Institut für Formale Methoden der Informatik

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

Sprachungleichungen über Forest-Sprachen

Marcial Gaißert

Studiengang:	Informatik
Prüfer/in:	Prof. Dr. rer. nat. habil. Volker Diekert
Betreuer/in:	Carlos Camino
Beginn am:	15. Oktober 2019
Beendet am:	8. Juni 2020

Kurzfassung

In dieser Arbeit untersuchen wir die Komplexität des Äquivalenzproblems und des Teilmengenproblems für Forests, jeweils auch unter Substitutionen, auf Eingabe der in [BW07] definierten Forest-Algebren und Forest-Automaten. Hierbei stehen im Zentrum dieser Arbeit Äquivalenz- und Teilmengenprobleme unter Substitutionen an den Blättern der Forests, für die wir Vollständigkeit für verschiedene Klassen in der Polynomialzeithierarchie sowie in einem Fall für DP zeigen. Für Substitutionen an inneren Knoten zeigen wir P-Vollständigkeit bei gegebener Substitution sowie in einem Fall PSPACE-Schwierigkeit für die Frage der Existenz einer (I_{oi}-)Substitution.

Inhaltsverzeichnis

Tabellenverzeichnis

1. Grundlagen

1.1. Komplexitätstheoretische Grundlagen

Neben den üblichen Definitionen für NP, P, coNP und PSPACE werden wir in dieser Arbeit die Definitionen der Klassen $\Sigma_i^{\mathsf{P}} = \mathsf{NP}^{\Sigma_{i-1}^P}$ mit $\Sigma_0^{\mathsf{P}} = \mathsf{P}$ in der Polynomialzeithierarchie, von $\mathsf{DP} = \mathsf{NP} \wedge \mathsf{coNP} = \{L \cap R \mid L \in \mathsf{NP} \wedge R \in \mathsf{coNP}\}$, sowie die Schreibweise A^B für die Klasse A relativ zu einem Orakel für eine Sprache oder Komplexitätsklasse B verwenden. Für eine genaue Behandlung der genannten Themen sei auf die einführende Literatur zur Komplexitätstheorie verwiesen.

1.2. Forests und Kontexte

Angelehnt an [BW07] definieren wir:

Definition 1.2.1 (Forests und Bäume)**.** *Wir definieren die Menge $\mathcal{F}_B(A)$ der Forests sowie die Menge $\mathcal{T}_B(A)$ der Bäume über A mit Blättern aus B induktiv wie folgt:*

- *Es ist $\mathcal{T}_B(A) \subset \mathcal{F}_B(A)$.*
- *$0 \in \mathcal{F}_B(A)$, wobei 0 für den leeren Forest steht.*
- *Sei $b \in B$. Dann ist $b \in \mathcal{T}_B(A)$.*
- *Sind $f_1, \ldots, f_k \in \mathcal{T}_B(A)$, dann ist $f_1 + f_2 + \cdots + f_k \in \mathcal{F}_B(A)$.*
- *Ist $f \in \mathcal{F}_B(A) \setminus \{0\}$ ein Forest und $a \in A$, dann ist $a \circ f \in \mathcal{T}_B(A)$.*

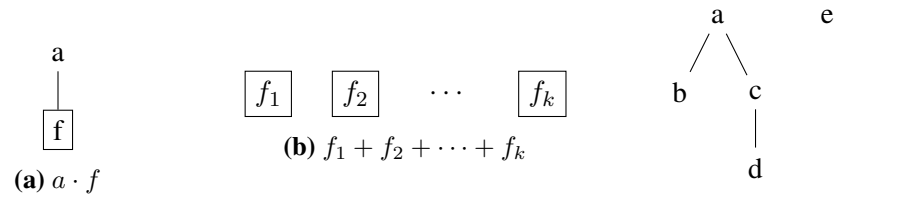
Wir schreiben auch $\mathcal{F}(A) = \mathcal{F}_A(A)$. Es ist $\mathcal{F}_B(A) \subseteq \mathcal{F}(A \cup B)$.

Für beliebige Forests $f = (f_1 + f_2 + \cdots + f_k)$ und $g = (g_1 + g_2 + \cdots + g_l)$ mit $\forall i : f_i, g_i \in \mathcal{T}_B(A)$ definieren wir $f + g = f_1 + f_2 + \cdots + f_k + g_1 + g_2 + \cdots + g_l$.

Forests können auch wie in Abbildung 1.1 grafisch dargestellt werden. Hierbei stehen die umrandeten Knoten jeweils für beliebige Forests.

Definition 1.2.2 (Forest-Sprache)**.** *Eine* Forest-Sprache *über A ist eine Teilmenge $L \subseteq \mathcal{F}(A)$.*

Definition 1.2.3 (Kontext)**.** *Ein Kontext über A ist ein Forest aus $\mathcal{F}_{A \cup \{1\}}(A)$, in dem die 1 nur einmal vorkommt. Die Menge der Kontexte über A bezeichnen wir mit $\mathcal{C}(A)$.*



(a) $a \cdot f$ **(b)** $f_1 + f_2 + \cdots + f_k$ **(c)** Der Forest $a \cdot (b + cd) + e$

Abbildung 1.1.: Grafische Darstellung von Forests

Die 1 in einem Kontext c ist als Platzhalter für einen Forest zu betrachten. Dieser kann durch Einsetzen gefüllt werden:

Definition 1.2.4 (Einsetzen in einen Kontext)**.** *Sei $c \in \mathcal{C}(A)$ ein Kontext und $f \in \mathcal{F}_B(A)$ ein Forest. Dann definieren wir induktiv über alle Forests c:*

Fall 1: *$c = \varepsilon$. Dann ist $c \cdot f = \varepsilon$.*

Fall 2: *$c = a \in A$. Dann ist $c \cdot f = a$.*

Fall 3: *$c = 1$. Dann ist $c \cdot f = f$.*

Fall 4: *$c = g_1 + g_2$ für zwei Forests g_1 und g_2. Dann ist $c \cdot f = g_1 \cdot f + g_2 \cdot f$.*

Fall 5: *$c = a \circ g$ für einen Forest g. Dann ist $c \cdot f = a \circ (g \cdot f)$.*

Da ein Kontext ein spezieller Forest ist, erhalten wir damit auch eine Operation über Kontexten. Diese ist assoziativ und 1 ist ihr neutrales Element.

Der einzige Kontext, der auf der linken Seite von $\circ$ stehen kann, ist 1, und $1 \circ f$ ist nur dann wieder ein Kontext, wenn $f = 0$. In diesem Fall können wir also einfach 1 schreiben. Daher werden wir im folgenden sowohl für $\circ$ als auch für $\cdot$ einfach $\cdot$ schreiben oder den Operator wie bei der Multiplikation weglassen.

Definition 1.2.5. *Sei $f \in \mathcal{F}(A)$. Dann sei:*

- $\mathcal{SF}(f) = \{g \in \mathcal{F}(A) \mid \exists c \in \mathcal{C}(A) : cg = f\}$ *die Menge der* Unterforests
- $\mathcal{SC}(f) = \{c \in \mathcal{C}(A) \mid \exists d \in \mathcal{C}(A), g \in \mathcal{F}(A) : dcg = f\}$ *die Menge der* Unterkontexte

von f. Für Forest-Sprachen $L \subseteq \mathcal{F}(A)$ sei $\mathcal{SF}(L) = \bigcup_{f \in L} \mathcal{SF}(f)$ und $\mathcal{SC}(L) = \bigcup_{f \in L} \mathcal{SC}(f)$.

1.3. Forest-Algebren

Wir definieren, ebenfalls in Anlehnung an [BW07]:

Definition 1.3.1 (Forest-Algebra)**.** *Eine Forest-Algebra ist ein 5-Tupel $(H, V, \text{act}, \text{in}_L, \text{in}_R)$, wobei*

- $(H, +, 0)$ *und* $(V, \cdot, 1)$ *Monoide,*
- $\text{act} : H \times V \to H$ *eine Operation von V auf H und*

- $\text{in}_L, \text{in}_R : H \to V$ *Abbildungen*

sind sowie gilt:

Einfügen $\text{in}_L(g)h = g + h$, $\text{in}_R(g)h = h + g$

Treue $\forall v \neq w \exists h \in H\colon vh \neq wh$

Operation $\forall u, v \in V \forall h \in H\colon \text{act}(\text{act}(h, u), v) = \text{act}(h, uv)$

Hierbei schreiben wir $vh = \text{act}(h, v)$ *sowie* $1 + g = \text{in}_R(g), g + 1 = \text{in}_L(g)$ *für* $v \in V, h \in H$*. Zur Vereinfachung der Notation schreiben wir auch nur* (H, V) *für die Forest-Algebra.*

Wir nennen H *das horizontale Monoid und* V *das vertikale Monoid.*

Ein Homomorphismus über Forest-Algebren muss nun sowohl die Elemente des horizontalen als auch des vertikalen Monoids abbilden:

Definition 1.3.2. *Ein Forest-Algebra-Homomorphismus von* (H, V) *auf* (G, W) *ist ein Paar* α, β *wobei*

- $\alpha : H \to G$*,* $\beta : V \to W$ *Monoid-Homomorphismen sind und*
- $\alpha(vh) = \beta(v)\alpha(h)$ *und*
- $\beta\left(\text{in}_L(h)\right) = \text{in}_L\left(\alpha(h)\right)$*,* $\beta\left(\text{in}_R(h)\right) = \text{in}_R\left(\alpha(h)\right)$

für alle $v \in V$ *und* $h \in H$ *gelten.*

Die Freie Forest-Algebra

Definition 1.3.3. *Die Freie Forest-Algebra* A^Δ *über* A *ist die Forest-Algebra* $(\mathcal{F}(A), \mathcal{C}(A), \text{act}, \text{in}_L, \text{in}_R)$ *mit*

- $\text{act}(h, v) = v \cdot h$ *ist das Einsetzen des Forests in den Kontext*
- $\text{in}_L(h) = 1 + h,\ \text{in}_R(h) = h + 1$ *mit* $+$ *als horizontale Operation auf Forests.*

1.3.1. Erkennung durch Forest-Algebren

Definition 1.3.4. *Eine Forest-Sprache* $L \subseteq \mathcal{F}_B(A)$ *wird* erkannt *von einer Forest-Algebra* (H, V) *mit einem Homomorphismus* $(\alpha, \beta) : (A \cup B)^\Delta \to (H, V)$ *und einer Zielmenge* $G \subseteq H$*, wenn*

$$\forall t \in \mathcal{F}_B(A) : \alpha(t) \in G \iff t \in L$$

Definition 1.3.5. *Eine Forest-Sprache* $L \subseteq \mathcal{F}_B(A)$ *nennen wir* erkennbar *genau dann, wenn sie von einer* endlichen *Forest-Algebra* (H, V) *erkannt wird. Die Menge der erkennbaren Forest-Sprachen über einem Alphabet* A *nennen wir* $\mathcal{F}\mathsf{Rec}_A$*.*

Anmerkungen zum Erkennbarkeitsbegriff

Wie von [DS10, S. 3, Remark 3] bemerkt, ist die Forderung der Treue von Forest-Algebren für die Klasse der erkennbaren Sprachen unerheblich. In [BW07] wird für den erkennenden Homomorphismus im Gegensatz zu hier und zu [BWS12] Surjektivität gefordert. Diese stellt, wie [BWS12] anmerken, eine gewisse Minimalitätsanforderung für den erkennenden Homomorphismus dar, ändert die Klasse der erkennbaren Forest-Sprachen jedoch ebenfalls nicht.

