ で定義される集合である。また、書換え列 $s \to_{\mathcal{R}} t$ が $B \subseteq \mathcal{P}\mathsf{os}_{\mathcal{F}}(s)$ に基づくとき $s \xrightarrow{\mathrm{B}}_{\mathcal{R}} t$ と表す。

ここで、最内書換えを定義する。

定義 3.7. 他の可約項を真に部分項として含まない可約項を**最内可約項**という。最内可約項のみ書換える書換えを**最内書換え**という。$\mathcal{R}$ を書換え系とする。s が t に最内書換えでワンステップ書き換わるとき $s \xrightarrow{\mathrm{i}}_{\mathcal{R}} t$ で表す。

補題 3.8 ([14])**.** $\mathcal{R}$ を項書換え系、σ を正規化された代入、t を項とする。このとき、$t\sigma$ から始まる最内書換えの系列は $\mathcal{P}\mathsf{os}_{\mathcal{F}}(t)$ に基づく。

以下は [14] の定理 4.5 から直接得られる系である。

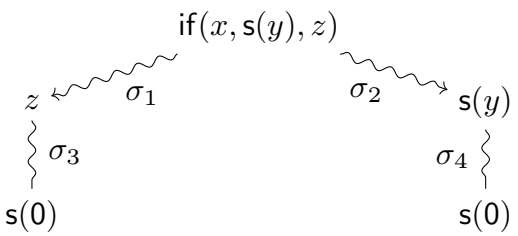
系 3.9. σ を代入、$\mathcal{R}$ を完備な項書換え系、s を項、t を正規形かつ基底項であるとする。このとき、$s\sigma \approx_{\mathcal{R}} t$ ならば $\tau \leqslant_{\mathcal{R}} \sigma\,[\mathcal{V}\mathsf{ar}(s)]$ となるベーシックナローイングの系列 $s \rightsquigarrow^{\star}_{\tau} t$ が存在する。

ベーシックナローイングで完全集合を求めてみる。次の例で記述されている図はナローイング $\rightsquigarrow$ とナローイングで得られる代入および定義 2.23 で定義した関係 $\sim$ と単一化で得られる最汎単一化子で構成されている計算木である。

例 3.10. 完備な項書換え系 $\mathcal{R}$

$$\mathsf{if}(\mathsf{true},x,y) \to x \qquad\qquad \mathsf{if}(\mathsf{false},x,y) \to y$$

を考える。等式 $\mathsf{if}(x,\mathsf{s}(y),z) \approx \mathsf{s}(\mathsf{0})$ の $\mathcal{R}$ 単一化子の完全集合の要素は次の計算木により求めることができる。



ただし各 σ_i は以下の代入である。

$$\sigma_1 = \{x \mapsto \mathsf{false}, x_1 \mapsto \mathsf{s}(y), y_1 \mapsto z\} \qquad \sigma_2 = \{x \mapsto \mathsf{true}, x_2 \mapsto \mathsf{s}(y), y_2 \mapsto z\}$$
$$\sigma_3 = \{z \mapsto \mathsf{s}(0)\} \qquad \sigma_4 = \{y \mapsto 0\}$$

計算木のルートから葉までの経路に現れる代入の合成を計算すれば

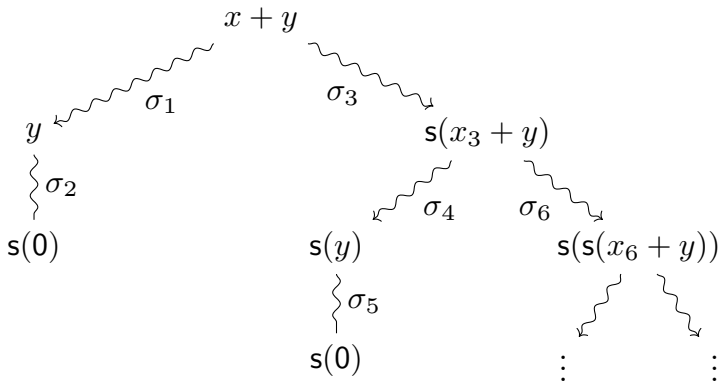
$$\sigma_1\sigma_3 = \{x \mapsto \mathsf{false}, z \mapsto \mathsf{s}(0), x_1 \mapsto \mathsf{s}(y), y_1 \mapsto \mathsf{s}(0)\}$$
$$\sigma_2\sigma_4 = \{x \mapsto \mathsf{true}, y \mapsto 0, x_2 \mapsto \mathsf{s}(0), y_2 \mapsto z\}$$

となる。計算木は有限であるので完全集合の要素は全て列挙されたことが分かる。したがって、$\{\sigma_1\sigma_3, \sigma_2\sigma_4\}$ は等式 $\mathsf{if}(x, \mathsf{s}(y), z) \approx \mathsf{s}(0)$ の $\mathcal{R}$ 単一化子の完全集合である。

例 3.11. 完備な項書換え系 $\mathcal{R}$

$$0 + x \to x \qquad \mathsf{s}(x) + y \to \mathsf{s}(x + y)$$

を考える。等式 $x + y \approx \mathsf{s}(0)$ の $\mathcal{R}$ 単一化子の完全集合は次の計算木により求めることができる。



ただし各 σ_i は以下の代入である。

$$\sigma_1 = \{x \mapsto 0, x_1 \mapsto y\} \qquad \sigma_2 = \{y \mapsto \mathsf{s}(0)\} \qquad \sigma_3 = \{x \mapsto \mathsf{s}(x_3), y_3 \mapsto y\}$$
$$\sigma_4 = \{x_1 \mapsto 0, y_4 \mapsto y\} \qquad \sigma_5 = \{y \mapsto 0\} \qquad \sigma_6 = \{x_3 \mapsto \mathsf{s}(x_6), y_6 \mapsto y\}$$

計算木のルートから葉までの経路に現れる代入の合成を計算すれば

$$\sigma_1\sigma_2 = \{x \mapsto 0, y \mapsto \mathsf{s}(0), x_1 \mapsto y\} \quad \sigma_3\sigma_4\sigma_5 = \{x \mapsto \mathsf{s}(0), y \mapsto 0, y_3 \mapsto y, y_4 \mapsto y\}$$

となり、代入 $\sigma_1\sigma_2, \sigma_3\sigma_4\sigma_5$ は等式 $x + y \approx \mathsf{s}(0)$ の $\mathcal{R}$ 単一化子の完全集合の要素であるが、上記の計算木は有限ではないので完全集合を全て列挙されたのかは判定できない。

ナローイングの系列の導出規則の定義に必要な関数をここで定義する。

定義 3.12. 項書換え系 $\mathcal{R}$、項 s,t、集合 $B \subseteq \mathcal{P}\mathsf{os}_{\mathcal{F}}(s)$、$s$ と t の最汎単一化子 $\mathrm{mgu}(s,t)$ を考える。関数 F を

$$F(\sigma, s, t, B) = \{(\sigma\tau, s', t, \mathcal{B}(B, p, r)) \mid s \rightsquigarrow_{[p,\ell\to r,\tau]} s' \text{ and } p \in B\}$$

と定義し、関数 G を

$$G(\sigma, s, t) = \{\sigma\tau \mid \tau = \mathrm{mgu}(s,t) \text{ and } \tau\colon \mathcal{V} \to \mathcal{T}(\mathcal{F}, \mathcal{V})\}$$

と定義する。ただし、関数 $\mathcal{B}$ は定義 3.5 で定義される関数である。

ベーシックナローイングの系列の不変量を議論するためベーシックナローイングの系列を導出規則で表す。

定義 3.13. 代入 σ、項 s、基底項の正規形 t、集合 $B \subseteq \mathcal{P}\mathsf{os}_{\mathcal{F}}(s)$ の組 (σ, s, t, B) の集合を S とする。このとき導出規則を次のように定める。

$$(U, S \uplus \{(\sigma, s, t, B)\}) \vdash (U \cup G(\sigma, s, t), S \cup F(\sigma, s, t, B))$$

ただし、関数 F, G は定義 3.12 で定義される関数である。(U, S) に上記の導出規則を適用し (U', S') が得られるとき $(U, S) \vdash_B (U', S')$ と表す。また、導出列

$$(U_0, S_0) \vdash_B (U_1, S_1) \vdash_B \cdots \vdash_B (U_n, S_n) \vdash_B \cdots$$

を**実行**という。

例 3.14. 完備な項書換え系 $\mathcal{R}$

$$\mathsf{0} + x \to x \qquad\qquad \mathsf{s}(x) + y \to \mathsf{s}(x_2 + y)$$

を考える。ここで、

$$F(\mathsf{id}_{\mathcal{V}}, x + y, \mathsf{s}(0), \{\epsilon\}) = \{(\sigma_1, y, \mathsf{s}(0), \varnothing), (\sigma_2, \mathsf{s}(x + y), \mathsf{s}(0), \{\epsilon, 1\})\}$$

かつ

$$G(\mathsf{id}_{\mathcal{V}}, x + y, \mathsf{s}(0)) = \varnothing$$

なので

$$\begin{aligned}&(\varnothing, \{(\mathsf{id}_{\mathcal{V}}, x + y, \mathsf{s}(0), \{\epsilon\})\})\\ \vdash_B &(\varnothing, \{(\sigma_1, y, \mathsf{s}(0), \varnothing), (\sigma_2, \mathsf{s}(x_2 + y), \mathsf{s}(0), \{\epsilon, 1\})\})\end{aligned}$$

が成り立つ。ただし各 σ_i は以下の代入である。

$$\sigma_1 = \{x \mapsto \mathsf{0}, y_1 \mapsto y\} \qquad\qquad \sigma_2 = \{x \mapsto \mathsf{s}(x_2), y_2 \mapsto y\}$$

ベーシックナローイングの導出列の不変量をここで定義する。

定義 3.15. $\mathcal{R}$ を項書換え系、σ を代入、s を項、t を基底項の正規形、集合 $B \subseteq \mathcal{P}\mathsf{os}_{\mathcal{F}}(s)$ の組 (σ, s, t, B) の集合を S、U を代入の集合、V を変数の集合とする。関数 $\mathcal{I}$ を

$$\mathcal{I}(U, V) = \{\sigma\tau{\restriction_V} \mid \sigma \in U \text{ and } \tau\colon \mathcal{V} \to \mathcal{T}(\mathcal{F}, \mathcal{V})\}$$

と定義し、関数 $\mathcal{U}$ を

$$\mathcal{U}(U, S, V) = \mathcal{I}(U, V) \cup \{\sigma\tau{\restriction_V} \mid (\sigma, s, t, B) \in S \text{ and } s\tau \xrightarrow{\mathrm{B}}{}^{*}_{\mathcal{R}} t\}$$

と定義する。

補題 3.16. 変数の集合 V、項書換え系 $\mathcal{R}$、代入 σ、項 s、基底項の正規形 t、集合 $B \subseteq \mathcal{P}\mathsf{os}_{\mathcal{F}}(s)$ を考える。$\mathcal{U}(\varnothing, \{(\sigma, s, t, B)\}, V) = \mathcal{U}(G(\sigma, s, t), F(\sigma, s, t, B), V)$ が成り立つ。ただし、関数 F, G は定義 *3.12* で定義される関数である。