1.4. Automaten über Forests

Wie [BW07] definieren wir:

Definition 1.4.1. *Ein Forest-Automat M über A ist ein Tupel $M = ((Q, +, 0), A, \delta, F)$ wobei*

- $(Q, +, 0)$ *ein endliches Monoid,*
- $\delta : A \times Q \to Q$ *die Übergangsfunktion und*
- $F \subseteq Q$ *die Menge der akzeptierenden Zustände sind.*

Definition 1.4.2. *Sei M ein Forest-Automat und $f \in \mathcal{F}(A)$ ein Forest. Dann definiere f^M induktiv wie folgt:*

- $0^M = 0$
- $(as)^M = \delta(a, s^M)$ *für* $a \in A, s \in \mathcal{F}(A)$
- $(f_1 + f_2 + \cdots + f_k)^M = f_1^M + f_2^M + \cdots + f_k^M$

Ein Forest f wird von M akzeptiert, wenn $f^M \in F$.

Definition 1.4.3. *Die von einem Forest-Automaten M erkannte Sprache ist*

$$L(M) := \left\{ f \in \mathcal{F}(A) \mid f^M \in F \right\}$$

Bojańczyk und Walukiewicz zeigen in [BW07], dass eine Forest-Sprache von einer endlichen Forest-Algebra erkannt wird genau dann, wenn sie von einem Forest-Automaten erkannt wird. Die erkennende Forest-Algebra zu einem gegebenen Forest-Automaten ist hierbei (Q, Q^Q).

Im Allgemeinen gibt es auch keine kleinere Forest-Algebra zu einem Forest-Automaten. Betrachte hierzu den Forest-Automaten $(\mathbb{Z}/n\mathbb{Z}, \{a, b\}, \delta, F)$ mit

$$\delta\colon\ (a, q) \mapsto \begin{cases} 1 & q = 0 \\ 0 & q = 1 \\ q & \text{sonst} \end{cases}$$

$$(b, q) \mapsto \begin{cases} 0 & q \in \{0, 1\} \\ q & \text{sonst} \end{cases}$$

Es für jede Abbildung $\mathbb{Z}/n\mathbb{Z} \to \mathbb{Z}/n\mathbb{Z}$ einen zugehörigen Kontext über $\mathbb{Z}/n\mathbb{Z}$, also muss das vertikale Monoid der äquivalenten Forest-Algebra mindestens $|Q|^{|Q|}$ Elemente haben.

Neben den von [BW07] definierten Forest-Automaten ist auch eine Erweiterung der von [Lib05] definierten NUTAs auf Forests möglich. Da bei diesen im horizontalen üblicherweise nichtdeterministische endliche Automaten oder reguläre Ausdrücke verwendet werden, ist hier schon der äquivalente Forest-Automat exponentiell groß.

2. Konstruktionen auf Forest-Algebren

2.1. Produkt-Forest-Algebren und -Automaten

Sowohl für Forest-Algebren als auch für Forest-Automaten lässt sich mittels einer üblichen Produkt-Konstruktion die zugehörige Produkt-Forest-Algebra bzw. der zugehörige Produkt-Forest-Automat konstruieren.

Eine Forest-Algebra bzw. einen Forest-Automat für das Komplement erhält man durch Komplementierung der erkennenden Menge bzw. der Endzustandsmenge.

Beide Konstruktionen sind in logarithmischem Platz möglich.

2.2. Endliche Forest-Sprachen

Schränken wir die freie Forest-Algebra auf diejenigen Elemente ein, die tatsächlich in einem Forest oder einer Forest-Sprache vorkommen, und fassen die restlichen Elemente zu $\perp$ zusammen, erhalten wir:

Lemma 1. Sei $L \subseteq \mathcal{F}(A)$. Dann wird L erkannt von der Forest-Algebra

$$(\mathcal{SF}(L) \dot\cup \{\perp\}, \mathcal{SC}(L) \dot\cup \{\perp\}).$$

Beweis. Die Forest-Algebra verhalte sich wie A^Δ, wobei alle Forests, die nicht in $\mathcal{SF}(L)$ sind, und alle Kontexte, die nicht in $\mathcal{SC}(L)$ sind, mit $\perp$ identifiziert werden. Die erkennende Menge ist dann einfach L selbst und der erkennende Homomorphismus bildet alle Forests und Kontexte auf sich selbst oder $\perp$ ab. □

Diese Forest-Algebra für endliche Forest-Sprachen ist in logarithmischem Platz konstruierbar: Jeder Forest ist durch die Knotenfolge auf der obersten Ebene (die durch Anfang und Ende repräsentiert werden kann) eindeutig definiert, nachdem diese durch Vergleich mit den anderen vorkommenden Folgen zusammengefasst wurden. Jeder Kontext ist hier durch zwei Forests eindeutig repräsentiert, wobei ebenfalls noch ein Zusammenfassen derselben nötig ist.

2.3. Forest-Sprachen für Aussagenlogik

Wir werden mehrfach die Darstellung von aussagenlogischen Formeln als Forest benötigen. Eine Konstruktion für eine ähnliche Sprache ohne Negation findet sich in [BWS12].

Definition 2.3.1. *Sei F eine aussagenlogische Formel mit atomaren Formeln $A_1, \ldots, A_n$. Dann definiere $\langle F \rangle_{\mathcal{F}} \subseteq \mathcal{F}(\{\wedge, \vee, \neg, A_1, \ldots, A_n\})$ induktiv wie folgt:*

- $\langle A_i \rangle_{\mathcal{F}} = A_i$
- $\langle F \wedge G \rangle_{\mathcal{F}} = \wedge \cdot (\langle F \rangle_{\mathcal{F}} + \langle G \rangle_{\mathcal{F}})$
- $\langle F \vee G \rangle_{\mathcal{F}} = \vee \cdot (\langle F \rangle_{\mathcal{F}} + \langle G \rangle_{\mathcal{F}})$
- $\langle \neg F \rangle_{\mathcal{F}} = \neg \cdot \langle F \rangle_{\mathcal{F}}$

Definition 2.3.2. *Seien F, $A_1, \ldots, A_n$ wie in Definition 2.3.1. Sei weiterhin $\mathcal{A}$ eine für F passende Belegung. Dann definiere $\langle F \rangle_{\mathcal{F}}^{\mathcal{A}}$ wie $\langle F \rangle_{\mathcal{F}}$, mit der Anpassung, dass*

$$\langle A_i \rangle_{\mathcal{F}}^{\mathcal{A}} = \begin{cases} \wedge & A_i^{\mathcal{A}} = 1 \\ \vee & A_i^{\mathcal{A}} = 0 \end{cases}$$

Lemma 2. Die Sprache

$$\mathsf{True}_{\mathcal{F}} = \left\{ \langle F \rangle_{\mathcal{F}}^{\mathcal{A}} \mid \mathcal{A} \models F, F \text{ aussagenlogische Formel, } \mathcal{A} \text{ Belegung} \right\}$$

ist erkennbar.

Beweis. Sei $H = \mathbb{B}^{\leq 2} \cup \{\bot\}$ und $V = H^H$ mit[1]:

$$\begin{aligned} \varphi_R : \{\wedge, \vee, \neg\}^{\Delta} &\to (H, V) \\ 0 &\mapsto \varepsilon \\ \vee &\mapsto 0 \\ \wedge &\mapsto 1 \\ \vee \cdot 1 &\mapsto (H \to H, x_1 \ldots x_k \mapsto x_1 \vee \cdots \vee x_k) && k \in \{0, 1, 2\} \\ \wedge \cdot 1 &\mapsto (H \to H, x_1 \ldots x_k \mapsto x_1 \wedge \cdots \wedge x_k) && k \in \{0, 1, 2\} \\ \neg \cdot 1 &\mapsto (H \to H, x \mapsto 1 - x, x_1 \ldots x_k \mapsto \bot) && k \neq 1 \end{aligned}$$

Es ist zu beachten, dass hier als Konstanten 0 und 1 leere Und- bzw. Oder-Verknüpfungen verwendet werden. $\bot$ ist der Wert, der nicht wohlgeformten Eingaben zugewiesen wird.

Der Vertikale Operator "$\cdot$" ist dann einfach Funktionsanwendung (bzw. Komposition) und der horizontale gegeben durch $w + v = wv$, wenn $|wv| \leq 2$ und $w, v \neq \bot$ und $w + v = \bot$ sonst. Dann wird von (H, V) mit φ die oben angegebene Sprache erkannt. □

[1] Hierbei sei $\mathbb{B}^{\leq 2} = \{\varepsilon\} \cup \mathbb{B} \cup \mathbb{B}^2$

3. Allgemeine Probleme auf Forest-Sprachen

Wenn im Folgenden eine erkennbare Forest-Sprache L über einem Alphabet A als Eingabe angegeben ist, so gehen wir — solange nicht anders angegeben — davon aus, dass diese kodiert ist

beim Beweis von Schwierigkeitsresultaten als endliche Forest-Algebra $((H_L, +_L, 0_L), (V_L, \cdot_L, 1_L))$ mit einem Homomorphismus $\varphi_L : A^\Delta \to (H_L, V_L)$ und einer Menge $E_L \subseteq H_L$, sodass $\varphi_L(L) = E_L$.

bei der Angabe von Algorithmen als endlicher Forest-Automat $M_L = ((H_L, +_L, 0_L), \delta_L, E_L)$. Die Bezeichner wurden hier denen der Forest-Algebren angeglichen.

Da jedes der Probleme mit einer Forest-Algebra als Eingabe mittels

$$\begin{aligned} \delta : A \times Q_L &\to Q_L \\ a, q &\mapsto \varphi_L(a) \cdot_L q \end{aligned}$$

auf das selbe Problem mit einem Forest-Automaten als Eingabe reduziert werden kann, können die Resultate so für beide möglichen Kodierungen Anwendung finden. Wo immer es aus dem Kontext ersichtlich ist werden wir die Indizes der binären Operatoren weglassen.

Wir werden drei Probleme ohne Substitutionen auf Forest-Sprachen betrachten:

Definition 3.0.1 (Leerheitsproblem für Forest-Sprachen)**.** *Das Problem* $\mathcal{F}$Empty *sei:*

Eingabe: *Eine erkennbare Forest-Sprache L.*

Frage: *Ist $L = \emptyset$?*

Definition 3.0.2 (Teilmengenproblem für Forest-Sprachen)**.** *Das Problem* $\mathcal{F}$Subset *sei:*

Eingabe: *Zwei erkennbare Forest-Sprachen L und R.*

Frage: *Ist $L \subseteq R$?*

Definition 3.0.3 (Äquivalenzproblem)**.** *Das Problem* $\mathcal{F}$Equiv *sei:*

Eingabe: *Zwei erkennbare Forest-Sprachen L und R.*

Frage: *Ist $L = R$?*

Für diese Probleme zeigen wir:

Satz 1. $\mathcal{F}$Empty, $\mathcal{F}$Subset und $\mathcal{F}$Equiv sind P-vollständig.

Beweis. Mittels des effektiven Abschlusses unter booleschen Operationen (Abschnitt 2.1) erhalten wir:

$\mathcal{F}$Empty $\leq_{\log}$ $\mathcal{F}$Subset**.** Es ist $L = \emptyset \iff L \subseteq \emptyset$.

$\mathcal{F}$Subset $\leq_{\log}$ $\mathcal{F}$Equiv**.** Es ist $L \subseteq R \iff L \cup R = R$.

$\mathcal{F}$Equiv $\leq_{\log}$ $\mathcal{F}$Empty**.** Es ist $L = R \iff L \triangle R = \emptyset$.

Damit bleibt noch zu zeigen:

$\mathcal{F}$Subset **ist** P**-schwierig.** Wie von [Lad75] gezeigt, ist das Problem CVP der Auswertung eines Schaltkreises P-vollständig. Wir wollen nun CVP $\leq_{\log}$ $\mathcal{F}$Subset zeigen.

Der gegebene Schaltkreis kann als gerichteter kreisfreier Graph G mit Knotenbeschriftungen aus $\{\wedge, \vee, \neg\}$ aufgefasst werden. Die gegebene Belegung sei hier bereits eingesetzt und die Konstanten 0 und 1 seien als leere $\vee$- bzw. $\wedge$-Verknüpfung dargestellt. Ohne Einschränkung gehen wir davon aus, dass die Knoten von G zusätzlich durchnummeriert sind und maximalen Eingangsgrad 2 haben.

Durch Vervielfachen der Knoten mit einem Ausgangsgrad > 1 erhält man hieraus einen Forest $f = \langle F \rangle_{\mathcal{F}}^{\mathcal{A}}$, mit einer aussagenlogischen Formel F und einer Belegung $\mathcal{A}$, wobei $\mathcal{A}(F)$ dem Ergebnis entspricht, das man durch Auswerten des Schaltkreises erhält.

Betrachte nun die in Lemma 1 definierte Forest-Algebra. Sowohl die Menge der Unterforests als auch der Unterkontexte von f können lediglich polynomiell groß in der Knotenzahl von G sein. Durch die Nummerierung der Knoten ist ein Vergleich derselben, um diese jeweils nur einmal zu erhalten, hinfällig: Es entsprechen sich genau diejenigen Unterforests bzw. Unterkontexte, die durch die gleichen Knoten in G definiert sind. Die Forest-Algebra ist damit aus G in logarithmischem Platz konstruierbar.

In Lemma 2 wurde gezeigt, dass die Sprache $\mathsf{True}_{\mathcal{F}}$ der zu wahr auswertenden booleschen Ausdrücke erkennbar ist.

Nun ist $\{f\} \subseteq \mathsf{True}_{\mathcal{F}}$ genau dann, wenn der gegebene Schaltkreis zu 1 auswertet.

$\mathcal{F}$Empty $\in$ P**.** Sei ein Forest-Automat $M_L = ((Q, 0, +), A, \delta, F)$ gegeben.

Berechne die Menge

$$\text{reachable}\,(M_L) = \left\{ f^{M_L} \mid f \in \mathcal{F}\,(A) \right\}$$

mithilfe eines Markierungsalgorithmus: Markiere zu Beginn lediglich die $0 \in Q$ und anschließend, bis sich keine neuen Markierungen mehr ergeben, $q + q'$ und $\delta(a, q)$ für $a \in A$ und bereits markierte $q, q' \in Q$.

Sicherlich werden wir so lediglich solche $q \in Q$ markieren, für die ein $f \in \mathcal{F}\,(A)$ existiert mit $f^{M_L} = q$; ein entsprechender Zeuge f für jedes q ließe sich bei jeder Markierung zusätzlich konstuieren. Auch kann jeder Forest mittels der beiden Operationen konstruiert werden, sodass auch alle erreichbaren Zustände markiert sind.

Schließlich ist nun zu prüfen, ob $\text{reachable}\,(M_L) \cap F = \emptyset$.

□

4. Substitutionen über Forests und Forest-Sprachen

4.1. Substitutionsbegriffe für Forests allgemein

Um die Eigenschaften, die allen von uns definierten Substitutionsbegriffen gemeinsam sind, allgemein behandeln zu können, formulieren wir zunächst einige allgemeine Forderungen an einen Substitutionsbegriff:

Definition 4.1.1 (Substitutionsbegriff)**.** *Ein Substitutionsbegriff für Forest-Sprachen S definiert...*

- *... eine Menge $D_S(A, C)$ in Abhängigkeit zweier Alphabete A und C. Dies ist die Menge der möglichen rechten Seiten einer Substitution, also der Objekte, durch die ein einzelnes $x \in X$ ersetzt wird. Im Fall $A = C$ schreiben wir auch kurz $D_S(A)$.*
- *... für jedes $\sigma : X \to 2^{D_S(A,C)}$ mit $A \cap X = \emptyset$ eine Erweiterung zu $\sigma_S : \mathcal{F}(X \cup A) \to 2^{\mathcal{F}(C \cup A)}$ mit $\sigma_S(0) = 0$.*

  Damit sei dann

$$\begin{aligned} \sigma_S : 2^{\mathcal{F}(X \cup A)} &\to 2^{\mathcal{F}(C \cup A)} \\ L &\mapsto \bigcup_{f \in L} \sigma_S(L) \end{aligned}$$

  und für $\sigma, \sigma' : X \to 2^{D_S(A,C)}$ und $L \subseteq \mathcal{F}(X \cup A)$ gelte

$$\big(\forall x \in X : \sigma(x) \subseteq \sigma'(x)\big) \implies \sigma_S(L) \subseteq \sigma'_S(L)$$

Um nur lokal verwendete Substitutionen knapp notieren zu können, schreibe

$$\begin{aligned} [x \leftarrow D] : \{x\} &\to 2^{D_S(A,C)} \\ x &\mapsto D. \end{aligned}$$

Die so definierten Funktionen und ihre Erweiterungen verwenden wir in Postfixschreibweise.

Definition 4.1.2 (Relationen über Substitutionen)**.** *Sei S ein Substitutionsbegriff für Forests sowie $\sigma : X \to D_S(A, C)$ eine Substitution. Dann schreibe:*

$$\begin{aligned} \sigma' \subseteq \sigma &\iff \forall x \in X : \sigma'(x) \subseteq \sigma(x) && \text{für } \sigma' \colon X \to 2^{D_S(A,C)} \\ \theta \in \sigma &\iff \forall x \in X : \theta(x) \in \sigma(x) && \text{für } \theta \colon X \to D_S(A, C) \end{aligned}$$

Definition 4.1.3 (Saturierungsbegriff)**.** *Sei S ein Substitutionsbegriff für Forests, R eine Forest-Sprache und $\sigma : X \to 2^{D_S(A,C)}$. Dann sei eine saturierte Substitution $\hat{\sigma}^{S,R}$ eine Erweiterung von σ so, dass: $\sigma \subseteq \hat{\sigma}^{S,R}$ und $\forall f \in \mathcal{F}(X \cup A) : \sigma_S(f) \subseteq R \iff \hat{\sigma}^{S,R}_S(f) \subseteq R$.*

4.2. Probleme mit Substitutionen über Forest-Sprachen

4.2.1. Erreichbarkeit von Zutandskombinationen

Wir definieren für einen Substitutionsbegriff S:

Definition 4.2.1 (Erreichbarkeit von Zustandskombinationen)**.** *Das Problem* $\mathcal{F}\mathsf{Reachable}_S$ *sei:*

Eingabe: *Zwei erkennbare Forest-Sprachen* L *und* R*, eine Substitution* σ*, sowie Teilmengen* $M_1, \ldots, M_k \subseteq H_L \times H_R$*.*

Frage: $\forall 1 \leq i \leq k : M_i \cap \text{reachable}_S(\sigma, L, R) \neq \emptyset$*?*

Dabei sei $\text{reachable}_S(\sigma, L, R) = \left\{(f^{M_L}, f'^{M_R}) \mid f \in \mathcal{F}(A \cup X), f' \in \sigma_S(f)\right\}$.

4.2.2. Teilmengen- und Äquivalenzproblem mit Substitution

Bezüglich einem Substitutionsbegriff S definieren wir die folgenden Probleme mit gegebenen Substitutionen:

Definition 4.2.2 (Teilmengenproblem mit Substitution)**.** *Das Problem* $\mathcal{F}\mathsf{Subset}_S$ *sei:*

Eingabe: *Zwei erkennbare Forest-Sprachen* L *und* R *sowie eine Substitution* σ*.*

Frage: *Ist* $\sigma_S(L) \subseteq R$*?*

Definition 4.2.3 (Umgekehrtes Teilmengenproblem mit Substitution)**.** *Das Problem* $\mathcal{F}\mathsf{Subset}'_S$ *sei:*

Eingabe: *Zwei erkennbare Forest-Sprachen* L *und* R *sowie eine Substitution* σ*.*

Frage: *Ist* $\sigma_S(L) \supseteq R$*?*

Definition 4.2.4 (Äquivalenzproblem mit Substitution)**.** *Das Problem* $\mathcal{F}\mathsf{Equiv}_S$ *sei:*

Eingabe: *Zwei erkennbare Forest-Sprachen* L *und* R *sowie eine Substitution* σ*.*

Frage: *Ist* $\sigma_S(L) = R$*?*

Satz 2. $\mathcal{F}\mathsf{Subset}_S \leq_{\log} \mathcal{F}\mathsf{Equiv}_S$

Beweis. Seien σ, L und R eine $\mathcal{F}\mathsf{Subset}_S$-Instanz. Ohne Einschränkung sei $\sigma_S(R) = R$ (durch Umbenennen aller $x \in X$ in R). Dann betrachte $\mathcal{F}\mathsf{Equiv}_S$ auf Eingabe von σ, $L \cup R$ und R.