証明. $\mathcal{U}(\varnothing, \{(\sigma, s, t, B), V\}) \subseteq \mathcal{U}(G(\sigma, s, t), F(\sigma, s, t, B), V)$ を示す。ここで、$\mathcal{U}(\varnothing, \{(\sigma, s, t, B), V\})$ の任意の元 σ' を固定して考える。このとき、$(\sigma, s, t, B) \in S$ かつ $s\tau \xrightarrow{\mathrm{B}}{}^{*} t$ が成り立つような代入 τ が存在して、$\sigma' = \sigma\tau{\restriction_V}$ が成り立つ。$s\tau \xrightarrow{\mathrm{B}}{}^{0} t$ のとき、s, t は単一化可能なので、s, t の最汎単一化子 θ が存在する。したがって、$\sigma\theta \in G(\sigma, s, t)$ が成り立つ。また、ある代入 θ' が存在して $\tau = \theta\theta'$ が成り立つので、$\sigma' = \sigma\tau{\restriction_V} = \sigma\theta\theta'{\restriction_V} \in \mathcal{U}(G(\sigma, s, t), F(\sigma, s, t, B), V)$ が成り立つ。$s\tau \xrightarrow{\mathrm{B}}{}^{n}_{\mathcal{R}} t$ かつ $n > 0$ のとき、ある項 s' が存在して $s\tau \xrightarrow{\mathrm{B}}_{\mathcal{R}} s' \xrightarrow{\mathrm{B}}{}^{n-1}_{\mathcal{R}} t$ が成り立つ。このとき、$(\mathcal{V}\mathsf{ar}(s) \cup V) \cap \mathcal{V}\mathsf{ar}(\ell) = \varnothing$ が成り立つように改名した $\mathcal{R}$ の規則 $\ell \to r$ と位置 $p \in B$ が存在して $s\tau|_p = \ell\theta$ が成り立つ。このとき、$p \in B$ であるので、$s\tau|_p = s|_p\tau = \ell\theta$ かつ $\mathcal{V}\mathsf{ar}(s) \cap \mathcal{V}\mathsf{ar}(\ell) = \varnothing$ が成り立ち、$s|_p$ と ℓ は単一化可能なので、$s|_p$ と ℓ の最汎単一化子 θ' が存在する。ここで、ナローイングの系列 $s \rightsquigarrow_{[p, \ell \to r, \theta']} s'$ が存在して $p \in B$ が成り立つ。よって、$(\sigma\theta', s', t, \mathcal{B}(B, p, r)) \in F(\sigma, s, t, B)$ が成り立つ。また、代入 θ' は $s|_p$ と ℓ の最汎単一化子なので、ある代入 θ'' が存在して $\theta'\theta'' = \tau \cup \theta$ が成り立つ。$V \cap \mathcal{V}\mathsf{ar}(\ell) = \varnothing$ であるので、$(\tau \cup \theta){\restriction_V} = \tau$ が成り立つ。よって、$\theta'\theta''{\restriction_V} = \tau$ が成り立つ。また、$s\theta'\theta'' \xrightarrow{\mathrm{B}}_{\mathcal{R}} s'\theta'' \xrightarrow{\mathrm{B}}{}^{n-1}_{\mathcal{R}} t$ が成り立つので、$\sigma' = \sigma\tau{\restriction_V} = \sigma\theta'\theta''{\restriction_V} \in \mathcal{U}(G(\sigma, s, t), F(\sigma, s, t, B), V)$ が成り立つ。よって、$\mathcal{U}(\varnothing, \{(\sigma, s, t, B)\}, V) \subseteq \mathcal{U}(G(\sigma, s, t), F(\sigma, s, t, B), V)$ が示され

た。次に $\mathcal{U}(G(\sigma,s,t),F(\sigma,s,t,B),V) \subseteq \mathcal{U}(\varnothing,\{(\sigma,s,t,B)\},V)$ を示す。ここで、$\mathcal{U}(G(\sigma,s,t),F(\sigma,s,t,B),V)$ の任意の元 σ' を固定して考える。$\sigma' \in \mathcal{I}(G(\sigma,s,t),V)$ のとき $\theta \in G(\sigma,s,t)$ とある代入 θ' が存在して $\sigma' = \theta\theta'\restriction_V$ が成り立つ。このとき、s と t の単一化子 τ が存在して、$\theta = \sigma\tau$ が成り立つ。したがって、$(\sigma,s,t,B) \in G(\sigma,s,t)$ かつ $s\tau = t$ が成り立ち、$s\tau$ は基底項なので $s\tau\theta' = t$ が成り立つ。よって、$\sigma' = \theta\theta'\restriction_V = \sigma\tau\theta'\restriction_V \in \mathcal{U}(\varnothing,\{(\sigma,s,t,B)\},V)$ が成り立つ。$\sigma' \notin \mathcal{I}(G(\sigma,s,t),V)$ のとき、$(\tau,s',t,\mathcal{B}(B,p,r)) \in F(\sigma,s,t,B)$ かつ $s'\theta \xrightarrow{\mathrm{B}}{}^{*}_{\mathcal{R}} t$ が成り立つ代入 θ が存在して $\sigma' = \tau\theta\restriction_V$ が成り立つ。このとき、ナローイングの系列 $s \rightsquigarrow_{[p,\ell\to r,\theta']} s'$ が存在して $p \in B, \tau = \sigma\theta'$ が成り立つ。ここで、$s\theta'\theta \xrightarrow{\mathrm{B}}_{\mathcal{R}} s'\theta \xrightarrow{\mathrm{B}}{}^{*}_{\mathcal{R}} t$ が成り立つので、$\sigma' = \tau\theta\restriction_V = \sigma\theta'\theta\restriction_V \in \mathcal{U}(\varnothing,\{(\sigma,s,t,B)\},V)$ が成り立つ。よって、$\mathcal{U}(G(\sigma,s,t),F(\sigma,s,t,B),V) \subseteq \mathcal{U}(\varnothing,\{(\sigma,s,t,B)\},V)$ が示された。 □

定理 3.17. s を項、t を基底項の正規形、$\mathcal{R}$ を完備な項書換え系とする。このとき、$U_0 = \varnothing, S_0 = \{(\mathsf{id}_{\mathcal{V}}, s, t, \mathcal{P}\mathsf{os}_{\mathcal{F}}(s))\}$ とする。導出列

$$(U_0,S_0) \vdash_B (U_1,S_1) \vdash_B \cdots \vdash_B (U_n,S_n)$$

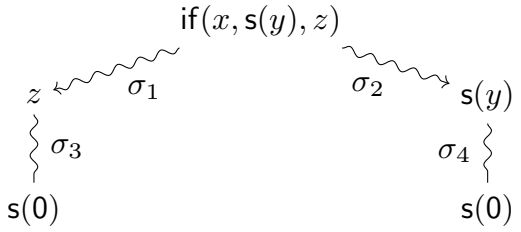
が存在して $S_n = \varnothing$ が成り立つとき U_n は $s \approx t$ の $\mathcal{R}$ 単一化子の完全集合である。

後述する定理 3.23 に包含されるので証明を割愛する。

例 3.18. 完備な項書換え系 $\mathcal{R}$

$$\mathsf{if}(\mathsf{true},x,y) \to x \qquad\qquad \mathsf{if}(\mathsf{false},x,y) \to y$$

を考える。等式 $\mathsf{if}(x,\mathsf{s}(y),z) \approx \mathsf{s}(\mathsf{0})$ の $\mathcal{R}$ 単一化子の完全集合は次の計算木により求めることができる。



ただし各 σ_i は以下の代入である。

$$\sigma_1 = \{x \mapsto \mathsf{false}, x_1 \mapsto \mathsf{s}(y), y_1 \mapsto z\} \qquad \sigma_2 = \{x \mapsto \mathsf{true}, x_2 \mapsto \mathsf{s}(y), y_2 \mapsto z\}$$
$$\sigma_3 = \{z \mapsto \mathsf{s}(\mathsf{0})\} \qquad \sigma_4 = \{y \mapsto \mathsf{0}\}$$

先に定義した導出規則の観点では、この計算木は次の実行に対応する。

$$\begin{aligned}&(\varnothing, \{(\mathsf{id}_{\mathcal{V}}, \mathsf{if}(x, \mathsf{s}(y), z), \mathsf{s}(0), \{\epsilon, 2\})\})\\ \vdash_B\ &(\varnothing, \{(\sigma_1, z, \mathsf{s}(0), \varnothing), (\sigma_2, \mathsf{s}(y), \mathsf{s}(0), \varnothing)\})\\ \vdash_B\ &(\{\sigma_1\sigma_3, \sigma_2\sigma_4\}, \varnothing)\end{aligned}$$

合成を計算すれば

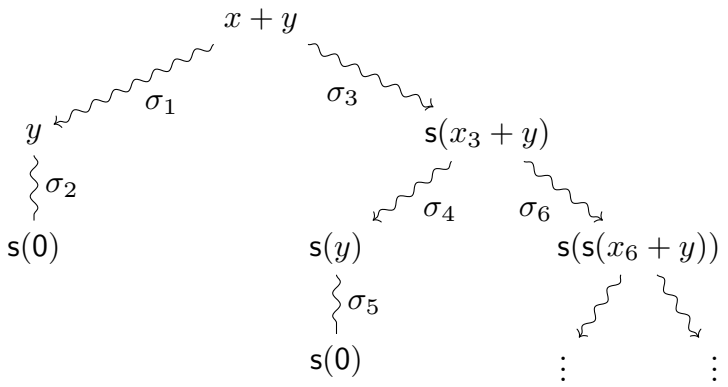
$$\begin{aligned}\sigma_1\sigma_3 &= \{x \mapsto \mathsf{false}, z \mapsto \mathsf{s}(0), x_1 \mapsto \mathsf{s}(y), y_1 \mapsto \mathsf{s}(0)\}\\ \sigma_2\sigma_4 &= \{x \mapsto \mathsf{true}, y \mapsto 0, x_2 \mapsto \mathsf{s}(0), y_2 \mapsto z\}\end{aligned}$$

となる。定理 3.17 より $\{\sigma_1\sigma_3, \sigma_2\sigma_4\}$ は完全集合となる。

例 3.19. 完備な項書換え系 $\mathcal{R}$

$$0 + x \to x \qquad\qquad \mathsf{s}(x) + y \to \mathsf{s}(x + y)$$

を考える。等式 $x + y \approx \mathsf{s}(0)$ の $\mathcal{R}$ 単一化子の完全集合は次の計算木により求めることができる。



ただし各 σ_i は以下の代入である。

$$\begin{aligned}&\sigma_1 = \{x \mapsto 0, x_1 \mapsto y\} \qquad \sigma_2 = \{y \mapsto \mathsf{s}(0)\} \qquad \sigma_3 = \{x \mapsto \mathsf{s}(x_3), y_3 \mapsto y\}\\ &\sigma_4 = \{x_3 \mapsto 0, y_4 \mapsto y\} \qquad \sigma_5 = \{y \mapsto 0\} \qquad \sigma_6 = \{x_3 \mapsto \mathsf{s}(x_6), y_6 \mapsto y\}\end{aligned}$$

この計算木は次の実行に対応する。

$$
\begin{aligned}
&(\varnothing, \{(\mathsf{id}_{\mathcal{V}}, x+y, \mathsf{s}(\mathsf{0}), \{\epsilon\})\}) \\
\vdash_B &(\varnothing, \{(\sigma_1, y, \mathsf{s}(\mathsf{0}), \varnothing), (\sigma_3, \mathsf{s}(x_3+y), \mathsf{s}(\mathsf{0}), \{\epsilon, 1\})\}) \\
\vdash_B &(\{\sigma_1\sigma_2\}, \{(\sigma_3, \mathsf{s}(x_3+y), \mathsf{s}(\mathsf{0}), \{\epsilon, 1\})\}) \\
\vdash_B &(\{\sigma_1\sigma_2\}, \{(\sigma_3\sigma_4, \mathsf{s}(y), \{\epsilon\}), (\sigma_3\sigma_6, \mathsf{s}(\mathsf{s}(x_2+y)), \mathsf{s}(\mathsf{0}), \{\epsilon, 1, 11\})\}) \\
\vdash_B &(\{\sigma_1\sigma_2, \sigma_3\sigma_4\sigma_5\}, \{(\sigma_3\sigma_6, \mathsf{s}(\mathsf{s}(x_6+y)), \mathsf{s}(\mathsf{0}), \{\epsilon, 1, 11\})\}) \\
\vdash_B &\cdots
\end{aligned}
$$

しかし、この実行は有限列でないので等式 $x+y \approx \mathsf{s}(\mathsf{0})$ の $\mathcal{R}$ 単一化子の完全集合を実行から求めることはできない。

3.3 完全集合の計算

3.2 節ではベーシックナローイングの不変量を議論するために導出規則を定義したが、ベーシックナローイングが完全集合の全ての要素を列挙できたのか判定できない問題は解決していない。3.3 節では新たな導出規則を定義してこの問題を解決する。

定義 3.20. $\mathcal{R}$ を項書換え系、s, t を項、U を代入の集合、S を代入と項と位置の集合の組の集合、$B \subseteq \mathcal{P}\mathsf{os}_{\mathcal{F}}(s)$、$\mathrm{mgu}(s,t)$ を s と t の最汎単一化子とする。このとき導出規則を次のように定める。

$$
\frac{U, S \uplus \{(\sigma, s, t, B)\}}{U \cup G(\sigma, s, t), S \cup F(\sigma, s, t, B)} \qquad \frac{U, S \uplus \{(\sigma, s, t, B)\}}{U, S} \text{ if } s \not\rightsquigarrow^{\star} t
$$

ただし、関数 F, G は定義 3.12 で定義される関数である。(U, S) に上記の導出規則を適用し (U', S') が得られるとき $(U, S) \vdash_C (U', S')$ と表す。また、導出列

$$
(U_0, S_0) \vdash_C (U_1, S_1) \vdash_C \cdots \vdash_C (U_n, S_n) \vdash_C \cdots
$$

を**実行**という。

補題 3.21. 代入 σ、項 s、基底項の正規形 t、集合 $B \subseteq \mathcal{P}\mathsf{os}_{\mathcal{F}}(s)$ の組 (σ, s, t, B) の集合を S、代入の集合を U、変数の集合を V とする。このとき、$(U, S) \vdash_C (U', S')$ ならば $\mathcal{U}(U, S, V) = \mathcal{U}(U', S', V)$ が成り立つ。

証明. S の元 (σ, s, t, B) に導出規則を適用すると仮定する。このとき、

$$
\mathcal{U}(U, S, V) = \mathcal{U}(U, S \setminus \{(\sigma, s, t, B)\}, V) \cup \mathcal{U}(\varnothing, \{(\sigma, s, t, B)\}, V)
$$