Es gilt:

$$\sigma_S(L) \subseteq R \iff \sigma_S(L) \cup R = R \iff \sigma_S(L) \cup \sigma_S(R) = R \iff \sigma_S(L \cup R) = R.$$

Zur Berechnung der Vereinigung zweier Forest-Sprachen berechne die Produkt-Forest-Algebra bzw. den Produkt-Forest-Automaten. □

Lemma 3. Sei ein System von Sprachungleichungen mit $\sim \in \{=, \subseteq\}$, über erkennbaren Sprachen L_i und R_i:

$$\begin{aligned} \sigma_S(L_1) &\sim R_1 \\ &\vdots \\ \sigma_S(L_k) &\sim R_k \end{aligned}$$

Dann existieren erkennbare Forest-Sprachen L und R, für die $\sigma_S(L) \sim R$ genau dann gilt, wenn σ jede der obigen Sprachungleichungen erfüllt.

Wenn die L_i bzw. R_i jeweils von Forest-Algebren der Größe[1] l_i bzw. r_i erkannt werden, so wird L von einer Forest-Algebra einer Größe in $\mathcal{O}(l_1 \cdot l_2 \cdots l_k)$ bzw. R von einer Forest-Algebra einer Größe in $\mathcal{O}(r_2 \cdot r_2 \cdots r_k)$ erkannt.

Beweis. Seien die L_i und R_i gegeben als Forest-Algebren und $m_1, \ldots, m_k$ neue Alphabetzeichen. Wähle dann:

$$L = \bigcup_{i=1}^{k} m_i \cdot L_i \qquad\qquad R = \bigcup_{i=1}^{k} m_i \cdot R_i$$

Es ist nun:

$$\begin{aligned} \sigma_S(L) \sim R &\iff \bigcup_{i=1}^{k} m_i \cdot \sigma_S(L_i) \sim \bigcup_{i=1}^{k} m_i \cdot R_i \\ &\iff \forall 1 \leq i \leq k : \sigma_S(L_i) \sim R_i \end{aligned}$$

Die letzte Äquivalenz gilt, da hier nur Bäume mit Wurzeln aus $\{m_1, \ldots, m_k\}$ vorkommen und diese nur gleich sein können, wenn sie die gleiche Wurzel haben. □

Satz 3. $\mathcal{F}\mathsf{Subset}_S \leq_{\log} \overline{\mathcal{F}\mathsf{Reachable}_S}$

Beweis. Betrachte $\mathcal{F}\mathsf{Reachable}_S$ auf den selben σ, L und R mit $M_1 = E_L \times \overline{E_R}$. Dann gibt es ein $m \in M \cap \text{reachable}_S(\sigma, L, R)$ genau dann, wenn es Forests $f_l \in L$ und $f_r \in \sigma(f_l)$ gibt mit $f_r \notin R$. □

Satz 4. $\mathcal{F}\mathsf{Subset}'_S \leq_{\log} \mathcal{F}\mathsf{Reachable}_S$

Beweis. Betrachte $\mathcal{F}\mathsf{Reachable}_S$ auf den selben σ, L und R. Sei nun $E_R = \{r_1, \ldots, r_k\}$. Dann wähle als Teilmengen $M_i = M_{r_i} = \{(l, r_i) \mid l \in E_L\}$.

Sei nun $\sigma_S(L) \not\supseteq R$. Dann gibt es ein $f_r \in R$, sodass $\forall f_l \in L : f_r \notin \sigma_S(f_l)$, also $M_r \cap \text{reachable}_S(\sigma, L, R) = \emptyset$.

Sei andererseits ein $M_i \cap \text{reachable}_S(\sigma, L, R) = \emptyset$. Dann ist $r_i \in E_R$ und $(l, r_i) \notin \text{reachable}_S(\sigma, L, R)$ für alle $l \in E_L$. Betrachte nun einen Forest $f_r \in R$ mit $f_r^{M_R} = r_i$. Für diesen kann es kein $f_l \in L$ geben mit $f_r \in \sigma_S(f_l)$. □

[1] Die Größe kann hier sowohl als Größe der horizontalen oder der vertikalen Forest-Algebra, oder auch als Summe der beiden verstanden werden.

Satz 5. $\mathcal{F}\mathsf{Equiv}_S \leq_{\log} \mathcal{F}\mathsf{Reachable}_S \wedge \overline{\mathcal{F}\mathsf{Reachable}_S}$

Beweis. Es ist $\sigma_S(L) = R \iff \sigma_S(L) \subseteq R \wedge \sigma_S(L) \supseteq R$. □

Mit diesen Problemen werden wir uns später im Kontext konkreter Substitutionsdefinitionen beschäftigen.

4.2.3. Existenz einer Substitution

Ebenfalls bezüglich einem Substitutionsbegriff S definieren wir die folgenden Probleme, die nach der Existenz einer Substitution fragen:

Definition 4.2.5 (Existenz einer Substitution)**.** *Das Problem $\mathcal{F}\exists\mathsf{Subset}_S$ sei:*

Eingabe: *Zwei erkennbare Forest-Sprachen L und R und eine Menge X von ersetzbaren Zeichen.*

Frage: *Existiert eine Substitution σ mit $\sigma_S(L) \subseteq R$?*

Definition 4.2.6 (Existenz einer Substitution für Äquivalenz)**.** *Das Problem $\mathcal{F}\exists\mathsf{Equiv}_S$ sei:*

Eingabe: *Zwei erkennbare Forest-Sprachen L und R und eine Menge X von ersetzbaren Zeichen.*

Frage: *Existiert eine Substitution σ mit $\sigma_S(L) = R$?*

Satz 6. $\mathcal{F}\exists\mathsf{Subset}_S \leq_{\log} \mathcal{F}\exists\mathsf{Equiv}_S$

Beweis. Analog zu Satz 2 □

Für $\mathcal{F}\exists\mathsf{Subset}_S$ genügt es, die einelementigen Substitutionen zu betrachten, also diejenigen $\sigma : X \mapsto 2^{D_S(A,C)}$ mit $|\sigma(x)| = 1$ für alle $x \in X$, denn, wenn überhaupt ein σ existiert mit $\sigma_S(L) \subseteq R$, dann gilt das auch für jedes $\sigma' \subseteq \sigma$, also auch insbesondere für mindestens ein einelementiges.

4.3. Blattsubstitution

Zunächst definieren wir Substitutitonen lediglich für Blätter. Das Ersetzen der Blätter funktioniert mittels:

Definition 4.3.1 (Blattersetzung)**.** *Sei $\theta : X \to \mathcal{F}_C(A)$. Dann erweitere θ zu*

$$\begin{aligned}
\theta'_l : \quad \mathcal{F}_{A\cup X}(A) &\to \mathcal{F}_{A\cup C}(A) \\
x &\mapsto \theta(x) \quad x \in X \\
ag &\mapsto a\theta_l(g) \quad a \in A, g \in \mathcal{F}_X(A) \\
g_1 + \cdots + g_k &\mapsto \theta_l(g_1) + \cdots + \sigma_l(g_k) \\
0 &\mapsto 0 \\
\theta_l : 2^{\mathcal{F}_X(A)} &\to 2^{\mathcal{F}_C(A)} \\
L &\mapsto \left\{\theta'_l(f) \mid f \in L\right\}
\end{aligned}$$

Damit erhalten wir wie Engelfriet und Schmidt in [ES77] zwei Substitutionsbegriffe

Definition 4.3.2 (oi-Blattsubstitution, L_{oi}-Substitution)**.** *Sei $\sigma : X \to 2^{\mathcal{F}_C(A)} \setminus \{\emptyset\}$. Erweitere dies zu:*

$$\begin{aligned}
\sigma_{L_{oi}} : \quad & \mathcal{F}_X(A) \to 2^{\mathcal{F}_C(A)} \\
& x \mapsto \sigma(x) \quad x \in X \\
& af \mapsto \left\{af' \mid f' \in \sigma'_{L_{oi}}(f)\right\} \quad a \in A, f \in \mathcal{F}_X(A) \\
& g_1 + \cdots + g_k \mapsto \left\{g'_1 + \cdots + g'_k \mid \forall 1 \leq i \leq k : g'_i \in \sigma'_{L_{oi}}(g_i)\right\} \quad \forall 1 \leq i \leq k : g_i \in \mathcal{F}_X(A) \\
& 0 \mapsto \{0\}
\end{aligned}$$

Definition 4.3.3 (io-Blattsubstitution)**.** *Sei $\sigma : X \to 2^{\mathcal{F}_C(A)} \setminus \{\emptyset\}$. Dann erweitere dies zu:*

$$\begin{aligned}
\sigma_{L_{io}} : & \mathcal{F}_X(A) \to 2^{\mathcal{F}_C(A)} \\
& f \mapsto \left\{\theta'_l(f) \mid \theta : X \to \mathcal{F}_C(A), \theta \in \sigma\right\}
\end{aligned}$$

Wir werden für diese Substitutionsbegriffe die folgenden Resultate erhalten:

Tabelle 4.1.: Resultate für $L.$-Substitutionen (jeweils Vollständigkeit)

S	$\mathcal{F}$Reachable$_S$	$\mathcal{F}$Subset$_S$	$\mathcal{F}$Subset$'_S$	$\mathcal{F}$Equiv$_S$	$\mathcal{F}\exists$Subset$_S$	$\mathcal{F}\exists$Equiv$_S$
L_{oi}	P	P	P	P	NP	NP
L_{io}	NP	coNP	NP	DP	NP	Σ_2^{P}

4.3.1. Saturierung von Blattsubstitutionen

Für eine (io- oder oi-) Blattsubstitution definieren wir:

Definition 4.3.4 (Saturierte Blattsubstitution)**.** *Sei $\sigma : X \to 2^{\mathcal{F}(A)} \setminus \{\emptyset\}$ und $R \in \mathcal{F}\text{Rec}_A$ eine erkennbare Forest-Sprache, die von der endlichen Forest-Algebra $\mathcal{A} = (H, V)$ mit dem Homomorphismus φ bzw. von einem endlichen Forest-Automaten $M = (H, A, \delta, E)$ erkannt wird.*

Dann definiere die saturierte Blattsubstitution bezüglich R als

$$\begin{aligned}
\hat{\sigma}^{L_{io/oi},R} : & X \to \mathcal{F}(A) \\
& x \mapsto \bigcup_{s \in \sigma(x)} [s]_R
\end{aligned}$$

wobei $[s]_R = \{t \in \mathcal{F}(A) \mid \varphi(s) = \varphi(t)\}$ bzw. $[s]_R = \left\{t \in \mathcal{F}(A) \mid s^M = t^M\right\}$

Schreibe auch $\hat{\sigma}^{L_{io/oi},M}$ oder $\hat{\sigma}^{L_{io/oi},\mathcal{A}}$, um die Forest-Algebra oder den Forest-Automaten explizit anzugeben.

Lemma 4 (Saturierte Blattsubstitution). Sei $\sigma : X \to \mathcal{F}(A)$, $S \in \{L_{\text{oi}}, L_{\text{io}}\}$ und $L \subseteq \mathcal{F}_V(A)$, $R \subseteq \mathcal{F}(A)$ Forest-Sprachen. Dann ist $\sigma_S(L) \subseteq R$ genau dann, wenn $\hat{\sigma}_S^{S,R}(L) \subseteq R$.

Beweis. Seien σ, L, R wie angegeben.

$\sigma_S(L) \subseteq R \implies \hat{\sigma}_S^{S,R}(L) \subseteq R$. Sei $r \in \hat{\sigma}_S^{S,R}(L)$. Dann gibt es ein $l \in L$ mit $r \in \hat{\sigma}_S^{S,R}(\{l\})$. Es gibt nun für jedes Vorkommen eines $x \in X$ in l einen zugehörigen Forest $s \in \hat{\sigma}^{S,R}(x)$, durch den dieses in r ersetzt wurde. Dann gibt es nach der Definition von $\hat{\sigma}^{S,R}$ auch ein $s' \in \sigma(x)$ mit $\varphi(s) = \varphi(s')$ (möglicherweise $s = s'$). Betrachte nun das $r' \in \mathcal{F}(A)$, das man erhält, wenn man in l jedes Vorkommen eines $x \in X$ durch das so an dieser Stelle erhaltene s' ersetzt. Damit ist $\varphi(r) = \varphi(r')$ und $r' \in \sigma_S(l) \subseteq R$. Also ist $r \in R$.

$\hat{\sigma}_S^{S,R}(L) \subseteq R \implies \sigma_S(L) \subseteq R$. Offensichtlich ist $\sigma_S(L) \subseteq \hat{\sigma}_S^{S,R}(L) \subseteq R$.

□

Da eine saturierte $L_{\text{io/oi}}$-Substitution durch eine Menge der Elemente der horizontalen Algebra für jedes $x \in X$ eindeutig bestimmt ist, existieren $\leq 2^{|H| \cdot |X|}$ verschiedene saturierte Substitutionen, die sich alle in Platz $\leq |X| \cdot (|H| + |R|)$ kodieren lassen.

4.3.2. Erreichbarkeit von Zustandskombinationen

Satz 7. $\mathcal{F}\textsf{Reachable}_{L_{\text{oi}}} \in \textsf{P}$.

Beweis. Berechne $\text{reachable}_{L_{\text{oi}}}(\sigma, M_L, M_R)$ mit folgendem Markierungsalgorithmus:

Markiere zu Beginn die Paare $(0^{M_L}, 0^{M_R})$ sowie (x^{M_L}, r) für alle $x \in X$, wenn es ein $\xi \in E_{\sigma(x)}$ gibt, sodass (ξ, r) erreichbar ist im Produktautomat $M_{\sigma(x)} \times M_R$. Diese (ξ, r) lassen sich wiederum mit einem Markierungsalgorithmus wie im Beweis zu Satz 1 in Polynomialzeit bestimmen. Markiere anschließend wiederholt $(p_1 + q_1, p_2 + q_2)$ und $(\delta(a, p_1), \delta(a, q_1))$ für bereits markierte (p_1, q_1) und (p_2, q_2) sowie $a \in A$. □

Satz 8. $\mathcal{F}\textsf{Reachable}_{L_{\text{io}}} \in \textsf{NP}$

Beweis. Es ist $\sigma_{L_{\text{io}}}(L) = \bigcup_{\theta \in \sigma} \theta(L) = \bigcup_{\theta \in \sigma} \theta_{L_{\text{oi}}}(L)$.

Rate zunächst ein $\theta \in \sigma$ für jedes $1 \leq i \leq k$ und entscheide dann $\mathcal{F}\textsf{Reachable}_{L_{\text{oi}}}$ auf der Eingabe θ, L, R, M_i. □

4.3.3. Teilmengenproblem mit Substitution

Satz 9. $\mathcal{F}\mathsf{Subset}_{L_{\text{oi}}}$ ist P-vollständig.

Beweis. $\mathcal{F}\mathsf{Subset}_{L_{\mathbf{oi}}}$ **ist** P**-schwierig.** Reduziere von $\mathcal{F}$Subset. Belasse beide Teilmengen und wähle eine Substitution σ, die jedes substituierte Blatt durch sich selbst ersetzt, also $\forall x \in X : \sigma(x) = x$ und damit auch $\forall f \in \mathcal{F}_{X \cup A}(A) : \sigma_{L_{\text{oi}}}(f) = f$. Die Wahl der Menge X ist unerheblich.

$\mathcal{F}\mathsf{Subset}_{L_{\mathbf{oi}}} \in$ P**.** Folgt direkt aus Satz 7 und Satz 3.

□

Satz 10. $\mathcal{F}\mathsf{Subset}_{L_{\text{io}}}$ ist coNP-vollständig.

Beweis. $\mathcal{F}\mathsf{Subset}_{L_{\mathbf{io}}} \in$ coNP**.** Folgt direkt aus Satz 8 und Satz 3.

$\mathcal{F}\mathsf{Subset}_{L_{\mathbf{io}}}$ **ist** coNP**-schwierig.** Reduziere vom Tautologieproblem für aussagenlogische Formeln. Sei also eine aussagenlogische Formel F gegeben.

Betrachte nun $\mathcal{F}\mathsf{Subset}_{L_{\text{io}}}$ auf Eingabe von

$$\begin{aligned} L &= \langle F \rangle_{\mathcal{F}} \\ \sigma\colon\ X &\to 2^{\mathcal{F}(A)} \\ x &\mapsto \{\wedge, \vee\} \\ R &= \mathsf{True}_{\mathcal{F}} \end{aligned}$$

wobei X die Menge der atomaren Formeln in F sei.

Jedes $\theta \in \sigma$ ersetzt jede Variable durch entweder $\wedge$ oder $\vee$ und entspricht damit einer Belegung. Es ist also $\sigma_{L_{\text{io}}}(L) = \left\{ \langle F \rangle_{\mathcal{F}}^{\mathcal{A}} \mid \mathcal{A} \text{ passende Belegung} \right\} \subseteq \mathsf{True}_{\mathcal{F}}$ genau dann, wenn F eine Tautologie ist.

□

Satz 11. $\mathcal{F}\mathsf{Subset}'_{L_{\text{io}}}$ ist NP-vollständig.

Beweis. $\mathcal{F}\mathsf{Subset}'_{L_{\mathbf{io}}} \in$ NP**.** Folgt direkt aus Satz 8 und Satz 4.

$\mathcal{F}\mathsf{Subset}'_{L_{\mathbf{io}}}$ **ist** NP**-schwierig.** Wir reduzieren von dem Problem X3C, vergleiche hierzu [GJ79, S.53]:

Eingabe: Eine endliche Menge X mit $|X| = 3q$ und 3-elementige Teilmengen $C \subseteq \binom{X}{3}$.

Frage: Gibt es ein $C' \subseteq C$ so, dass es für jedes $x \in X$ genau ein $c \in C'$ gibt mit $x \in c$?

Konstruiere aus einer X3C-Instanz mit $X = \{x_1, \ldots, x_k\}$ und $C = \{c_1, \ldots, c_l\}$ die folgende $\mathcal{F}\mathsf{Subset}'_{L_{\text{io}}}$-Instanz. Hierbei seien ohne Einschränkung die c_i geordnet ($c_i = c_{i1}c_{i2}c_{i3}$).

$$\begin{aligned}
A &= \{r\} \dot\cup \{1, \ldots, l\} \dot\cup X \\
R &= \{x_1 w_1 + \cdots + x_k w_k \mid \forall 1 \leq i \leq k \colon w_i \in \{r\}^*\} \\
L &= \{x_1(l_{11} + \cdots + l_{1l}) + \cdots + x_k(l_{k1} + \cdots + l_{kl})\} \\
&\quad \cup \{x_1 w_1 + \cdots + x_k w_k \mid \forall 1 \leq i \leq k \colon w_i \in \{r\}^* \setminus \{r\}\} \\
\sigma &\colon \{1, \ldots, l\} \to \mathcal{F}(A) \\
&\qquad\quad i \mapsto \{0, r\}
\end{aligned}$$

wobei

$$l_{ki} = \begin{cases} i & x_k \in c_i, x_k = c_{ij} \\ 0 & \text{sonst} \end{cases}$$
$$\{r\}^* = \{0, r, r + r, r + r + r, \ldots\}$$

Eine Substitution σ entspricht hier genau einer Teilmenge $\{c_i \mid \sigma(i) = r\} = C' \subseteq C$. Genau dann, wenn $\sigma(L) \supset R$ gilt, ist $x_1 r + \cdots + x_k r \in \sigma(L)$, womit jedes $x \in X$ genau von einem $c_i \in C'$ getroffen wird.

In dieser Konstruktion ist $\sigma_S(L) \subseteq R$ trivial gegeben, was wir später für den Beweis von Satz 13 verwenden werden.

□

4.3.4. Äquivalenzproblem mit Substitution

Satz 12. $\mathcal{F}\mathsf{Equiv}_{L_{\text{oi}}}$ ist P-vollständig.

Beweis. Die P-Schwierigkeit folgt direkt aus Satz 2 und Satz 9. Dass $\mathcal{F}\mathsf{Equiv}_{L_{\text{oi}}} \in \mathsf{P}$, folgt aus Satz 7 und Satz 5. □

Satz 13. $\mathcal{F}\mathsf{Equiv}_{L_{\text{io}}}$ ist $\mathsf{DP} = \mathsf{NP} \wedge \mathsf{coNP}$-vollständig.