かつ

$$\mathcal{U}(U', S', V) = \mathcal{U}(U, S \setminus \{(\sigma, s, t, B)\}, V) \cup \mathcal{U}(G(\sigma, s, t), F(\sigma, s, t, B), V)$$

が成り立つので、$\mathcal{U}(\varnothing, \{(\sigma, s, t, B)\}, V) = \mathcal{U}(G(\sigma, s, t), F(\sigma, s, t, B), V)$ を示せば良い。導出規則

$$\frac{U, S \uplus \{(\sigma, s, t, B)\}}{U, S} \text{ if } s \not\rightsquigarrow^\star t$$

を適用したとき、

$$\mathcal{U}(\varnothing, \{(\sigma, s, t, B)\}, V) = \mathcal{U}(G(\sigma, s, t), F(\sigma, s, t, B), V) = \varnothing$$

なので $\mathcal{U}(U, S, V) = \mathcal{U}(U', S', V)$ が成り立つ。また、導出規則

$$\frac{U, S \uplus \{(\sigma, s, t, B)\}}{U \cup G(\sigma, s, t), S \cup F(\sigma, s, t, B)}$$

を適用したとき、補題 3.16 より $\mathcal{U}(U, S, V) = \mathcal{U}(U', S', V)$ が成り立つ。 □

補題 3.22. 代入 σ、項 s、基底項の正規形 t、集合 $B \subseteq \mathcal{P}\mathsf{os}_{\mathcal{F}}(s)$ の組 (σ, s, t, B) の集合を S、代入の集合を U、変数の集合を V とする。このとき、$(U, S) \vdash_C^* (U', S')$ ならば $\mathcal{U}(U, S, V) = \mathcal{U}(U', S', V)$ が成り立つ。

証明. 任意の自然数 n について、$(U, S) \vdash_C^n (U', S')$ ならば $\mathcal{U}(U, S, V) = \mathcal{U}(U', S', V)$ が成り立つことを n に関する帰納法で示す。$n = 0$ のとき $U = U', S = S'$ なので $\mathcal{U}(U, S, V) = \mathcal{U}(U', S', V)$ は成り立つ。$n > 0$ のとき、ある (U'', S'') が存在して

$$(U, S) \vdash_C^{n-1} (U'', S'') \vdash_C (U', S')$$

が成り立つ。帰納法の仮定より $\mathcal{U}(U, S, V) = \mathcal{U}(U'', S'', V)$ が成り立ち、補題 3.21 より $\mathcal{U}(U'', S'', V) = \mathcal{U}(U', S', V)$ が成り立つので、$\mathcal{U}(U, S, V) = \mathcal{U}(U', S', V)$ が成り立つ。 □

定理 3.23. s を項、t を基底項の正規形、$\mathcal{R}$ を完備な項書換え系とする。このとき、$U_0 = \varnothing, S_0 = \{(\mathsf{id}_{\mathcal{V}}, s, t, \mathcal{P}\mathsf{os}_{\mathcal{F}}(s))\}$ とする。導出列

$$(U_0, S_0) \vdash_C (U_1, S_1) \vdash_C \cdots \vdash_C (U_n, S_n)$$

が存在して $S_n = \varnothing$ が成り立つとき U_n は $s \approx t$ の $\mathcal{R}$ 単一化子の完全集合である。

証明. まず、U_n が $s \approx t$ の $\mathcal{R}$ 単一化子の集合であることを示す。任意の U_n の元 σ を固定する。このとき、$\sigma\restriction_{\mathcal{V}\mathsf{ar}(s)} \in \mathcal{I}(U_n, \mathcal{V}\mathsf{ar}(s)) = \mathcal{U}(U_n, \varnothing, \mathcal{V}\mathsf{ar}(s))$ が成り立ち、補題 3.22 より $\mathcal{U}(U_n, \varnothing, \mathcal{V}\mathsf{ar}(s)) = \mathcal{U}(U_0, S_0, \mathcal{V}\mathsf{ar}(s)) = \{\tau\restriction_{\mathcal{V}\mathsf{ar}(s)} \mid s\tau \xrightarrow{\mathrm{B}}{}^{*}_{\mathcal{R}} t\}$ が成り立つ。したがって、$\sigma\restriction_{\mathcal{V}\mathsf{ar}(s)} \in \{\tau\restriction_{\mathcal{V}\mathsf{ar}(s)} \mid s\tau \xrightarrow{\mathrm{B}}{}^{*}_{\mathcal{R}} t\}$ が成り立つ。よって、σ は $s \approx t$ の $\mathcal{R}$ 単一化子である。σ は任意なので U_n は $s \approx t$ の $\mathcal{R}$ 単一化子の集合である。次に、U_n が $s \approx t$ の $\mathcal{R}$ 単一化子の完全集合であることを示す。任意の $s \approx t$ の $\mathcal{R}$ 単一化子 σ について、$\tau \leqslant_{\mathcal{R}} \sigma\,[\mathcal{V}\mathsf{ar}(s)]$ が成り立つ $\tau \in U_n$ が存在することを示せば良い。任意の $s \approx t$ の $\mathcal{R}$ 単一化子 σ を固定して考える。代入 σ' を σ の正規形とすると $\sigma' =_{\mathcal{R}} \sigma\,[\mathcal{V}\mathsf{ar}(s)]$ が成り立つので、$s\sigma' \approx_{\mathcal{R}} t\sigma'$ が成り立つ。このとき、$\mathcal{R}$ は完備な項書換え系であり、t は基底項かつ正規形であるので $s\sigma' \xrightarrow{\mathrm{i}}{}^{*}_{\mathcal{R}} t$ が成り立つ。ここで、補題 3.8 より、書換え列 $s\sigma' \xrightarrow{\mathrm{i}}{}^{*}_{\mathcal{R}} t$ は $\mathcal{P}\mathsf{os}_{\mathcal{F}}(s)$ に基づくので $\sigma \in \mathcal{U}(\varnothing, \{(\mathsf{id}_{\mathcal{V}}, s, t, \mathcal{P}\mathsf{os}_{\mathcal{F}}(s))\}, \mathcal{V}\mathsf{ar}(s))$ が成り立つ。補題 3.22 より $\mathcal{U}(\varnothing, \{(\mathsf{id}_{\mathcal{V}}, s, t, \mathcal{V}\mathsf{ar}(s))\}, \mathcal{V}\mathsf{ar}(s)) = \mathcal{U}(U_n, S_n, \mathcal{V}\mathsf{ar}(s)) = \mathcal{I}(U_n, \mathcal{V}\mathsf{ar}(s))$ が成り立つので、ある代入 $\tau \in U_n$ が存在して $\tau \leqslant_{\mathcal{R}} \sigma'\,[\mathcal{V}\mathsf{ar}(s)]$ が成り立つ。$\tau \leqslant_{\mathcal{R}} \sigma'\,[\mathcal{V}\mathsf{ar}(s)]$ と $\sigma' =_{\mathcal{R}} \sigma\,[\mathcal{V}\mathsf{ar}(s)]$ を併せると $\tau \leqslant_{\mathcal{R}} \sigma\,[\mathcal{V}\mathsf{ar}(s)]$ が成り立つ。よって、U_n は $s \approx t$ の $\mathcal{R}$ 単一化子の完全集合である。 □

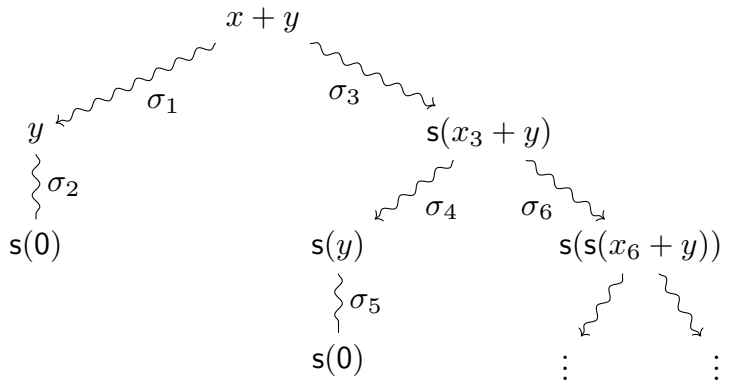
例 3.24. 等式系 $\mathcal{E}$

$$\mathsf{0} + x \approx x \qquad\qquad \mathsf{s}(x) + y \approx \mathsf{s}(x + y)$$

を考える。等式 $x + y \approx \mathsf{s}(\mathsf{0})$ の $\mathcal{E}$ 単一化子の完全集合を計算してみる。等式系 $\mathcal{E}$ に対応する完備な項書換え系 $\mathcal{R}$ は

$$\mathsf{0} + x \to x \qquad\qquad \mathsf{s}(x) + y \to \mathsf{s}(x + y)$$

である。次の計算木が存在する。



ただし各 σ_i は以下の代入である。

$$\sigma_1 = \{x \mapsto \mathsf{0}, x_1 \mapsto y\} \qquad \sigma_2 = \{y \mapsto \mathsf{s}(\mathsf{0})\} \qquad \sigma_3 = \{x \mapsto \mathsf{s}(x_3), y_1 \mapsto y\}$$
$$\sigma_4 = \{x_1 \mapsto \mathsf{0}, y_2 \mapsto y\} \qquad \sigma_5 = \{y \mapsto \mathsf{0}\} \qquad \sigma_6 = \{x_1 \mapsto \mathsf{s}(x_6), y_2 \mapsto y\}$$

この計算木は次の実行に対応する。

$$\begin{aligned}
&(\varnothing, \{(\mathsf{id}_{\mathcal{V}}, x + y, \mathsf{s}(\mathsf{0}), \{\epsilon\})\}) \\
\vdash_B &(\varnothing, \{(\sigma_1, y, \mathsf{s}(\mathsf{0}), \varnothing), (\sigma_3, \mathsf{s}(x_3 + y), \mathsf{s}(\mathsf{0}), \{\epsilon, 1\})\}) \\
\vdash_B &(\{\sigma_1\sigma_2\}, \{(\sigma_3, \mathsf{s}(x_3 + y), \mathsf{s}(\mathsf{0}), \{\epsilon, 1\})\}) \\
\vdash_B &(\{\sigma_1\sigma_2\}, \{(\sigma_3\sigma_4, \mathsf{s}(y), \{\epsilon\}), (\sigma_3\sigma_6, \mathsf{s}(\mathsf{s}(x_6 + y)), \mathsf{s}(\mathsf{0}), \{\epsilon, 1, 11\})\}) \\
\vdash_B &(\{\sigma_1\sigma_2, \sigma_3\sigma_4\sigma_5\}, \{(\sigma_3\sigma_6, \mathsf{s}(\mathsf{s}(x_6 + y)), \mathsf{s}(\mathsf{0}), \{\epsilon, 1, 11\})\}) \\
\vdash_B &\cdots
\end{aligned}$$

この実行は有限列でないので等式 $x + y \approx \mathsf{s}(\mathsf{0})$ の $\mathcal{R}$ 単一化子の完全集合を実行から求めることはできない。

例 3.24 の計算木は有限ではないため $\vdash_B$ の有限の実行列は得られないが、任意の代入 σ に対して $\mathsf{s}(\mathsf{s}(x_6 + y))\sigma \not\rightarrow^*_{\mathcal{R}} \mathsf{s}(\mathsf{0})$ が成り立つことを示すことができれば $\mathsf{s}(\mathsf{s}(x_6 + y)) \not\rightarrow^\star \mathsf{s}(\mathsf{0})$ が成り立ち、有限な計算木が得られる。この到達不能性解析は tcap 関数 [7] で行うことができる。ここで、tcap 関数を定義する。

定義 3.25. $\mathcal{R}$ を項書換え系とする。このとき関数 $\mathrm{tcap}_{\mathcal{R}}$ は以下の通り定義される。

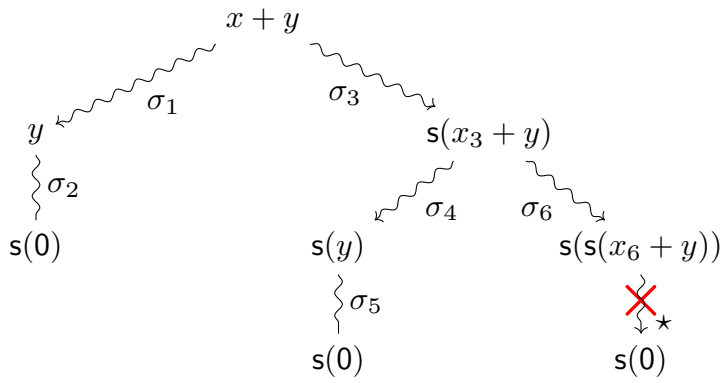
$$\mathrm{tcap}_{\mathcal{R}}(t) = \begin{cases} u & \text{if } t = f(t_1, \ldots, t_n) \text{ and } u \not\sim \ell \text{ for all } \ell \to r \in \mathcal{R} \\ y & \text{otherwise} \end{cases}$$

$u = f(\mathrm{tcap}_{\mathcal{R}}(t_1), \ldots, \mathrm{tcap}_{\mathcal{R}}(t_n))$ かつ y は新規変数である。

次の定理から tcap 関数を用いて到達不能性解析を行うことができる。

定理 3.26 ([7], [17])**.** 項 s, t と項書換え系 $\mathcal{R}$ について、$\mathrm{tcap}_{\mathcal{R}}(s) \not\sim t$ ならば $s\sigma \not\rightarrow^*_{\mathcal{R}} t$ が任意の代入 $\sigma\colon \mathcal{V} \to \mathcal{T}(\mathcal{F}, \mathcal{V})$ に対して成り立つ。

例 3.27 (例 3.24 の続き)**.** $\mathrm{tcap}_{\mathcal{R}}(\mathsf{s}(\mathsf{s}(x + y))) = \mathsf{s}(\mathsf{s}(z)) \not\sim \mathsf{s}(\mathsf{0})$ なので $\mathsf{s}(\mathsf{s}(x + y))\sigma \not\rightarrow^*_{\mathcal{R}} \mathsf{s}(\mathsf{0})$ が任意の代入 σ で成り立つ。したがって、$\mathsf{s}(\mathsf{s}(x_2 + y)) \not\rightarrow^\star \mathsf{s}(\mathsf{0})$ が成り立つ。次の計算木が得られる。



ただし各 σ_i は以下の代入である。

$$\sigma_1 = \{x \mapsto \mathsf{0}, x_1 \mapsto y\} \qquad \sigma_2 = \{y \mapsto \mathsf{s}(\mathsf{0})\} \qquad \sigma_3 = \{x \mapsto \mathsf{s}(x_3), y_3 \mapsto y\}$$
$$\sigma_4 = \{x_1 \mapsto \mathsf{0}, y_4 \mapsto y\} \qquad \sigma_5 = \{y \mapsto \mathsf{0}\} \qquad \sigma_6 = \{x_3 \mapsto \mathsf{s}(x_6), y_6 \mapsto y\}$$

この計算木は次の実行に対応する。

$$\begin{aligned}
&(\varnothing, \{(\mathsf{id}_{\mathcal{V}}, x+y, \mathsf{s}(\mathsf{0}), \{\epsilon\})\}) \\
\vdash_C &(\varnothing, \{(\sigma_1, y, \mathsf{s}(\mathsf{0}), \varnothing), (\sigma_3, \mathsf{s}(x_3+y), \mathsf{s}(\mathsf{0}), \{\epsilon, 1\})\}) \\
\vdash_C &(\{\sigma_1\sigma_2\}, \{(\sigma_3, \mathsf{s}(x_3+y), \mathsf{s}(\mathsf{0}), \{\epsilon, 1\})\}) \\
\vdash_C &(\{\sigma_1\sigma_2\}, \{(\sigma_3\sigma_4, \mathsf{s}(y), \{\epsilon\}), (\sigma_3\sigma_6, \mathsf{s}(\mathsf{s}(x_6+y)), \mathsf{s}(\mathsf{0}), \{\epsilon, 1, 11\})\}) \\
\vdash_C &(\{\sigma_1\sigma_2, \sigma_3\sigma_4\sigma_5\}, \{(\sigma_3\sigma_6, \mathsf{s}(\mathsf{s}(x_6+y)), \mathsf{s}(\mathsf{0}), \{\epsilon, 1, 11\})\}) \\
\vdash_C &(\{\sigma_1\sigma_2, \sigma_3\sigma_4\sigma_5\}, \varnothing)
\end{aligned}$$

合成を計算すれば

$$\sigma_1\sigma_2 = \{x \mapsto \mathsf{0}, y \mapsto \mathsf{s}(\mathsf{0}), x_1 \mapsto y\} \quad \sigma_3\sigma_4\sigma_5 = \{x \mapsto \mathsf{s}(\mathsf{0}), y \mapsto \mathsf{0}, y_3 \mapsto y, y_4 \mapsto y\}$$

となり、定理 3.23 より $\{\sigma_1\sigma_3, \sigma_3\sigma_4\sigma_5\}$ は完全集合となる。また、例 2.29 の集合

$$\{\{x \mapsto \mathsf{0}, y \mapsto \mathsf{s}(\mathsf{0})\}, \{x \mapsto \mathsf{s}(\mathsf{0}), y \mapsto \mathsf{0}\}\}$$

は $\{\sigma_1\sigma_3, \sigma_3\sigma_4\sigma_5\}$ の要素を $\mathcal{V}\mathsf{ar}(x+y)$ に制限した集合なので完全集合である。

tcap 関数を用いて到達不能性解析を行ったが到達不能性を示すことができない例が数多く存在する。

完備な項書換え系 $\mathcal{R}$

$$\begin{aligned}
\mathsf{0} + x &\to x & \mathsf{0} \times x &\to \mathsf{0} \\
\mathsf{s}(x) + y &\to \mathsf{s}(x+y) & \mathsf{s}(x) \times y &\to x \times y + y
\end{aligned}$$

を考える。$(x \times y + y + y)\sigma \not\to^*_{\mathcal{R}} \mathsf{s}(0)$ が任意の代入 σ で成り立つが、

$$\mathrm{tcap}_{\mathcal{R}}(x \times y + y + y) = z \sim \mathsf{s}(0)$$

なので tcap 関数を用いて到達不能性を示すことはできない。

到達不能性解析で用いられるもう一つの手法として木オートマトンが存在する。次の章では木オートマトンを用いて到達不能性解析を行う手法を説明する。

第 4 章

到達不能性問題

ある基底項に書換えで到達する全ての基底項を計算する手法としてグローイング近似 [16, 11] と木オートマトン完備化 [6] が存在し、到達不能性解析に使用される。グローイング近似では線形項書換え系を要求されるため、項書換え系を線形項書換え系に変換する必要がある。そのため、グローイング近似で計算された基底項がある基底項に書換えで到達しても、もとの項書換え系で到達しない可能性がある。一方、木オートマトン完備化を用いる手法は項書換え系を線形項書換え系に変換する必要は無いが、変数消去項書換え系を扱うことができない。そこで、4 章では木オートマトン完備化を用いて到達不能性解析を行う手法を変数消去項書換え系を扱えるように拡張を行う。4.1 節では木オートマトンの定義と木オートマトンに関する基本的な性質を説明する。4.2 節では木オートマトン完備化による到達不能性解析を説明する。4.3 節では木オートマトンに型導入をすることで 4.2 節で説明する手法の効率化を行う。

4.1 木オートマトン

この節では木オートマトンの定義および木オートマトンに関する基本的な性質を説明する。

定義 4.1. $\mathcal{F} \cap Q = \varnothing$ を満たす状態の有限集合 Q、$Q_f \subseteq Q$ を満たす最終状態の集合 Q_f、遷移規則の集合 Δ の四つ組 $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ を木オートマトンという。また、遷移規則は

- $f(p_1, \ldots, p_n) \to q$
- $p \to q$

のいずれかの形である。ただし、$f^{(n)} \in \mathcal{F}, p_1, \ldots, p_n, p, q \in Q$ である。また、$p \to q$ を**イプシロン遷移**という。

定義 4.2. 木オートマトン $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ と基底項 $t \in \mathcal{T}(\mathcal{F})$ について、ある最終状態 $q \in Q_f$ が存在して、$t \to_\Delta^* q$ が成り立つとき、t は**受理**されるという。$\mathcal{A}$ に受理されるすべての基底項 $t \in \mathcal{T}(\mathcal{F})$ を $L(\mathcal{A})$ と表す。基底項の集合 L について $L = L(\mathcal{A})$ が成り立つとき L は**正則**であるという。

補題 4.3 ([3])**.** 基底項の集合は正則である。

例 4.4. 木オートマトン $\mathcal{A} = (\{\mathsf{0}^{(0)}, \mathsf{s}^{(1)}\}, \{q_1, q_2\}, \{q_2\}, \Delta)$ を考える。ただし、遷移規則の集合 Δ は

$$\mathsf{0} \to q_1 \qquad\qquad \mathsf{s}(q_1) \to q_2$$

である。このとき、$\mathsf{s}(\mathsf{0}) \to_\Delta \mathsf{s}(q_1) \to_\Delta q_2$ が成り立つので $\{\mathsf{s}(\mathsf{0})\}$ は正則である。

定義 4.5. $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ を木オートマトンとする。Δ がイプシロン遷移を含まず任意の Δ の元 $\ell \to q$, $\ell' \to q'$ に対して、$\ell = \ell'$ ならば $q = q'$ が成り立つとき $\mathcal{A}$ は**決定性木オートマトン**であるという。

ここで、任意の木オートマトンを同じ正則言語を受理する決定性木オートマトンに変換する方法を説明する。

補題 4.6 ([3])**.** 任意の正則言語はイプシロン遷移を遷移規則に含まない木オートマトンに受理される。

木オートマトン $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ をイプシロン遷移を遷移規則に含まない木オートマトンに変換する具体的な手続きを説明する。遷移規則の集合 Δ' を

$$\Delta' = \{f(q_1, \ldots, q_n) \to q' \mid f(q_1, \ldots, q_n) \to q \in \Delta, q \to_\Delta^* q' \text{ and } q' \in Q\}$$

とする。このとき、$\mathcal{B} = (\mathcal{F}, Q, Q_f, \Delta')$ とすると $L(\mathcal{A}) = L(\mathcal{B})$ が成り立つ。

補題 4.7 ([3])**.** 任意の正則言語は決定性木オートマトンに受理される。

補題 4.6 より任意の木オートマトンはイプシロン遷移を遷移規則の集合に含まない木オートマトンに変換できるので $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ をイプシロン遷移を遷移規則の集合に含まない木オートマトンとする。このとき $\mathcal{A}$ を決定性木オートマトンに変換する具体的な手続きを説明する。$Q' = \{A \mid A \subseteq Q\}$ を状態の集合、$Q'_f = \{A \in Q', \mid A \cap Q_f \neq \varnothing\}$ を最

終状態の集合、Δ' を遷移規則 $f(A_1, \ldots, A_n) \to B$ の集合とする。ただし、$A_1, \ldots, A_n \in Q$ かつ

$$B = \{q \mid f(p_1, \ldots, p_n) \to q \in \Delta \text{ and } p_i \in A_i \text{ for all } i \in \{1, \ldots, n\}\}$$

が成り立つ。このとき、$\mathcal{B} = (\mathcal{F}, Q', Q'_f, \Delta')$ とすると $L(\mathcal{A}) = L(\mathcal{B})$ が成り立つ。また、この構成方法を**部分集合構成**という。

例 4.8. $\mathcal{F} = \{\mathsf{0}^{(0)}, \mathsf{s}^{(1)}\}$ とする。木オートマトン $\mathcal{A} = (\mathcal{F}, \{q_1, q_2\}, \{q_3\}, \Delta_A)$ を考える。ただし、遷移規則の集合 Δ_A は

$$\mathsf{0} \to q_1 \qquad \mathsf{s}(q_1) \to q_2 \qquad q_2 \to q_3$$

である。ここで、補題 4.6 から木オートマトン $\mathcal{B} = (\mathcal{F}, \{q_1, q_2\}, \{q_3\}, \Delta_B)$ を得る。ただし、遷移規則の集合 Δ_B は

$$\mathsf{0} \to q_1 \qquad \mathsf{s}(q_1) \to q_2 \qquad \mathsf{s}(q_1) \to q_3$$

である。次に、補題 4.7 から木オートマトン $\mathcal{C} = (\mathcal{F}, \{\{q_1\}, \{q_1, q_2\}\}, \{\{q_3, q_3\}\}, \Delta_C)$ を得る。ただし、遷移規則の集合 Δ_C は

$$\mathsf{0} \to \{q_1\} \qquad \mathsf{s}(\{q_1\}) \to \{q_2, q_3\}$$

である。また、$\mathcal{C}$ は決定性木オートマトンであり $L(\mathcal{A}) = L(\mathcal{C})$ が成り立つ。

定義 4.9. 木オートマトン $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ について、任意の $f^{(n)} \in \mathcal{F}, p_1, \ldots, p_n \in Q$ に対して遷移規則 $f(p_1, \ldots, p_n) \to q \in \Delta$ が存在するとき $\mathcal{A}$ は**完全に定義されている**という。

定義 4.10. 項書換え系 $\mathcal{R}$ と基底項の集合 L について、基底項の集合 $(\to^*_{\mathcal{R}})[L]$ を

$$(\to^*_{\mathcal{R}})[L] = \{s \mid s \to^*_{\mathcal{R}} t \text{ and } t \in L\}$$