Beweis. $\mathcal{F}\mathsf{Equiv}_{L_{\text{io}}} \in \mathsf{DP}$. Folgt direkt aus Satz 8 und Satz 5.

$\mathcal{F}\mathsf{Equiv}_{L_{\text{io}}}$ **ist** DP**-schwierig.** Ein Problem in DP hat die Form $L_1 \cap \overline{L_2}$ für $L_1, L_2 \in \mathsf{NP}$. Eine Instanz desselben ist demnach eine Instanz von L_1 und eine von L_2.

Nach Satz 11 können wir dann die Instanz von L_1 in logarithmischem Platz in eine $\mathcal{F}\mathsf{Subset}'_S$-Instanz mit selbem Ergebnis umwandeln. Aufgrund der Form derselben in der Reduktion ist diese auch eine $\mathcal{F}\mathsf{Equiv}_S$-Instanz mit dem selben Ergebnis.

Nach Satz 9 können wir nun auch die Instanz von $\overline{L_2}$ in eine $\mathcal{F}\mathsf{Subset}_S$-Instanz und wegen Satz 2 auch in eine $\mathcal{F}\mathsf{Equiv}_S$-Instanz mit dem selben Ergebnis umwandeln.

Diese beiden $\mathcal{F}\mathsf{Equiv}_S$-Instanzen lassen sich durch disjunkte Vereinigung der X, L und R sowie durch ein neues σ, dass sich wie beide alten verhält (beachte die disjunkten Definitionsbereiche), zu einer $\mathcal{F}\mathsf{Equiv}_S$-Instanz zusammenfassen.

□

4.3.5. Existenz einer Substitution für Teilmenge

Wir haben bereits allgemein festgestellt, dass es für das Problem $\mathcal{F}\exists\mathsf{Subset}_S$ genügt, einelementige Substitutionen zu betrachten. Für einelementige Substitutionen ist aber $\sigma_{L_{\text{oi}}} = \sigma_{L_{\text{io}}}$. Betrachten wir hierfür jeweils die saturierte Substitution $\hat{\sigma}_R$, so lässt sich diese Substitution in polynomieller Länge kodieren. Wir erhalten nun zunächst:

Satz 14. $\mathcal{F}\exists\mathsf{Subset}_{L_{\text{oi}}}$ ist NP-vollständig.

Beweis. $\mathcal{F}\exists\mathsf{Subset}_{L_{\text{oi}}} \in \mathsf{NP}$**.** Jede saturierte Substitution lässt sich in polynomiellem Platz kodieren. Rate also eine solche Substitution σ und überprüfe danach in Polynomialzeit, ob $\sigma_{L_{\text{oi}}}(L) \subseteq R$.

$\mathcal{F}\exists\mathsf{Subset}_{L_{\text{oi}}}$ **ist** NP**-schwierig.** Reduziere von SAT. Sei eine aussagenlogische Formel F gegeben. Betrachte $\mathcal{F}\exists\mathsf{Subset}_{L_{\text{oi}}}$ auf Eingabe von

$$L = \langle F \rangle_{\mathcal{F}} \qquad\qquad R = \mathsf{True}_{\mathcal{F}}$$

Dann gibt es ein $\sigma : X \to 2^{\mathcal{F}(A)}$ mit $\sigma_{L_{\text{oi}}}(L) \subseteq R$ genau dann, wenn es ein einelementiges gibt. Diese Substitution entspricht dann genau einem Modell für F.

□

Da wie oben festgestellt für einelementige Substitutionen $\sigma_{L_{\text{io}}} = \sigma_{L_{\text{oi}}}$ ist und das Problem für einelementige Substitutionen äquivalent ist, folgt hieraus direkt:

Satz 15. $\mathcal{F}\exists\mathsf{Subset}_{L_{\text{io}}}$ ist NP-vollständig.

4.3.6. Existenz einer Substitution für Gleichheit

Satz 16. $\mathcal{F}\exists\mathsf{Equiv}_{L_{\text{oi}}}$ ist NP-vollständig.

Beweis. Die NP-Schwierigkeit folgt direkt aus Satz 14 und Satz 6. Dass $\mathcal{F}\exists\mathsf{Equiv}_{L_{\text{oi}}} \in \mathsf{NP}$, folgt analog zu $\mathcal{F}\exists\mathsf{Subset}_{L_{\text{oi}}} \in \mathsf{NP}$. □

Satz 17. $\mathcal{F}\exists\mathsf{Equiv}_{L_{\text{io}}}$ ist Σ_2^{P}-vollständig.

Beweis. $\mathcal{F}\exists\mathsf{Equiv}_{L_{\text{io}}}$ **ist** $\Sigma_2^{\mathsf{P}} = \mathsf{NP}^{\mathsf{NP}}$**-schwierig.** Reduziere von B_2 (vergleiche [Sto76]):

Eingabe: Eine Formel der Form $\exists x_1 \ldots \exists x_k \forall y_1 \ldots \forall y_l\, F$, wobei F eine aussagenlogische Formel in den Variablen $x_1, \ldots, x_k, y_1, \ldots, y_l$ ist.

Frage: Ist die logische Aussage wahr?

Betrachte das folgende Sprachungleichungssystem:

$$\sigma_{L_{\text{io}}}(\langle F\rangle_{\mathcal{F}}) \subseteq \mathsf{True}_{\mathcal{F}}$$
$$\sigma_{L_{\text{io}}}(y_1 + \cdots + y_l) = \{v_1 + \cdots + v_l \mid \forall 1 \leq i \leq l \colon v_i \in \mathbb{B}\}$$

Dann ist

$$\begin{aligned}\sigma_{L_{\text{io}}}(L) = R \\ &\iff \sigma_{L_{\text{io}}}\left(\{\langle F\rangle_{\mathcal{F}}\} \cup \mathsf{True}_{\mathcal{F}}\right) = \mathsf{True}_{\mathcal{F}} \wedge \sigma_{L_{\text{io}}}\left(\{y_1 + \cdots + y_l\}\right) = \{v_1 + \cdots + v_l \mid v_i \in \{0,1\}\} \\ &\iff \sigma_{L_{\text{io}}}\left(\{\langle F\rangle_{\mathcal{F}}\}\right) \subseteq \mathsf{True}_{\mathcal{F}} \wedge \forall 1 \leq i \leq l : \{0,1\} \subseteq \sigma(y_i)\end{aligned}$$

σ entspricht nun also einer Menge von Modellen, die unter der Veränderung der Belegung eines y_i abgeschlossen ist.

$\mathcal{F}\exists\mathsf{Equiv}_{L_{\mathbf{io}}} \in \Sigma_2^{\mathsf{P}} = \mathsf{NP}^{\mathsf{NP}}$ Rate eine Substitution σ und überprüfe mittels des Orakels, ob $\sigma_{I_{\text{io}}}(L) = R$.

□

4.4. Substitution an inneren Knoten

Definition 4.4.1 (I_{io}-Substitution und I_{oi}-Substitution)**.** *Sei* $\sigma : X \to 2^{\mathcal{F}_{B\cup\{\Box\}}(A)}$ *und* $x \in \{io, oi\}$.

$$\begin{aligned}\sigma'_{I_x} :&\mathcal{F}_{B\cup X}\left(A \cup X\right) \to 2^{\mathcal{F}_B(A)} \\ &xf \mapsto (\sigma(x))\left[\Box \leftarrow \sigma_{I_x}(f)\right]_{L_x} \quad x \in X, f \in \mathcal{F}_{B\cup X}\left(A \cup X\right) \\ &af \mapsto a\sigma_{I_x}(f) \quad a \in A, f \in \mathcal{F}_{B\cup X}\left(A \cup X\right) \\ g_1 + \cdots + g_k &\mapsto \left\{g'_1 + \cdots + g'_k \mid \forall 1 \leq i \leq k : g'_i \in \sigma_{I_x}(g_k)\right\} \quad \forall 1 \leq i \leq k : g_i \in \mathcal{F}_{B\cup X}\left(A \cup X\right) \\ &0 \mapsto \{0\}\end{aligned}$$

Wir werden hierfür die folgenden Resultate erhalten:

Tabelle 4.2.: Resultate für $I.$-Substitutionen

S	$\mathcal{F}\mathsf{Reachable}_S$	$\mathcal{F}\mathsf{Subset}_S$	$\mathcal{F}\mathsf{Equiv}_S$	$\mathcal{F}\exists\mathsf{Subset}_S$	$\mathcal{F}\exists\mathsf{Equiv}_S$
I_{oi}	P-vollständig	P-vollständig		offen	PSPACE-schwierig
I_{io}					offen

4.4.1. Saturierung von Substitutionen

Auch zu I_x-Substitutionen ($x \in \{\text{io}, \text{oi}\}$) definieren wir wieder saturierte Substitutionen:

Definition 4.4.2. *Sei $\sigma : X \to 2^{\mathcal{F}_{C\cup\{\square\}}(A)}$, $x \in \{io, oi\}$ und $R \subseteq \mathcal{F}_C(A)$ eine Forest-Sprache, die von der endlichen Forest-Algebra $\mathcal{A} = (H, V)$ mit dem Homomorphismus φ oder von einem Forest-Automaten $M = (H, A, \delta, E)$ erkannt wird. Dann definiere die saturierte Substitution bezüglich R als*

$$\begin{aligned} \hat{\sigma}_R : X &\to \mathcal{F}_{C\cup\{\square\}}(A) \\ x &\mapsto \bigcup_{s\in\sigma(x)} [s]_{R/\square} \end{aligned}$$

$$\text{wobei } [s]_{R/\square} := \left\{t \in \mathcal{F}(A) \mid \forall h \in H\colon \varphi(\varphi^{-1}(h)\,[s \leftarrow \square]) = \varphi(\varphi^{-1}(h)\,[t \leftarrow \square])\right\}$$

$$\text{bzw. } [s]_{R/\square} := \left\{t \in \mathcal{F}(A) \mid \forall h \in H \forall c \text{ mit } c^M = h\colon (c\,[s \leftarrow \square])^M = (c\,[t \leftarrow \square])^M\right\}.$$

Schreibe wie bei Blattsubstitutionen statt $\hat{\sigma}_R$ auch $\hat{\sigma}_{\mathcal{A}}$ oder $\hat{\sigma}_M$.

Lemma 5 (Saturierte Substitution). Seien $\sigma : X \to \mathcal{F}(A)$ eine Substitution, $S \in \{\text{io}, \text{oi}\}$ und $L \subseteq \mathcal{F}(A \cup X)$, $R \subseteq \mathcal{F}(A)$ Forest-Sprachen. Dann ist $\sigma_{I_S}(L) \subseteq R$ genau dann, wenn $\hat{\sigma}_{R,I_S}(L) \subseteq R$.

Beweis. Seien σ, L und R wie angegeben.

- Sei $\sigma_{I_S}(L) \subseteq R$ und $r \in \hat{\sigma}(L)$. Dann gibt es ein $l \in L$ mit $r \in \hat{\sigma}(l)$. Es gibt nun für jedes Vorkommen eines $x \in X$ in l einen zugehörigen Forest $s \in \hat{\sigma}_R(x)$, der eingesetzt werden kann um r zu erhalten. Dann gibt es nach der Definition von $\hat{\sigma}_R$ ein $s' \in \sigma(x)$ mit $\varphi((s)\left[\square \leftarrow \varphi^{-1}(h)\right]_{L_S}) = \varphi((s')\left[\square \leftarrow \varphi^{-1}(h)\right]_{L_S})$ bzw. $\left(s\left[\square \leftarrow f\right]_{L_S}\right)^M = \left(s'\left[\square \leftarrow f\right]_{L_S}\right)^M$ für alle $h \in H$ bzw. $f \in \mathcal{F}(A)$.

  Betrachte nun das $r' \in \mathcal{F}(A)$, das man erhält, wenn man in l jedes Vorkommen eines $x \in X$ durch das für diese Stelle erhaltene s' (statt durch s) ersetzt. Dann ist $\varphi(r) = \varphi(r')$ bzw. $r^M = r'^M$ und damit $r' \in \sigma_{I_S}(l) \subseteq R$. Also ist $r \in R$.

- Sei $\hat{\sigma}(L) \subseteq R$ Offensichtlich ist $\sigma \subseteq \hat{\sigma}_R$ und damit $\sigma_{I_S}(L) \subseteq \hat{\sigma}_{R,I_S}(L) \subseteq R$.

□

4.4.2. Erreichbarkeit von Zustandskombinationen

Satz 18. $\mathcal{F}\mathsf{Reachable}_{I_{\text{oi}}} \in \mathsf{P}$.

Beweis. Hierzu verwenden wir wieder einen Markierungsalgorithmus über den Paaren aus $H_L \times H_R$.

Zu Beginn sei lediglich $(0^{M_L}, 0^{M_R})$ markiert. Wir markieren, wenn (l, r) und (l', r') markiert sind auch $(l + l', r + r')$ sowie (al, ar) für $a \in A \setminus X$.

Nun bleibt noch der Fall, in dem eine Ersetzung stattfindet. Sei S die Menge der bisher markierten Paare. Dann berechne für jedes $x \in X$ und jedes $c_l \in H_L$ diejenigen $(s, r) \in \text{reachable}_{L_{\text{oi}}}\left(\left[\square \leftarrow \{c_r \mid (c_l, c_r \in S\}, M_{\sigma(x)}, M_R\right]\right)$ mit $s \in E_{\sigma(x)}$ und markiere jeweils auch $(\delta_L(x, c_l), r)$.

Wiederhole alle Markierungsschritte bis keine Veränderung mehr eintritt.

Wenn es einen Forest f gibt, der die gegebene Zustandskombination erreicht, kann dieser mit den Operationen $+$ und $a\cdot$ für $a \in A$ zusammengesetzt werden und das entsprechende Paar wird oben markiert. Zudem ließe sich in der Ausführung des obigen Algorithmus für jedes Paar ein Zeuge generieren, der diese Zustandskombination erreicht. □

Satz 19. $\mathcal{F}\mathsf{Reachable}_{I_{\text{io}}} \in \mathsf{P}$.

Beweis. Gehe vor wie im oi-Fall. Für die Elemente der Form $(\delta_L(x, c_l), r)$ bestimme aber stattdessen für jedes $(c_l, c_r) \in S$ separat $\text{reachable}_{L_{\text{oi}}}\left(\left[\square \leftarrow \{c_r\}\right], M_{\sigma(x)}, M_R\right)$. □

4.4.3. Teilmengenproblem mit Substitution

Satz 20. $\mathcal{F}\mathsf{Subset}_{I_{\text{oi}}}$ ist P-vollständig.

Beweis. **$\mathcal{F}\mathsf{Subset}_{I_{\text{oi}}}$ ist P-schwierig.** Reduziere von $\mathcal{F}\mathsf{Subset}_{I_{\text{oi}}}$.

> Für jedes $\sigma : X \to \mathcal{F}_C(A)$ und jede Forest-Sprache $L \subseteq \mathcal{F}_{C \cup X}(A)$ mit $A \cap X = \emptyset$ gilt $\sigma_{L_{\text{oi}}}(L) = \sigma_{I_{\text{oi}}}(L)$. Belasse also die gegebene Eingabe.

$\mathcal{F}\mathsf{Subset}_{I_{\text{oi}}} \in \mathsf{P}$. Folgt direkt aus Satz 18 und Satz 3.

□

Satz 21. $\mathcal{F}\mathsf{Subset}_{I_{\text{io}}}$ ist P-vollständig.

Beweis. **$\mathcal{F}\mathsf{Subset}_{I_{\text{io}}} \in \mathsf{P}$.** Folgt direkt aus Satz 19 und Satz 3.

$\mathcal{F}\mathsf{Subset}_{I_{\text{io}}}$ ist P-schwierig. Reduziere von $\mathcal{F}\mathsf{Subset}_{L_{\text{oi}}}$ analog zu Satz 20.

> Auch hier ist für jedes $\sigma : X \to \mathcal{F}_C(A)$ und jeden Forest-Sprache $L \subseteq \mathcal{F}_{C \cup X}(A)$ $\sigma_{L_{\text{oi}}}(L) = \sigma_{I_{\text{io}}}(L)$. Belasse also auch hier die Eingabe.

□

4.4.4. Äquivalenzproblem mit Substitution

Satz 22. $\mathcal{F}\mathsf{Equiv}_{I_{\text{oi}}}$ ist P-vollständig.

Beweis. Dass $\mathcal{F}\mathsf{Equiv}_{I_{\text{oi}}} \in \mathsf{P}$, folgt direkt aus Satz 18 und Satz 5. P-Schwierigkeit folgt mit Satz 20 und Satz 2. □

Satz 23. $\mathcal{F}\mathsf{Equiv}_{I_{\text{io}}}$ ist P-vollständig.

Beweis. Dass $\mathcal{F}\mathsf{Equiv}_{I_{\text{io}}} \in \mathsf{P}$, folgt direkt aus Satz 19 und Satz 5. P-Schwierigkeit folgt mit Satz 21 und Satz 2. □

4.4.5. Existenz einer Substitution für Gleichheit

Satz 24. $\mathcal{F}\exists\mathsf{Equiv}_{I_{\text{oi}}}$ ist PSPACE-schwierig

Beweis. Reduziere von QBF.

Auf Eingabe $\exists x_1 \forall y_1 \ldots \exists x_k \forall y_k F$ gehen wir wie folgt vor:

Zunächst erweitern wir $\mathsf{True}_{\mathcal{F}}$ um einen Operator $=$, der als negiertes exklusives Oder funktioniert, um $\searrow$ und $\swarrow$, die jeweils den rechten bzw. linken Teil des Paares kopieren und den anderen ignorieren, sowie um Zeichen assert(x), label(x) für $x \in X$, die auf die Identitätsfunktion abgebildet werden. Die Konstruktion der erkennenden Forest-Algebra funktioniert analog zu der für $\mathsf{True}_{\mathcal{F}}$.

Zudem konstruieren wir die Forest-Sprache über Binärbäumen, die nur $\swarrow$, $\searrow$, label(x) und assert(x) beachtet (und alle anderen Zeichen als Identität behandelt). Diese soll nur solche Forests enthalten, in denen unter jedem assert(x) der durch die $\swarrow$ und $\searrow$ angegebene Pfad durch einen Binärbaum zu einem label(x) führt. Ein Element der horizontalen Algebra ist dabei aus $(1 \cup X)^{\leq 2}$ und beschreibt, welche label(x) in dem linken und rechten Teilbäumen erreicht werden. Diese sind eindeutig. Ein Element der vertikalen Algebra sind nun nicht beliebige Funktionen, sondern sind eindeutig beschrieben dadurch:

- Welche (einzelnen) $x \in X$ in dem eingesetzten horizontalen Element stehen dürfen,
- welche (einzelnen) $x \in X$ dann oben in dem rechten bzw. linken Teilbaum erreicht werden — diese können entweder ein konstantes $x \in X$ oder ein $\swarrow$ oder $\searrow$ sein, um diese von einem Teil des eingesetzten horizontalen Elements zu übernehmen.

Dies lässt sich als ein Element von $X^{\leq 2} \times (\{\swarrow, \searrow\} \cup X)^{\leq 2}$ darstellen.

Wir fordern nun (für neue $?_1$ und $?_2$):

$$\sigma_{I_{\text{oi}}}\left(\left\{\begin{array}{c} N \\ | \\ ?_{1/2} \end{array}\right\}\right) = \left\{\begin{array}{cc} \swarrow & \searrow \\ | & | \\ ?_{1/2} & ?_{1/2} \end{array}\right\}$$

Damit wertet der Forest mit label(x) an der Wurzel, N in den inneren Knoten dazwischen und mit assert$(y) \cdot f_y$ darunter für Forests f_y gleich aus wie f_x oder zu $\bot$. Dieses Konstrukt können wir also als eine Art Wörterbuch-Datenstruktur verwenden.

Wir fordern zusätzlich:

$$\sigma_{I_{\text{oi}}}\left(\left\{\begin{array}{c} F \\ | \\ ?_{1/2} \end{array}\right\}\right) = \left\{ \begin{array}{l} \vee \\ \quad\diagdown\ \diagup \\ \quad\neg \quad \langle F \rangle_{\mathcal{F}} \left[x \leftarrow \begin{array}{c} \text{assert}(x) \\ | \\ ?_{1/2} \end{array} \right] \\ \quad\quad | \\ \quad\quad \wedge \\ \quad\quad | \\ \quad = \quad \cdots \quad = \\ t_1 \quad x_1 \qquad t_k \quad x_k \end{array} \right.$$

$$: t_i \in \mathcal{F}\left(\left\{\begin{array}{c} \wedge, \vee, \neg, \begin{array}{c} \text{assert}(x) \\ | \\ 1 \end{array} \end{array} \middle| \; j \leq i, x_j \in X \right\}\right) \Bigg\}$$

$$\sigma_{I_{\text{oi}}}\left(\left\{\begin{array}{c} F \\ | \\ V_1(\mathcal{A}) \end{array} \middle| \; \mathcal{A} \text{ passende Belegung für } F \right\}\right) \subseteq \{\bot\} \cup \mathsf{True}'_{\mathcal{F}}$$

mit

$$V_{k+1}(\mathcal{A}) = 0$$

$$V_i(\mathcal{A}) = \begin{array}{c} N \\ \quad\diagdown\ \diagup \\ \quad N \quad V_{i+1}(\mathcal{A}) \\ \quad\diagdown\ \diagup \\ \text{label}(x_i) \quad \text{label}(y_i) \\ \quad | \qquad\quad | \\ \mathcal{A}(x_i) \qquad \mathcal{A}(y_i) \end{array}$$

Nach Satz 2 können die Sprachungleichungen mit $\subseteq$ zu solchen mit $=$ umgeformt werden und ein System zu einer einzelnen Gleichung.