と定義する。

4.2 到達不能性解析

木オートマトンを用いて到達不能性解析を行う方法をこの節で説明する。

定義 4.11. $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ を決定性木オートマトン、$\mathcal{R}$ を項書換え系とする。任意の $\ell \to r \in \mathcal{R}$ と 任意の代入 $\theta\colon \mathcal{V} \to Q$ と任意の $q \in Q$ に対して以下の含意が成り立つとき $\mathcal{A}$ は $\ell \to r$ に対して**互換性**を持つという。

$$r\theta \to_\Delta^* q \implies \ell\theta \to_\Delta^* q$$

また、もし $\mathcal{A}$ が任意の $\mathcal{R}$ の元に対して互換性を持ち以下の条件のいずれかが成り立つとき $\mathcal{A}$ は $\mathcal{R}$ に対して**互換性**を持つという。

1. 任意の $\mathcal{R}$ の元 $\ell \to r$ に対して $\mathcal{V}\mathsf{ar}(\ell) \subseteq \mathcal{V}\mathsf{ar}(r)$ が成り立つ。
2. 任意の基底項がある状態に到達可能である。

定理 4.12. $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ を決定性木オートマトン、$\mathcal{R}$ を項書換え系とする。このとき、$\mathcal{A}$ が $\mathcal{R}$ に対して互換性を持つならば $(\to_\mathcal{R}^*)[L(\mathcal{A})] \subseteq L(\mathcal{A})$ が成り立つ。

証明. $\mathcal{A}$ が $\mathcal{R}$ に対して互換性を持つと仮定する。$u \in L(\mathcal{A})$ かつ $t \to_\mathcal{R} u$ であるとき $t \in L(\mathcal{A})$ が成り立つことを示せば良い。まず、ある $q_f \in Q_f$ が存在して $u \to_\Delta^* q_f$ が成り立ち、ある文脈 C が存在して $t = C[\ell\sigma] \to_\mathcal{R} C[r\sigma] = u$ が成り立つ。ここで、任意の変数 $x \in \mathcal{V}\mathsf{ar}(r)$ について状態 q_x を次のように定義する。$\mathcal{A}$ は決定性木オートマトンであり、$\sigma(x)$ は遷移 $C[r\sigma] \to_\Delta^* q_f$ の中で同じ状態に到達するので、状態 q_x を遷移 $C[r\sigma] \to_\Delta^* q_f$ の中で $\sigma(x)$ が到達する状態とする。ここで、任意の規則 $\ell \to r \in \mathcal{R}$ について $\mathcal{V}\mathsf{ar}(\ell) \subseteq \mathcal{V}\mathsf{ar}(r)$ であると仮定する。代入 $\tau = \{x \mapsto q_x \mid x \in \mathcal{V}\mathsf{ar}(\ell)\}$ を定義する。このとき、任意の変数 $x \in \mathcal{V}\mathsf{ar}(\ell)$ に対して $\sigma(x) \to_\Delta^* \tau(x)$ が成り立つ。ある状態 p が存在して $u \to_\Delta^* C[r\tau] \to_\Delta^* C[p] \to_\Delta^* q_f$ が成り立つので、$\mathcal{A}$ が $\mathcal{R}$ に対して互換性を持つことから $\ell\tau \to_\Delta^* p$ が成立する。よって、$t = C[\ell\sigma] \to_\Delta^* C[\ell\tau] \to_\Delta^* q_f$ が成り立つ。したがって、$t \in L(\mathcal{A})$ が成立する。次に、任意の基底項がある状態に到達すると仮定する。代入 $\tau = \{x \mapsto q_x \mid x \in \mathcal{V}\mathsf{ar}(r)\} \cup \{x \mapsto q \mid x \in \mathcal{V}\mathsf{ar}(\ell) \setminus \mathcal{V}\mathsf{ar}(r), q \in Q \text{ and } \sigma(x) \to_\mathcal{A}^* q\}$ を定義する。このとき、任意の変数 $x \in \mathcal{V}\mathsf{ar}(\ell)$ に対して $\sigma(x) \to_\Delta^* \tau(x)$ が成り立つ。ある状態 p が存在して $u \to_\Delta^* C[r\tau] \to_\Delta^* C[p] \to_\Delta^* q_f$ が成り立つので、$\mathcal{A}$ が $\mathcal{R}$ に対して互換性を持つことから $\ell\tau \to_\Delta^* p$ が成立する。よって、$t = C[\ell\sigma] \to_\Delta^* C[\ell\tau] \to_\Delta^* q_f$ が成り立つ。したがって、$t \in L(\mathcal{A})$ が成立する。 □

次に、木オートマトン $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ を $(\to_\mathcal{R}^*)[L(\mathcal{B})] \subseteq L(\mathcal{B})$ が成り立つ木オートマトン $\mathcal{B}$ に変換する手続きである**木オートマトン完備化**を説明する。

次のステップを $\mathcal{A}_{0,0} = (\mathcal{F}, Q_0, Q_0', \Delta_0), Q_0 = Q, Q_0' = Q_f, \Delta_0 = \Delta$ と置いて停止するまで行う。

1. $\Delta_{n+1} = \Delta_n \cup \{\ell\sigma \to r\sigma \mid \ell \to r \in \mathcal{R} \text{ and } \sigma\colon \mathcal{V} \to Q_m\}$ を得る。
2. $\Delta_{n+2} = \{\ell \to r{\downarrow_{\Delta_{n+1}}} \mid \ell \to r \in \Delta_{n+1}\}$ を得る。
3. $\Delta_{n+3} = \{\ell \to r \mid \ell \to r \in \Delta_{n+2} \text{ and } r \in Q_m\}$ を得る。
4. $\Delta_{n+4} = \{\ell{\downarrow_{\Delta_{n+3}}} \to r \mid \ell \to r \in \Delta_{n+3}\}$ を得る。
5. $\Delta_{n+5} = \{\ell \to r \mid \ell \to r \in \Delta_{n+4} \text{ and } \ell \neq r\}$ を得る。
6. $\Delta_{n+5} = \Delta_{n+4}$ が成り立たなければ 8 に進む。$\Delta_{n+5} = \Delta_{n+4}$ が成り立つならば $\mathcal{R}$ が非変数消去であるとき手続きを終える。$\mathcal{R}$ が変数消去であるとき 7 に進む。
7. $\mathcal{A}_{n+5,m} = (\mathcal{F}, Q_m, Q'_m, \Delta_{n+5})$ が完全に定義されているならば手続きを終える。完全に定義されていないならば $\mathcal{A}_{n+6,m} = (\mathcal{F}, Q_m, Q'_m, \Delta_{n+6})$ が完全に定義されるように Δ_{n+5} に遷移規則を加え Δ_{n+6} に変形して 9 に進む。
8. 任意の $\ell \to r \in \Delta_{n+5}$ について $\ell{\downarrow_{\Delta_{n+6}}} = r$ が成り立つように Δ_{n+5} に遷移規則を加え Δ_{n+6} を得る。
9. 木オートマトン $\mathcal{A}_{n+6,m} = (\mathcal{F}, Q_m, Q'_m, \Delta_{n+6})$ を部分集合構成で決定性木オートマトン $\mathcal{A}_{n+7,m+1} = (\mathcal{F}, Q_{m+1}, Q'_{m+1}, \Delta_{n+7})$ に変形する。1 に戻る。

この操作を停止するまで続けて得られた木オートマトン $\mathcal{B} = (\mathcal{F}, Q, Q_f, \Delta_n)$ は $\mathcal{R}$ に対して互換性を持つので定理 4.12 より $\mathcal{B}$ は $(\to^*_{\mathcal{R}})[L(\mathcal{B})] \subseteq L(\mathcal{B})$ を満たす。また、この手続きの第 7 ステップがなければ通常の木オートマトン完備化 (Genet [6]) の手続きと同じである。

次の系を用いて木オートマトンによる到達可能性解析を行う。次の系は [3] の定理 1.7.11 の証明から得られる。

系 4.13 ([3])**.** $\mathcal{A} = (\mathcal{F}, Q, Q_f, \Delta)$ を決定性木オートマトンとする。このとき、任意の代入 $\sigma\colon \mathcal{V} \to Q$ について $t\sigma \to^*_{\Delta} q \notin Q_f$ であるならば任意の代入 $\tau\colon \mathcal{V} \to \mathcal{T}(\mathcal{F})$ について $t\tau \notin L(\mathcal{A})$ が成り立つ。

例 4.14. 項書換え系 $\mathcal{R} = \{\mathsf{0} + x \to x, \mathsf{s}(x) + y \to \mathsf{s}(x + y)\}$、等式 $\mathsf{s}(\mathsf{s}(x)) \approx \mathsf{s}(\mathsf{0})$ と $\mathsf{s}(\mathsf{0})$ を受理する木オートマトン $\mathcal{A} = (\mathcal{F}, \{q_0, q_1\}, \{q_1\}, \{\mathsf{0} \to q_0, \mathsf{s}(q_0) \to q_1\})$ を考える。$\mathcal{A}$ に木オートマトン完備化を行うと $\mathcal{A}' = (\mathcal{F}, \{q_0, q_1\}, \{q_1\}, \Delta)$ が得られる。ただし、$\mathcal{F} = \{0^{(0)}, \mathsf{s}^{(1)}\}$ で Δ は

$$\mathsf{0} \to q_0 \qquad \mathsf{s}(q_0) \to q_1 \qquad \begin{array}{l} q_0 + q_0 \to q_0 \\ q_0 + q_1 \to q_1 \\ q_1 + q_0 \to q_1 \end{array}$$

である。また、定理 4.12 より $\mathcal{A}'$ は $(\to^*_{\mathcal{R}})[L(\mathcal{A}')] \subseteq L(\mathcal{A}')$ を満たす。このとき、任意

の代入 $\sigma\colon \mathcal{V} \to \{q_0, q_1\}$ について $t\sigma \to_\Delta^* q \notin \{q_1\}$ が成り立ち、$\mathcal{A}'$ は決定性木オートマトンなので系 4.13 より任意の代入 $\tau\colon \mathcal{V} \to \mathcal{T}(\mathcal{F})$ について $\mathsf{s}(\mathsf{s}(x))\tau \notin L(\mathcal{A}')$ が成り立つ。$\mathcal{A}'$ は $(\to_\mathcal{R}^*)[L(\mathcal{A}')] \subseteq L(\mathcal{A}')$ を満たすので、任意の代入 $\tau\colon \mathcal{V} \to \mathcal{T}(\mathcal{F})$ について $\mathsf{s}(\mathsf{s}(x))\tau \not\to_\mathcal{R}^* \mathsf{s}(\mathsf{0})$ が成り立つ。

4.3 型導入

木オートマトンが項書換え系に対して互換性を持つことを示すときに項の変数に状態を代入する必要がある。一般に、項の変数に状態を割り当てる組み合わせは変数の個数の冪乗通り存在するため効率上問題がある。この節では木オートマトンに型を導入することでこの問題を部分的に解決する。

例 4.15. 項書換え系 $\mathcal{R}$

$$\mathsf{h}(\mathsf{g},\mathsf{a},\mathsf{a}) \to \mathsf{h}(\mathsf{f},\mathsf{a},\mathsf{a}) \qquad\qquad \mathsf{h}(x,\mathsf{b},y) \to \mathsf{h}(x,y,y)$$

と項 $\mathsf{h}(\mathsf{g},\mathsf{b},\mathsf{b})$ を受理する木オートマトン $\mathcal{A} = (\{\mathsf{a}^{(0)}, \mathsf{b}^{(0)}, \mathsf{f}^{(0)}, \mathsf{g}^{(0)}, \mathsf{h}^{(3)}\}, Q, \{q_2\}, \Delta)$ について考える。ただし、$Q = \{\mathsf{0},\mathsf{1},\mathsf{2}\}, \Delta = \{\mathsf{g} \to \mathsf{0}, \mathsf{b} \to \mathsf{1}, \mathsf{h}(\mathsf{0},\mathsf{1},\mathsf{1}) \to \mathsf{2}\}$ である。このとき、$\mathcal{A}$ は $\mathcal{R}$ に対して互換性を持つ。例えば $\mathcal{A}$ が $\mathcal{R}$ の規則 $\mathsf{h}(x,\mathsf{b},y) \to \mathsf{h}(x,y,y)$ に対して互換性を持つことは

$$\mathsf{h}(q_1,q_2,q_2) \to_\Delta^* q_3 \implies \mathsf{h}(q_1,\mathsf{b},q_2) \to_\Delta^* q_3$$

が任意の $(q_1,q_2,q_3) \in Q \times Q \times Q$ で成り立つことを示せば良い。

$\mathsf{h}(x,y,y)$ の変数に状態を割り当てる組み合わせは 9 通りある。一般の場合は項 t に状態 Q を割り当てる組み合わせは $|\mathcal{V}\mathsf{ar}(t)|^{|Q|}$ 通りある。ここで、木オートマトンに型を導入することで効率的に木オートマトンの互換性を持つことを示すことが可能になる。

定理 4.16 ([18])**.** 多ソート項書換え系 $\mathcal{R}$ とソート除去演算子 Θ *[18]* について、項 s が型付けされるならば $s \to_\mathcal{R}^* t$ と $\Theta(s) \to_{\Theta(\mathcal{R})}^* \Theta(t)$ は同値である。

この定理から木オートマトンに型を導入した後に木オートマトンが項書換え系に対して互換性を持つのか示すことができる。したがって、項の変数の型と一致しない状態を割り当てる必要がなくなるため変数への状態の割り当てを全て試す必要がなくなる。

例 4.17 (例 4.15 の続き)**.** 項書換え系 $\mathcal{R}$ と項 $\mathsf{h}(x, y, y)$ は次の多ソートシグネチャ上の項書換え系と項として考えることができる。

$$\mathsf{h}\colon A \times B \times B \to C \qquad \mathsf{f}\colon A \qquad \mathsf{g}\colon A \qquad \mathsf{a}\colon B \qquad \mathsf{b}\colon B$$

また、状態 $\mathsf{0}, \mathsf{1}, \mathsf{2}$ に対して次のように型が割り当たる。

$$\mathsf{0}\colon A \qquad \mathsf{1}\colon B \qquad \mathsf{2}\colon C$$

$\mathsf{h}(q_1, q_2, q_2) \to_\Delta^* q_3$ であることから q_1, q_2, q_3 に対して次のように型が割り当たる。

$$q_1\colon A \qquad q_2\colon B \qquad q_3\colon C$$

したがって、$\mathcal{A}$ が $\mathcal{R}$ の規則 $\mathsf{h}(x, \mathsf{b}, y) \to \mathsf{h}(x, y, y)$ に対して互換性を持つことは

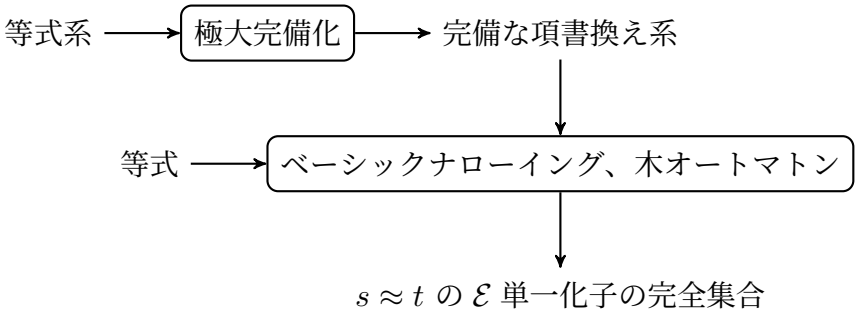
$$\mathsf{h}(q_1, q_2, q_2) \to_\Delta^* q_3 \implies \mathsf{h}(q_1, \mathsf{b}, q_2) \to_\Delta^* q_3$$

が $(q_1, q_2, q_3) \in \{\mathsf{0}\} \times \{\mathsf{1}\} \times \{\mathsf{2}\}$ で成り立つことを示せば良い。

第 5 章

評価

3 章、4 章で述べた提案手法の評価を行うため等式の完全集合を求めるツールを実装した。本ツールは入力を等式と等式系、出力を完全集合とする。また、定理 3.23 の仮定を成り立たせるために等式系を完備な項書換え系に変換する必要があるため極大完備化 [12] を実装した。極大完備化で使用する簡約順序は GWPO[15] である。次の図は入力を等式系 $\mathcal{E}$、等式 $s \approx t$、出力を $s \approx t$ の $\mathcal{E}$ 単一化子の完全集合とした本ツールの概要図である。



なお実験に使用したツールと実験データは以下のウェブページにある。

`https://www.jaist.ac.jp/~hirokawa/software/hondo/`

本ツールがどの様に完全集合を計算するのか具体的に見ていく。等式 $x \times y \approx \mathsf{s}(0)$ と等式系 $\mathcal{E}$

$$\begin{array}{cc} 0 + x \approx x & 0 \times x \approx x \\ \mathsf{s}(x) + y \approx \mathsf{s}(x + y) & \mathsf{s}(x \times y) \approx x \times y + y \end{array}$$

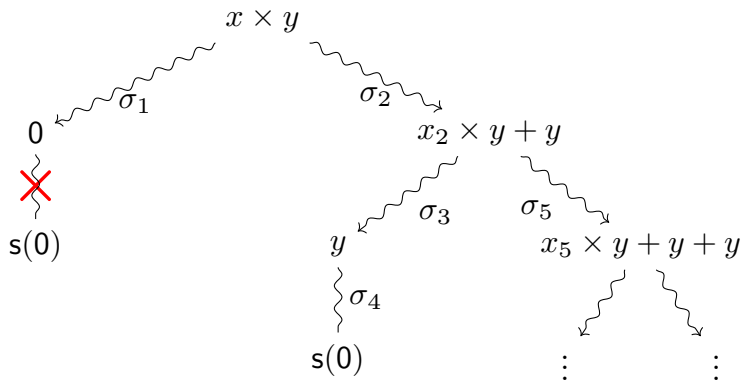
を考える。この等式系に完備化を行うと完備な項書換え系

$$\begin{array}{ll} \mathsf{0}+x \to x & \mathsf{0}\times x \to x \\ \mathsf{s}(x)+y \to \mathsf{s}(x+y) & \mathsf{s}(x\times y) \to x\times y+y \end{array}$$

が得られる。すると、$\mathsf{s}(\mathsf{0})$ は正規形かつ基底項なので定理 3.23 より導出列

$$(U_0, S_0) \vdash_C \cdots \vdash_C (U_n, S_n) \vdash_C \cdots$$

を計算すれば完全集合が求まる。ただし、$U_0 = \varnothing, S_0 = \{(\mathsf{id}_{\mathcal{V}}, x\times y, \mathsf{s}(\mathsf{0}), \mathcal{P}\mathsf{os}_{\mathcal{F}}(x+y))\}$ である。次の計算木は上の導出列に対応する。



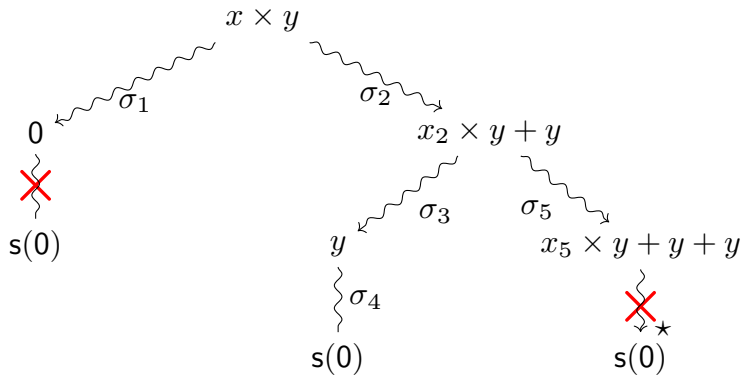
ただし各 σ_i は以下の代入である。

$$\begin{array}{lll} \sigma_1 = \{x \mapsto \mathsf{0}, x_1 \mapsto y\} & \sigma_2 = \{x \mapsto \mathsf{s}(x_2), y_2 \mapsto y\} & \sigma_3 = \{x_2 \mapsto \mathsf{0}, y_3 \mapsto y\} \\ \sigma_4 = \{y \mapsto \mathsf{s}(\mathsf{0})\} & \sigma_5 = \{x_2 \mapsto \mathsf{s}(x_5), y_5 \mapsto y\} & \end{array}$$

ここで、$\mathcal{A} = (\mathcal{F}, \{q_0, q_1\}, \{q_1\}, \{\mathsf{0} \to q_0, \mathsf{s}(q_0) \to q_1\})$ は $\mathsf{s}(\mathsf{0})$ を受理する木オートマトンで、木オートマトン完備化を行うと $\mathcal{A}' = (\mathcal{F}, \{q_0, q_1, q_2\}, \{q_1\}, \Delta)$ が得られる。ただし、$\mathcal{F} = \{0^{(0)}, \mathsf{s}^{(1)}\}$ で Δ は

$$\begin{array}{llll} \mathsf{0} \to q_0 & \mathsf{s}(q_0) \to q_1 & q_0+q_0 \to q_0 & q_0\times q_0 \to q_0 \\ & \mathsf{s}(q_1) \to q_2 & q_0+q_1 \to q_1 & q_0\times q_1 \to q_0 \\ & \mathsf{s}(q_2) \to q_2 & q_1+q_0 \to q_1 & q_1\times q_0 \to q_0 \\ & & q_1+q_1 \to q_2 & q_1\times q_1 \to q_1 \\ & & q_0+q_2 \to q_2 & q_2\times q_0 \to q_0 \\ & & q_2+q_0 \to q_2 & q_0\times q_2 \to q_0 \\ & & q_1+q_2 \to q_2 & q_1\times q_2 \to q_2 \\ & & q_2+q_1 \to q_2 & q_2\times q_1 \to q_2 \\ & & q_2+q_2 \to q_2 & q_2\times q_2 \to q_2 \end{array}$$

である。ここで、$\mathcal{A}'$ は完全に定義されているので定理 4.12 より $(\to_{\mathcal{R}}^*)[L(\mathcal{A}')] \subseteq L(\mathcal{A}')$ が成り立ち、任意の代入 $\sigma\colon \mathcal{V} \to \{q_0, q_1, q_2\}$ に対して、$(x_5 \times y + y + y)\sigma \not\to_{\Delta}^* q_1$ であるので系 4.13 より任意の代入 $\tau\colon \mathcal{V} \to \mathcal{T}(\mathcal{F})$ に対して $(x_5 \times y + y + y)\tau \not\to_{\mathcal{R}}^* \mathsf{s}(0)$ が成り立つ。したがって、$x_5 \times y + y + y \not\rightsquigarrow^\star \mathsf{s}(0)$ が成り立つので次の計算木が得られる。



ただし各 σ_i は以下の代入である。

$$\sigma_1 = \{x \mapsto 0, x_1 \mapsto y\} \quad \sigma_2 = \{x \mapsto \mathsf{s}(x_2), y_2 \mapsto y\} \quad \sigma_3 = \{x_2 \mapsto 0, y_3 \mapsto y\}$$
$$\sigma_4 = \{y \mapsto \mathsf{s}(0)\} \quad \sigma_5 = \{x_2 \mapsto \mathsf{s}(x_5), y_5 \mapsto y\}$$

合成を計算すれば

$$\sigma_2\sigma_3\sigma_4 = \{x \mapsto \mathsf{s}(0), y \mapsto \mathsf{s}(0), y_2 \mapsto y, y_3 \mapsto y\}$$

となる。この計算木に対応する実行

$$(U_0, S_0) \vdash_C \cdots \vdash_C (U_n, S_n)$$

が存在するので、ある自然数 n が存在して $S_n = \varnothing$ が成り立つ。よって、定理 3.23 より $x \times y \approx \mathsf{s}(0)$ の $\mathcal{E}$ 単一化子の完全集合 $U_n = \{\sigma_2\sigma_3\sigma_4\}$ が求まる。

5.1 評価

等式単一化問題を 30 問使用して本ツールの評価を行い表 5.1 にまとめた。使用したコンピュータは Intel Core i7-1165G7 4.70GHz プロセッサと 16GB のメモリを搭載したものである。60 秒以内に完全集合が出力されれば yes を出力されなければ timeout を result に表示する。また、完全集合が出力された際に経過した秒数を time (s) に表示する。また、評価で使用した入力ファイルとその出力を付録 A に記述した。

表 5.1

file	result	time (s)
ackermann.ari	yes	0.753
add.ari	yes	0.062
append.ari	yes	0.063
average.ari	timeout	
binal.ari	timeout	
count_leaves.ari	yes	4.373
depth.ari	timeout	
difference.ari	yes	0.163
div.ari	timeout	
empty.ari	timeout	
fibonacci.ari	yes	0.553
gcd.ari	timeout	
head.ari	timeout	
inorder.ari	timeout	
insert.ari	yes	0.042
isort.ari	yes	0.193
length.ari	timeout	
max_min.ari	timeout	
minus.ari	timeout	
mirror.ari	yes	22.486
msort.ari	yes	43.564
mul.ari	timeout	
postorder.ari	yes	3.813
preorder.ari	yes	0.808
reverse.ari	yes	0.322
shuffle.ari	yes	2.553
sum.ari	yes	20.524
tail.ari	timeout	
times.ari	yes	0.213
zero.ari	timeout	

ここで、評価で用いた問題を見ていく。

例 5.1 (isort.ari)**.** シグネチャ $\mathcal{F}$ を $\{[\,]^{(0)}, \mathsf{0}^{(0)}, \mathsf{1}^{(0)}, :^{(2)}, \mathsf{insert}^{(2)}, \mathsf{isort}^{(1)}\}$ とする。ただし、関数記号 : は右連結の中置記法である。入力の等式は $\mathsf{isort}(xs) \approx \mathsf{0} : \mathsf{0} : \mathsf{1} : [\,]$ であ

る。等式系 $\mathcal{E}$ は

$$\begin{array}{rr} \mathsf{insert}(x, [\,]) \approx x : [\,] & \mathsf{isort}([\,]) \approx [\,] \\ \mathsf{insert}(0, xs) \approx 0 : xs & \mathsf{isort}(x : xs) \approx \mathsf{insert}(x, \mathsf{isort}(xs)) \\ \mathsf{insert}(1, x : xs) \approx x : 1 : xs & \end{array}$$

であり、この等式系は 0 と 1 を要素に持つリストに対する挿入ソートを表している。本ツールの出力は

$$\{\{xs \mapsto 0 : 0 : 1 : [\,]\}, \{xs \mapsto 1 : 0 : 0 : [\,]\}, \{xs \mapsto 0 : 1 : 0 : [\,]\}\}$$

であり、この集合は $\mathsf{isort}(xs) \approx 0 : 0 : 1 : [\,]$ の $\mathcal{E}$ 単一化子の完全集合である。

例 5.2 (head.ari)**.** シグネチャ $\mathcal{F}$ を $\{[\,]^{(0)}, 0^{(0)}, 1^{(0)}, \mathsf{error}^{(0)}, :^{(2)}, \mathsf{head}^{(1)}\}$ とする。ただし、関数記号 : は右連結の中置記法である。入力の等式は $\mathsf{head}(x : xs) \approx 0$ である。等式系 $\mathcal{E}$ は

$$\mathsf{head}([\,]) \approx \mathsf{error} \qquad \mathsf{head}(0 : xs) \approx 0 \qquad \mathsf{head}(1 : xs) \approx 1$$

であり、この等式系は 0 と 1 を要素に持つリストの先頭の要素を取り出す関数を表している。$\mathsf{head}(x : xs) \approx 0$ の $\mathcal{E}$ 単一化子の完全集合は

$$\{\{x \mapsto 0\}\}$$

であるが、本ツールはこの集合を出力できない。

完全集合を出力できなかった問題 14 問の内の 10 問が等式系を完備化した後に得られた項書換え系が変数消去であった。変数消去項書換え系に対して互換性を持つ木オートマトンを決定性木オートマトンに変換する際に互換性が失われてしまい木オートマトン完備化の手続きが停止しないことがある。次の例は変数消去項書換え系に対して互換性を持つ木オートマトンに木オートマトン完備化を行うと停止しない例である。

例 5.3. 変数消去項書換え系 $\mathcal{R} = \{\mathsf{f}(x) \to \mathsf{a}\}$ を考える。木オートマトン

$$\mathcal{A} = (\{\mathsf{a}^{(0)}, \mathsf{f}^{(1)}\}, \{q_0, q_1\}, \{q_0\}, \{\mathsf{a} \to q_0, \mathsf{f}(q_0) \to q_1, q_1 \to q_0\})$$

は $\mathcal{R}$ に対して互換性を持つ。この木オートマトンに対して木オートマトン完備化を行うと決定性オートマトン

$$\mathcal{A}' = (\{\mathsf{a}^{(0)}, \mathsf{f}^{(1)}\}, \{\{q_0\}, \{q_0, q_1\}\}, \{\{q_0\}\}, \{\mathsf{a} \to \{q_0\}, \mathsf{f}(\{q_0\}) \to \{q_0, q_1\}\})$$

が得られるがこの木オートマトンは $\mathcal{R}$ に対して互換性を持たない。したがって、この木オートマトンに対しては木オートマトン完備化の手続きは停止しない。

5.2 関連研究

ナローイングが実装されているツールとして Maude [13] と Curry [8] が存在する。Maude は結合則と可換則を法としたバリアントナローイング [4] が可能であり、一方 Curry は遅延ナローイングを用いることで無限リストを扱うことが可能である。ベーシックナローイングはベーシックポジションを用いることでナローイングを行う位置を管理していたが、結合則と可換則を法にするナローイングを考えるとその位置が変化してしまう。バリアントナローイングはバリアント [4] を用いることでこの問題を解決している。一方、遅延ナローイングは項を遅延して評価するためベーシックナローイングと比べて効率的に等式単一化子を計算できる。いずれのツールも [14] の定理 3.8 に基づいて完全集合の要素を列挙することができるが、完全集合の要素全てを列挙できたか判定することはできない。そこで、本論文で提案した等式単一化不能性を判定する手法と Maude および Curry で使われる手法を併せることで効率的に完全集合を求めることが可能になると期待できる。

第 6 章

結論

本論文ではナローイングに基づく完全集合の自動計算を行う手法を提案した。ベーシックナローイングの系列を導出規則

$$\frac{U, S \uplus \{(\sigma, s, t, B)\}}{U \cup G(\sigma, s, t), S \cup F(\sigma, s, t, B)} \qquad \frac{U, S \uplus \{(\sigma, s, t, B)\}}{U, S} \text{ if } s \not\rightsquigarrow^{\star} t$$

で表すことで導出列の正規形として完全集合を求めることができる。これ自体はベーシックナローイングの完全性 ([14] の定理 4.5) から容易に得られる事実であるが、二番目の導出規則の条件 $s \not\rightsquigarrow^{\star} t$ を判定することで完全集合の要素が全て列挙されたのか判定することが可能になった。

本論文では木オートマトンを用いて到達不能性解析を行う手法を提案した。木オートマトンで到達不能性を示す際に項の変数に状態を割り当てる必要があり、その状態を割り当てる組み合わせは項の変数の冪乗であるので効率上問題があった。そこで、木オートマトンに型を導入することで項の変数に状態を割り当てる組み合わせを減らし木オートマトンによる到達不能性解析の効率化を行った。

一章で述べた例 1.1、例 1.2 および例 1.3 は本ツールで解けたが、例 5.2 は本ツールでは解けなかった。その理由は、例 5.2 の等式系を完備化して得られた完備な項書換え系は変数消去であり、5 章で考察したように、項書換え系が変数消去であると多くの場合は木オートマトン完備化が停止しないからである。そこで、本論文では木オートマトンを用いて到達不能性解析を行ったが co-rewrite pair など到達不能性解析で用いられている有効な手法を用いることで例 5.2 のような問題を解けるようになり、より多くの等式系の完全集合を計算することが可能になると期待できる。

謝辞　本研究の遂行にあたり、終始熱心にご指導いただきました廣川直准教授に感謝申し上げます。また、研究室での生活を支えてくれた研究室の皆様に感謝申し上げます。特に齊藤哲平さんには研究室での生活を初め、大学生活全般にわたってお世話になりました。ここに感謝申し上げます。

参考文献

[1] F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.

[2] F. Baader and W. Snyder. Unification theory. In J.A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume 1, pages 445–532. Elsevier and MIT Press.

[3] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, C. Löding, S. Tison, and M. Tommasi. Tree automata techniques and applications, 2008.

[4] S. Escobar and J. Meseguer. Variant narrowing and equational unification. *Electronic Notes in Theoretical Computer Science*, 238(3):103–119, 2009. Proc. 7th International Workshop on Rewriting Logic and its Applications.

[5] M. Fay. First-order unification in equational theories. In *Proc. 4th International Workshop on Automated Deduction*, pages 161–167, 1979.

[6] T. Genet. Decidable approximations of sets of descendants and sets of normal forms. In *Proc. 9th International Conference on Rewriting Techniques and Applications*, volume 1379 of *Lecture Notes in Computer Science*, pages 151–165, 1998.

[7] J. Giesl, R. Thiemann, and P. Schneider-Kamp. Proving and disproving termination of higher-order functions. In *Proc. 5th Frontiers of Combining Systems*, volume 3717 of *Lecture Notes in Computer Science*, pages 216–231, 2005.

[8] M. Hanus. Curry: An integrated functional logic language (vers. 0.8.2), 2006.

[9] J.-M. Hullot. Canonical forms and unification. In *Proc. 5th International Conference on Automated Deduction*, volume 87 of *Lecture Notes in Computer Science*, pages 318–334, 1980.

[10] J.-M. Hullot. *Compilation de Formes Canoniques dans les Théories Equa-*

tionelles. Thèse de troisième cycle, Université de Paris Sud, Orsay, 1980.

[11] F. Jacquemard. Decidable approximations of term rewriting systems. In *Proc. 7th International Conference on Rewriting Techniques and Applications*, volume 1103 of *Lecture Notes in Computer Science*, pages 362–376, 1996.

[12] D. Klein and N. Hirokawa. Maximal completion. In *Proc. 22th International Conference on Rewriting Techniques and Applications*, volume 10 of *Leibniz International Proceedings in Informatics*, pages 71–80, 2011.

[13] M.Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. *All About Maude - A High-Performance Logical Framework: How to Specify, Program, and Verify Systems in Rewriting Logic*, volume 4350 of *Lecture Notes in Computer Science*. Springer, 2007.

[14] A. Middeldorp and E. Hamoen. Completeness results for basic narrowing. *Applicable Algebra in Engineering, Communication and Computing*, 5:213–253, 1994.

[15] T. Saito and N. Hirokawa. Weighted path orders are semantic path orders. In *Proc. 14th Frontiers of Combining Systems*, volume 14279 of *Lecture Notes in Computer Science*, pages 63–80, 2023.

[16] C. Sternagel and T. Sternagel. Certifying Confluence of Almost Orthogonal CTRSs via Exact Tree Automata Completion. In *1st International Conference on Formal Structures for Computation and Deduction*, volume 52 of *Leibniz International Proceedings in Informatics*, pages 29:1–29:16, 2016.

[17] T. Sternagel. *Reliable Confluence Analysis of Conditional Term Rewrite Systems*. PhD thesis, The University of Innsbruck, 2017.

[18] J. van de Pol. Modularity in many-sorted term rewriting systems. Technical report, Utrecht University, 1992. Master's thesis.

付録 A

以下は実験で用いた問題に関する入力ファイルである。; はコメントであり、ツールの出力をそこに含めている。

ackermann.ari

```
(format ES)
(fun ack 2)
(fun s 1)
(fun 0 0)
(rule (ack 0 y) (s y))
(rule (ack (s x) 0) (ack x (s 0)))
(rule (ack (s x) (s y)) (ack x (ack (s x) y)))
(goal (ack x y) (s (s (s 0))))

; complete set is:
; {{(y, 0()), (x, ack(0(),ack(0(),0())))},
;  {(y, ack(0(),0())), (x, ack(0(),0()))},
;  {(y, ack(0(),ack(0(),0()))), (x, 0())}}
```

add.ari

```
(format ES)
(fun 0 0)
(fun s 1)
(fun add 2)
(rule (add 0 y) y)
(rule (add (s x) y) (s (add x y)))
(goal (add x y) (s (s 0)))

; complete set is:
; {{(y, s(s(0()))), (x, 0())},
;  {(y, s(0())), (x, s(0()))},
;  {(y, 0()), (x, s(s(0())))}}
```

append.ari

```
(format ES)
(fun append 2)
(fun cons 2)
(fun s 1)
(fun 0 0)
```

```
(fun nil 0)
(rule (append nil z) z)
(rule (append (cons x y) z) (cons x (append y z)))
(goal (append xs ys) (cons (s 0) nil) )

; complete set is:
; {{(ys, cons(s(0()),nil())), (xs, nil())},
;  {(ys, nil()), (xs, cons(s(0()),nil()))}}
```

average.ari

```
(format ES)
(fun 0 0)
(fun s 1)
(fun average 2)
(rule (average 0 0) 0)
(rule (average 0 (s 0)) 0)
(rule (average 0 (s (s 0))) (s 0))
(rule (average(s x) y) (average x (s y)))
(rule (average x (s ( s (s y)))) (s (average (s x) y)))
(goal (average x y) 0)

; timeout
```

binal.ari

```
(format ES)
(fun 0 0)
(fun s 1)
(fun plus 2)
(fun bin 2)
(rule (plus 0 x) x)
(rule (plus (s x) y) (s (plus x y)))
(rule (bin x 0) (s 0))
(rule (bin 0 (s x)) 0)
(rule (bin (s x) (s y)) (plus (bin x (s y)) (bin x y)))
(goal (bin x y) (s 0))

; timeout
```

count_leaves.ari

```
(format ES)
(fun 0 0)
(fun s 1)
(fun leaf 0)
(fun count 1)
(fun add 2)
(fun node 3)
(rule (count leaf) (s 0))
(rule (count (node l 0 r)) (add (count l) (count r)))
(rule (add 0 x) x)
(rule (add (s x) y) (s (add x y)))
(goal (count t) (s (s (s 0))))

; complete set is:
; {{(t, node(node(leaf(),0(),leaf()),0(),leaf()))},
;  {(t, node(leaf(),0(),node(leaf(),0(),leaf())))}}
```

depth.ari

```
(format ES)
(fun s 1)
(fun leaf 0)
(fun max 2)
(fun add 2)
(fun depth 1)
(fun node 3)
(fun 0 0)
(rule (add 0 x) x)
(rule (add (s x) y) (s (add x y)))
(rule (depth leaf) 0)
(rule (depth (node l 0 r)) (add (s 0) (max (depth l) (depth r))))
(rule (max (s x) (s y)) (s (max x y)))
(rule (max 0 x) x)
(rule (max x 0) x)
(goal (depth t) (s (s 0)))

; timeout
```

difference.ari

```
(format ES)
(fun s 1)
(fun d 2)
(fun p 1)
(fun - 1)
(fun 0 0)
(rule (d 0 x) (- x))
(rule (d (s x) (s y)) (d x y))
(rule (d (p x) (p y)) (d x y))
(rule (s (p x)) x)
(rule (p (s x)) x)
(goal (d x y) (s 0))

; complete set is:
; empty set
```

div.ari

```
(format ES)
(fun s 1)
(fun div 2)
(fun sub 2)
(fun 0 0)
(rule (sub 0 y) 0)
(rule (sub x 0) x)
(rule (sub (s x) (s y)) (sub x y))
(rule (div 0 (s y)) 0)
(rule (div (s x) (s y)) (s (div (sub x y) (s y))))
(goal (div x y) (s 0))

; timeout
```

empty.ari

```
(format ES)
(fun isEmpty 1)
(fun cons 2)
```

```
(fun nil 0)
(fun true 0)
(fun false 0)
(fun 0 0)
(rule (isEmpty nil) true)
(rule (isEmpty (cons x xs)) false)
(goal (isEmpty xs) false)

; timeout
```

fibonacci.ari

```
(format ES)
(fun 0 0)
(fun s 1)
(fun fib 1)
(fun add 2)
(rule (add 0 y) y)
(rule (add (s x) y) (s (add x y)))
(rule (fib 0) 0)
(rule (fib (s 0)) (s 0))
(rule (fib (s (s x))) (add (fib (s x)) (fib x)))
(goal (fib x) (s (s (s (s 0)))))

; complete set is:
; empty set
```

gcd.ari

```
(format ES)
(fun leq 2)
(fun s 1)
(fun if 3)
(fun p 1)
(fun gcd 2)
(fun false 0)
(fun true 0)
(fun 0 0)
(fun minus 2)
(rule (leq 0 y) true)
(rule (leq (s x) 0) false)
(rule (leq 0 (s x)) true)
(rule (leq (s x) (s y)) (leq x y))
(rule (if true (s x) (s y)) (gcd (minus x y) (s y)))
(rule (if false (s x) (s y)) (gcd (minus y x) (s x)))
(rule (minus x 0) x)
(rule (minus x (s y)) (p (minus x y)))
(rule (p (s x)) x)
(rule (gcd 0 y) y)
(rule (gcd (s x) 0) (s x))
(rule (gcd (s x) (s y)) (if (leq y x) (s x) (s y)))
(goal (gcd (s x) y) (s (s 0)))

; timeout
```

head.ari

```
(format ES)
(fun head 1)
(fun cons 2)
(fun 0 0)
```

```
(fun 1 0)
(fun nil 0)
(fun error 0)
(rule (head nil) error)
(rule (head (cons 0 xs)) 0)
(rule (head (cons 1 xs)) 1)
(goal (head (cons x xs)) 0)

; timeout
```

inorder.ari

```
(format ES)
(fun leaf 0)
(fun 0 0)
(fun 1 0)
(fun nil 0)
(fun inorder 1)
(fun append 2)
(fun cons 2)
(fun node 3)
(rule (inorder leaf) nil)
(rule (inorder (node l x r)) (append (inorder l) (append x (inorder r))))
(rule (append nil xs) xs)
(rule (append (cons x xs) ys) (cons x (append xs ys)))
(goal (inorder xs) (cons 0 (cons 1 nil)))

; timeout
```

insert.ari

```
(format ES)
(fun 0 0)
(fun 1 0)
(fun nil 0)
(fun cons 2)
(fun insert 2)
(rule (insert x nil) (cons x nil))
(rule (insert 0 ys) (cons 0 ys))
(rule (insert 1 (cons x xs)) (cons x (insert 1 xs)))
(goal (insert x xs) (cons 0 (cons 1 nil)))

; complete set is:
; {{(xs, cons(1(),nil())), (x, 0())},
;  {(xs, cons(0(),nil())), (x, 1())}}
```

isort.ari

```
(format ES)
(fun 0 0)
(fun 1 0)
(fun nil 0)
(fun cons 2)
(fun insert 2)
(fun isort 1)
(rule (insert x nil) (cons x nil))
(rule (insert 0 xs) (cons 0 xs))
(rule (insert 1 (cons x xs)) (cons x (insert 1 xs)))
(rule (isort nil) nil)
(rule (isort (cons x xs)) (insert x (isort xs)))
(goal (isort xs) (cons 0 (cons 0 (cons 1 nil))))
```

```
; complete set is:
;{{(xs, cons(0(),cons(0(),cons(1(),nil())))),
; {(xs, cons(1(),cons(0(),cons(0(),nil())))),
; {(xs, cons(0(),cons(1(),cons(0(),nil()))))}}
```

length.ari

```
(format ES)
(fun length 1)
(fun s 1)
(fun cons 2)
(fun nil 0)
(fun 0 0)
(rule (length nil) 0)
(rule (length (cons x y))  (s (length y)))
(goal (length xs) (s (s 0)))

; timeout
```

max_min.ari

```
(format ES)
(fun max 2)
(fun min 2)
(fun add 2)
(fun f 2)
(fun s 1)
(fun 0 0)
(rule (add 0 y) y)
(rule (add (s x) y) (s (add x y)))
(rule (f 0 y) 0)
(rule (f (s x) 0) (s x))
(rule (f (s x) (s y)) (f x y))
(rule (min x y) (f x (f x y)))
(rule (max x y) (f (add x y) (min x y)))
(goal (max x y) (s 0))

; timeout
```

minus.ari

```
(format ES)
(fun s 1)
(fun minus 2)
(fun 0 0)
(rule (minus x 0) x)
(rule (minus 0 x) 0)
(rule (minus (s x) (s y)) (minus x y))
(goal (minus x 0) (s 0))

; timeout
```

mirror.ari

```
(format ES)
(fun mirror 1)
(fun leaf 0)
(fun node 3)
```

```
(fun 0 0)
(rule (mirror leaf) leaf)
(rule (mirror (node l 0 r)) (node (mirror r) 0 (mirror l)))
(goal (mirror t) (node leaf 0 (node leaf 0 (node leaf 0 (node leaf 0 leaf)))))

; complete set is:
; {{(t, node(node(node(node(leaf(),0(),leaf()),0(),leaf()),0(),leaf()),0(),leaf()))}}
```

msort.ari

```
(format ES)
(fun msort 1)
(fun msort' 3)
(fun split 1)
(fun split' 3)
(fun merge 2)
(fun pair 2)
(fun cons 2)
(fun nil 0)
(fun 0 0)
(fun 1 0)
(rule (msort nil) nil)
(rule (msort (cons x nil)) (cons x nil))
(rule (msort (cons x (cons y ys))) (msort' x y (split ys)))
(rule (msort' x y (pair xs ys)) (merge (msort (cons x xs)) (msort (cons y ys))))
(rule (split nil) (pair nil nil))
(rule (split (cons x nil)) (pair (cons x nil) nil))
(rule (split (cons x (cons y ys))) (split' x y (split ys)))
(rule (split' x y (pair xs ys)) (pair (cons x xs) (cons y ys)))
(rule (merge nil xs) xs)
(rule (merge (cons x xs) nil) (cons x xs))
(rule (merge (cons 0 xs) (cons y ys)) (cons 0 (merge xs (cons y ys))))
(rule (merge (cons 1 xs) (cons y ys)) (cons y (merge (cons 1 xs) ys)))
(goal (msort xs)  (cons 0 (cons 0 (cons 1 nil))))

; complete set is:
; {{(xs, cons(0(),cons(1(),cons(0(),nil()))))},
;  {(xs, cons(0(),cons(0(),cons(1(),nil()))))},
;  {(xs, cons(1(),cons(0(),cons(0(),nil()))))}}
```

mul.ari

```
(format ES)
(fun @ 2)
(fun add 0)
(fun mul 0)
(fun 0 0)
(fun s 0)
(rule (@ (@ add 0) x) x)
(rule (@ (@ add (@ s x)) y) (@ s (@ (@ add x) y)))
(rule (@ (@ mul 0) x) 0)
(rule (@ (@ mul (@ s x)) y) (@ (@ add (@ (@ mul x) y))y))
(goal (@ (@ mul x) y) 0)

; timeout
```

postorder.ari

```
(format ES)
(fun leaf 0)
(fun nil 0)
```

```
(fun post 1)
(fun cons 2)
(fun append 2)
(fun node 3)
(fun 0 0)
(fun 1 0)
(rule (append nil z) z)
(rule (append (cons x y) z) (cons x (append y z)))
(rule (post leaf) nil)
(rule (post (node l x r)) (append (post l) (append (post r) (cons x nil))))
(goal (post xs) (cons 0 (cons 1 nil)))

; complete set is:
; {{(xs, node(leaf(),1(),node(leaf(),0(),leaf()))),
;  {(xs, node(node(leaf(),0(),leaf()),1(),leaf()))}}
```

preorder.ari

```
(format ES)
(fun leaf 0)
(fun nil 0)
(fun pre 1)
(fun cons 2)
(fun append 2)
(fun node 3)
(fun 0 0)
(fun 1 0)
(rule (append nil z) z)
(rule (append (cons x y) z) (cons x (append y z)))
(rule (pre leaf) nil)
(rule (pre (node l x r)) (cons x (append (pre l) (pre r))))
(goal (pre xs) (cons 0 (cons 1 nil)))

; complete set is:
; {{(xs, node(leaf(),0(),node(leaf(),1(),leaf()))),
;  {(xs, node(node(leaf(),1(),leaf()),0(),leaf()))}}
```

reverse.ari

```
(format ES)
(fun append 2)
(fun reverse 1)
(fun cons 2)
(fun nil 0)
(fun 1 0)
(fun 0 0)
(rule (append nil xs) xs)
(rule (append (cons x xs) ys) (cons x (append xs ys)))
(rule (reverse nil) nil)
(rule (reverse (cons x xs)) (append (reverse xs) (cons x nil)))
(goal (reverse xs) (cons 0 (cons 1 (cons 1 nil))))

; complete set is:
; {{(xs, cons(1(),cons(1(),cons(0(),nil())))),}}
```

shuffle.ari

```
(format ES)
(fun append 2)
(fun reverse 1)
(fun shuffle 1)
```

```
(fun s 1)
(fun 0 0)
(fun cons 2)
(fun nil 0)
(rule (append nil z) z)
(rule (append (cons x y) z) (cons x (append y z)))
(rule (reverse nil) nil)
(rule (reverse (cons x y)) (append (reverse y) (cons x nil)))
(rule (shuffle nil) nil)
(rule (shuffle (cons x y)) (cons x (shuffle (reverse y))))
(goal (shuffle xs)  (cons (s (s 0)) (cons 0 (cons (s 0) nil))))

; complete set is:
; {{(xs, cons(s(s(0())),cons(s(0()),cons(0(),nil()))))}}
```

sum.ari

```
(format ES)
(fun 0 0)
(fun s 1)
(fun add 2)
(fun sum 1)
(fun f1 2)
(fun f2 2)
(fun f3 2)
(fun f4 1)
(rule (add 0 y) y)
(rule (add (s x) y) (s (add x y)))
(rule (sum x) (f1 x 0))
(rule (f1 0 y) (f4 y))
(rule (f1 (s x) y) (f2 (s x) y))
(rule (f2 x y) (f3 x (add x y)))
(rule (f3 (s x) y) (f1 x y))
(rule (f4 y) y)
(goal (sum x) (s (s (s (s (s (s (s (s (s (s 0)))))))))))

; complete set is:
; {{(x, s(s(s(s(0()))))))}}
```

tail.ari

```
(format ES)
(fun tail 1)
(fun cons 2)
(fun 0 0)
(fun nil 0)
(rule (tail nil) nil)
(rule (tail (cons x xs)) xs)
(goal (tail xs) (cons 0 nil))

; timeout
```

times.ari

```
(format ES)
(fun s 1)
(fun add 2)
(fun times 2)
(fun 0 0)
(rule (add 0 x) x)
(rule (add (s x) y) (s (add x y)))
```

```
(rule (times 0 x) 0)
(rule (times (s x) y) (add (times x y) y))
(goal (times x y) (s 0))

; complete set is:
; {{(y, s(0())), (x, s(0()))}}
```

zero.ari

```
(format ES)
(fun 0 0)
(fun s 1)
(fun true 0)
(fun false 0)
(fun isZero 1)
(rule (isZero 0) true)
(rule (isZero (s x)) false)
(goal (isZero (s x)) true)

; timeout
```