Gibt es nun ein $\sigma_{I_{\text{oi}}}$, so ist die Formel $\forall y_1 \forall y_2 \ldots \forall y_k F'$ erfüllbar für ein F', in dem im Vergleich zu F die x_i durch Terme t_i ersetzt wurden, die nur von $y_1, \ldots, y_i$ abhängen. Diese definieren immer einen Wert für x_i.

Ist die ursprüngliche Formel erfüllbar, so kann man für jedes x_i eine solche Formel angeben, da jede Funktion $\mathbb{B}^r \to \mathbb{B}$ durch eine aussagenlogische Formel dargestellt werden kann. □

5. Fazit und Ausblick

Wir konnten in dieser Arbeit feststellen, dass das Prinzip der Saturierung wie bei regulären (Wort-)Sprachen auch bei regulären Forest-Sprachen anwendbar ist.

Weiterhin konnten wir einige Vollständigkeitsresultate für Probleme mit einfachen Substitutionsmodellen zeigen, durch Reduktion, meist von Logikproblemen, sowie durch Markierungsalgorithmen. Hier wäre eine Betrachtung der Probleme, die in Tabelle 4.2 als “offen” markiert sind, noch vielversprechend.

Daneben ist eine Ausweitung auf weitere Substitutionsmodelle möglich, beispielsweise Substitutionen in der Binärbaumkodierung eines Forests oder Substitutionen auf Forests mit definierten Verzweigungsgraden.

Auch können noch weitere Problemstellungen betrachtet werden wie die Berechnung einer maximalen oder der größten Substitution, die eine Sprachungleichung erfüllt. Die Betrachtung weiterer Entscheidungsprobleme scheint ebenfalls vielversprechend, um vollständige Probleme für weitere Klassen zu erhalten. Das Prinzip der Konstruktion von $\mathsf{True}_{\mathcal{F}}$ scheint hierbei leicht auf andere Strukturen über endlichen Mengen erweiterbar.

Die Beziehung zwischen Forest-Algebren und Forest-Automaten könnte ebenfalls weiter untersucht werden. Für die Lösung der hier betrachteten Probleme führen die beiden Beschreibungen trotz stark unterschiedlicher Größe zu gleichen Ergebnissen — das große vertikale Monoid bringt uns hier keinen Vorteil. Das wirft die Frage auf, inwiefern dies für weitere Probleme der Fall ist und für welche Probleme das vertikale Monoid gewinnbringend eingesetzt werden kann.

Literaturverzeichnis

[BW07] M. Bojańczyk, I. Walukiewicz. „Forest algebras“. In: *Logic and Automata: History and Perspectives*. Amsterdam University Press, 2007, S. 107–132 (zitiert auf S. 3, 9, 10, 12, 13).

[BWS12] M. Bojańczyk, I. Walukiewicz, H. Straubing. „Wreath Products of Forest Algebras, with Applications to Tree Logics“. In: *Logical Methods in Computer Science* Volume 8, Issue 3 (Sep. 2012). DOI: 10.2168/LMCS-8(3:19)2012. URL: https://lmcs.episciences.org/1215 (zitiert auf S. 12, 16).

[DS10] A. Delignat-Lavaud, H. Straubing. „An automaton model for forest algebras“. internship report. Aug. 2010. URL: https://antoine.delignat-lavaud.fr/doc/report-M1.pdf (zitiert auf S. 12).

[ES77] J. Engelfriet, E. M. Schmidt. „IO and OI. I“. In: *Journal of Computer and System Sciences* 15.3 (Dez. 1977), S. 328–353. DOI: 10.1016/s0022-0000(77)80034-2. URL: https://doi.org/10.1016%2Fs0022-0000%2877%2980034-2 (zitiert auf S. 23).

[GJ79] M. Garey, D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. Mathematical Sciences Series. W. H. Freeman, 1979. ISBN: 9780716710448. URL: https://books.google.de/books?id=fjxGAQAAIAAJ (zitiert auf S. 25).

[Lad75] R. E. Ladner. „The Circuit Value Problem is Log Space Complete for P“. In: *SIGACT News* 7.1 (Jan. 1975), S. 18–20. ISSN: 0163-5700. DOI: 10.1145/990518.990519. URL: https://doi.org/10.1145/990518.990519 (zitiert auf S. 18).

[Lib05] L. Libkin. *Logics for Unranked Trees: An Overview*. 2005. URL: https://homepages.inf.ed.ac.uk/libkin/papers/icalp05-proc.pdf (zitiert auf S. 13).

[Sto76] L. J. Stockmeyer. „The polynomial-time hierarchy“. In: *Theoretical Computer Science* 3.1 (1976), S. 1–22. ISSN: 0304-3975. DOI: https://doi.org/10.1016/0304-3975(76)90061-X. URL: http://www.sciencedirect.com/science/article/pii/030439757690061X (zitiert auf S. 27).

Alle URLs wurden zuletzt am 03. 06. 2020 geprüft.

A. Verwendete Notation

$\mathbb{B}$	$= \{0, 1\}$
$\mathbb{Z}$	Ganze Zahlen
$\mathbb{Z}/n\mathbb{Z}$	Resklassenring modulo n
$M \dot{\cup} N$	Disjunkte Vereinigung, $M \cup N$ mit $M \cap N = \emptyset$
$M \triangle N$	Symmetrische Differenz von M und N
2^M	Potenzmenge einer Menge M
$\mathcal{O}(f(n))$	$= \left\{ g \colon \mathbb{N} \to \mathbb{N} \mid \limsup_{n\to\infty} \frac{f(n)}{g(n)} < \infty \right\}$

A.1. Komplexitätstheorie

$A \leq_{\log} B$	$A \subseteq \Sigma^*$ ist in logarithmischem Platz reduzierbar auf $B \subseteq \Gamma^*$. D.h. es existiert eine in logarithmischem Platz berechenbare Funktion f mit $\forall w \in \Sigma^* \colon w \in A \iff f(w) \in B$.
${C_1}^{C_2}$	Klasse der Sprachen in C_1 mit Orakel für die Sprache C_2 oder die Klasse C_2. → Abschnitt 1.1
Σ_2^{P}	$= \mathsf{NP}^{\mathsf{NP}}$ → Abschnitt 1.1

A.2. Forests und Kontexte

$\mathcal{C}_C(A)$	Menge der Kontexte über einem Alphabet A mit Blättern aus C (wenn nicht angegeben: $C = A$). → Definition 1.2.3
$\mathcal{F}_C(A)$	Menge der Forests über einem Alphabet A mit Blättern aus C (wenn nicht angegeben: $C = A$). → Definition 1.2.1
$\mathcal{T}_C(A)$	Menge der Bäume über einem Alphabet A mit Blättern aus C (wenn nicht angegeben: $C = A$). → Definition 1.2.1
$a \circ f$	Anfügen eines Alphabetzeichens a über einem Forest f→ Definition 1.2.1
0	Der leere Forest→ Definition 1.3.1
1	Loch in einem Kontext→ Definition 1.2.3
1	Der leere Kontext→ Definition 1.3.1
$\mathcal{SF}(f)$	Die Menge aller Unterforests eines Forests f → Definition 1.2.5
$\mathcal{SC}(f)$	Menge aller in f vorkommenden Kontexte → Definition 1.2.5
$\langle O \rangle_{\mathcal{F}}$	Kodierung eines mathematischen Objekts O als Forest. → Definition 2.3.1 (für aussagenlogische Formeln)

A.3. Forest-Algebren

act	Operation des vertikalen Monoids auf dem horizontalen → Definition 1.3.1
in_L	→ Definition 1.3.1
in_R	→ Definition 1.3.1
$h_1{+}h_2$	Horizontale Verknüpfung von h_1 und h_2 → Definition 1.2.1 (freie Forest-Algebra), Definition 1.3.1
$v_1{\cdot}v_2$	Vertikale Verknüpfung von v_1 und v_2 → Definition 1.2.4 (freie Forest-Algebra), Definition 1.3.1
A^{Δ}	Die freie Forest-Algebra über einem Alphabet A. → Definition 1.3.3
reachable (L, R)	Menge der erreichbaren Zustände in $M_L \times M_R$. → Satz 1
$\text{reachable}_S\ (\sigma, L, R)$	Menge der erreichbaren Zustände in $M_L \times M_R$ unter der Substitution σ von der linken zur rechten Seite. → Definition 4.2.1

A.4. Probleme

$\mathcal{F}$Empty	Leerheitsproblem über erkennbaren Forest-Sprachen → Definition 3.0.1
$\mathcal{F}$Subset	Teilmengenproblem über erkennbaren Forest-Sprachen → Definition 3.0.2
$\mathcal{F}$Equiv	Äquivalenzproblem über ekennbaren Forest-Sprachen → Definition 3.0.3
$\mathcal{F}\text{Subset}_S$	Teilmengenproblem mit S-Substitution über erkennbaren Forest-Sprachen → Definition 4.2.2
$\mathcal{F}\text{Subset}'_S$	umgekehrtes Teilmengenproblem mit S-Substitution über erkennbaren Forest-Sprachen → Definition 4.2.3
$\mathcal{F}\text{Equiv}_S$	Äquivalenzproblem mit S-Substitution über erkennbaren Forest-Sprachen → Definition 4.2.4
$\mathcal{F}\exists\text{Subset}_S$	Existenz einer S-Substitution für Teilmenge → Definition 4.2.5
$\mathcal{F}\exists\text{Equiv}_S$	Existenz einer S-Substitution für Gleichheit → Definition 4.2.6
$\mathcal{F}\text{Reachable}_S$	Erreichbarkeit von Zustandskombinationen → Definition 4.2.1

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift