Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik
Institut für Theoretische Informatik

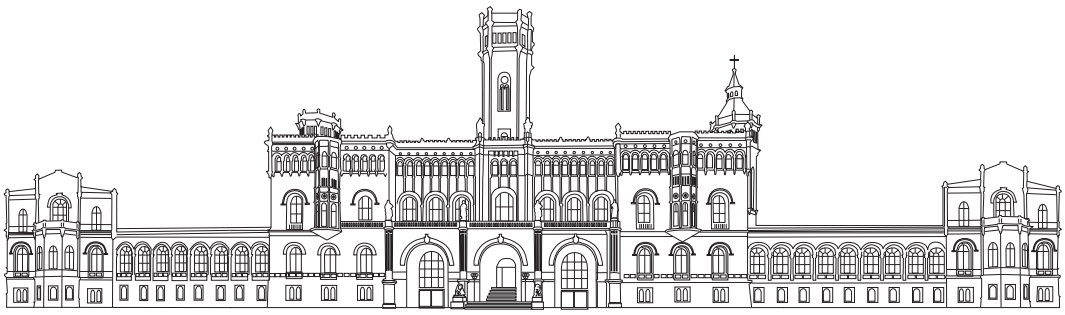
Bachelorarbeit

Entscheidbarkeit von Nichtstandard-Klassen der Prädikatenlogik

Decidability of Non-Standard Classes for Predicate Logic

Robin Schmöcker
Matrikelnr. 10024298

16. Juni 2021



Erstprüfer: Prof. Dr. rer. nat. Heribert Vollmer
Zweitprüfer: PD Dr. rer. nat. habil. Arne Meier
Betreuer: M. Sc. Anselm Haak

Erklärung der Selbstständigkeit

Hiermit versichere ich, die vorliegende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet zu haben. Die Arbeit hat in gleicher oder ähnlicher Form noch keinem anderen Prüfungsamt vorgelegen.

Lehrte, den 16. Juni 2021

_____

Robin Schmöcker

Inhaltsverzeichnis

1 Einleitung

Das klassische Entscheidungsproblem ist die Fragestellung, ob es einen Algorithmus gibt, der entscheidet, ob eine gegebene Formel der Prädikatenlogik erster Stufe erfüllbar ist. Dieses Entscheidungsproblem lässt sich auf verschiedene äquivalente Weisen formulieren, wie zum Beispiel, ob es einen Entscheidungsalgorithmus gibt, der entscheidet, ob eine gegebene Formel allgemeingültig ist. Diese Probleme sind deshalb äquivalent, da eine Formel genau dann allgemeingültig ist, wenn ihre Negation unerfüllbar ist [2].

Das Entscheidungsproblem wurde von Mathematikern wie David Hilbert als eins der zentralen Probleme der Mathematik angesehen, da es viele wichtige, teilweise noch heute ungelöste Probleme wie die Riemannsche Hypothese gab, die sich darauf reduzieren lassen, ob ein Satz der Prädikatenlogik erster Stufe erfüllbar ist [2].

Die Entscheidbarkeit von prädikatenlogischen Formeln ist jedoch ein viel schwierigeres Problem als die Entscheidbarkeit von aussagenlogischen Formeln, da es unendlich viele mögliche Interpretationen gibt. Tatsächlich wurde in den 1930er Jahren, unabhängig voneinander durch Church und Turing gezeigt, dass es keinen solchen Entscheidungsalgorithmus für beliebige prädikatenlogische Formeln erster Stufe gibt. Das Entscheidungsproblem wurde daher zu einem Klassifizierungsproblem: Welche Einschränkungen an Formeln muss ich wählen damit diese Klasse entscheidbar wird [2].

In dieser Arbeit beschäftigen wir uns mit ausgewählten, entscheidbaren Nichtstandard-Klassen der Prädikatenlogik erster Stufe. Im ersten Teil wird die Entscheidbarkeit der Klasse L_2 gezeigt. Das ist die einzige Klasse, die wir hier vorstellen, bei der Gleichheit erlaubt ist. Im zweiten Teil wird dann die Entscheidbarkeit für zwei Klassen ohne Gleichheit gezeigt, nämlich für die Klasse der Herbrand-Formeln und für die der Maslov-Formeln. Im Folgenden sei mit Prädikatenlogik implizit die Prädikatenlogik erster Stufe gemeint. Neben weiteren Einschränkungen, die im jeweiligen Kapitel genauer beschrieben werden, ist hier zur Orientierung der Symbolvorrat der jeweiligen Klassen aufgelistet.

Klasse	=	Relationen	Funktionen	Konstanten
L_2	Ja	Ja	Nein	Nein
Herbrand	Nein	Ja	Ja	Ja
Maslov	Nein	Ja	Nein	Ja

In dieser Arbeit interessieren wir uns nicht für die Komplexität der Entscheidungsalgorithmen, sodass diese Zwecks einfacherer Beweise nicht zwangsläufig optimal sind. Im Ausblick werden wir die Komplexitäten der Probleme jedoch nochmals aufgreifen.

Zum Ende dieser Arbeit wird das theoretisch vermittelte Wissen anwendbar gemacht. Dazu wurden die jeweiligen Entscheidungsalgorithmen in Python implementiert. Im letzten Kapitel gibt es eine Beschreibung dieser Implementierungen sowie konkrete Beispiele auf welche die Entscheidungsalgorithmen anwendbar sind und für welche diese liefern, ob die jeweilige Formel erfüllbar ist oder nicht.

Diese Arbeit ist eine detaillierte Aufbereitung einiger Aspekte des Kapitels *Other Decidable Cases* aus dem Buch *The Classical Decision Problem* [2] von Börger, Grädel und Gurevich. Wir schreiben *TCDP* als Abkürzung für das Buch *The Classical Decision Problem*.

2 Grundlagen

In diesem kurzen Kapitel führen wir Notationen ein und zeigen grundlegende Resultate, die über verschiedenste Kapitel hinweg benötigt werden.

2.1 Mathematische Grundlagen

Mit $\mathbb{N} = \{1, 2, \ldots\}$ bezeichnen wir die Menge der natürlichen Zahlen und mit $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$ die natürlichen Zahlen mit der 0.

Mit *id* ist die vom Kontext abhängige Identitätsfunktion gemeint. Ist zum Beispiel von $\mathbb{N}$ die Rede, dann ist $id\colon \mathbb{N} \mapsto \mathbb{N}, id(n) \mapsto n$.

Satz 2.1. *Ist $A = \{a_1, \ldots, a_n\}$ eine endliche, möglicherweise leere Menge mit n Elementen, dann ist $|\mathcal{P}(A)| = 2^n$.*

Beweis. Wir schreiben $\mathcal{P}(A)$ zunächst als disjunkte Vereinigung.

$$\mathcal{P}(\{a_1, \ldots, a_n\}) = \bigcup_{i=0}^{n} \{B \in \mathcal{P}(\{a_1, \ldots, a_n\}) : |B| = i\}$$

Daraus folgt dann direkt, dass

$$|\mathcal{P}(A)| = \sum_{i=0}^{n} \binom{n}{i} = \sum_{i=0}^{n} \binom{n}{i} 1^i 1^{n-i} = (1+1)^n = 2^n.$$

□

Satz 2.2. *Die Menge $\{0,1\}^{\mathbb{N}}$ aller unendlich langen $0,1$-Folgen ist überabzählbar.*

Beweis. Sei $m\colon \{0,1\} \mapsto \{0,1\}$ mit $m(0) = 1, m(1) = 0$ und ist $x \in \{0,1\}^{\mathbb{N}}$, dann bezeichnet x_i, das i-te Folgenglied. Wir nutzen nun ein Diagonalisierungsargument.

Angenommen $\{0,1\}^{\mathbb{N}}$ ist abzählbar, dann gibt es eine surjektive Funktion $f\colon \mathbb{N} \mapsto \{0,1\}^{\mathbb{N}}$. Wir definieren nun $x \in \{0,1\}^{\mathbb{N}}$ mit $x_i = m(f(i)_i)$. Dann ist $\forall n \in \mathbb{N}\colon x \neq f(x)$, da sich x nach Definition an mindestens einer Stelle unterscheidet. Damit ist aber $x \notin Im(f)$, also ist f nicht surjektiv, was ein Widerspruch ist. □

2.2 Prädikatenlogische Grundlagen

Diese Arbeit baut auf dem Skript der Veranstaltung *Logik und Formale* Systeme [11] auf, so dass hier dieselben Notationen wie dort verwendet werden. Außerdem werden alle dortigen Definitionen und Resultate als gegeben angesehen. Im Folgenden erinnern wir an besonders wichtige Resultate oder an Konventionen der Prädikatenlogik, die wir oft verwenden werden. Außerdem zeigen wir weitere, für diese Arbeit wichtige, Resultate, die nicht direkt im Logik Skript [11] vorkommen.

Wenn nicht explizit anders erwähnt, dann bezeichnen wir mit den Frakturbuchstaben $\mathfrak{A}, \mathfrak{B}, \mathfrak{C}, \ldots$ Strukturen. Die jeweiligen Buchstaben in normaler Schrift $A, B, C, \ldots$ bezeichnen dann die jeweiligen Universen.

Wie im Logik Skript [11] gehen wir davon aus, dass die Zeichen $\{\vee, \rightarrow, \leftrightarrow, \exists\}$ nicht Teil der Syntax der Prädikatenlogik sind, sondern lediglich Abkürzungen sind.

Ein aus dem Logik Skript [11] bekanntes Resultat ist das Koinzidenzlemma. Wir präsentieren hier eine etwas verallgemeinerte, aber im Kern gleiche, Version dieses Lemmas.

Lemma 2.3 (Koinzidenzlemma)**.** *Ist σ eine Signatur und ist σ' eine Einschränkung oder Erweiterung von σ oder $\sigma' = \sigma$ und ist φ eine σ, σ'-Formel und ist $\mathfrak{I}_1$ eine σ-Interpretation und $\mathfrak{I}_2$ eine σ'-Interpretation, deren Interpretationen von allen nichtlogischen Zeichen in φ übereinstimmen, dann gilt*

$$\mathfrak{I}_1 \models \varphi \iff \mathfrak{I}_2 \models \varphi.$$

Beweis. Durch Induktion über den Formelaufbau zu zeigen. □

Aus dem Koinzidenzlemma lässt sich direkt folgern, dass überflüssige Quantoren weggelassen werden können. Konkreter:

Lemma 2.4. *Ist φ eine Formel und ist x keine freie Variable in φ, dann gilt*

$$\forall x\varphi \equiv \varphi \equiv \exists x\varphi.$$

Beweis ist im Logik Skript [11] zu finden.

Im Großteil dieser Arbeit betrachten wir nur Sätze, das heißt Formeln ohne freie Variablen. Ob eine Interpretation $\mathfrak{I} = (\mathfrak{A}, \beta)$ einen Satz erfüllt, hängt dabei nicht von der Variablenbelegung β ab, sondern nur von $\mathfrak{A}$ (Koinzidenzlemma). Wir können also σ-Strukturen als Interpretationen von Sätzen ansehen. Das führt die folgende Definition ein.

Definition 2.5. *Ist σ eine Signatur, φ ein σ-Satz und $\mathfrak{A}$ eine σ-Struktur, dann schreiben wir $\mathfrak{A} \models \varphi$, falls für alle σ-Interpretationen $\mathfrak{I} = (\mathfrak{A}, \beta)$ gilt, dass $\mathfrak{I} \models \varphi$.*

Häufig kommt man in die Situation, dass eine Interpretation $\mathfrak{I}$ nur mit einer speziellen Belegung β eine Formel $\psi(x_1, \ldots, x_n)$ erfüllt. Damit wir nicht explizit $\beta(x_i) = a_i$ schreiben müssen, führen wir im Folgenden eine Notation ein, die dies erleichtert.

Definition 2.6. *Ist σ eine Signatur, $\psi(x_1, \ldots, x_n)$ eine σ-Formel, dann schreiben wir $\mathfrak{A} \models \psi[a_1, \ldots, a_n]$, wenn $\mathfrak{A}$ eine σ-Struktur ist und für die Interpretation $\mathfrak{I} = (\mathfrak{A}, \beta), \beta(x_i) = a_i$ für alle x_i, gilt $\mathfrak{I} \models \psi(x_1, \ldots, x_n)$.*

Es wird in mehreren Kapiteln die Rede von der Tiefe von Termen sein. Im Folgenden eine Definition dieses Begriffs.

Definition 2.7. *Ist σ eine Signatur und t ein σ-Term, dann ist die Tiefe d von t, $d(t) \coloneqq 0$, wenn t ein Konstantensymbol oder eine Variable ist. Ist $t = f(s_1, \ldots, s_n)$ für ein n-stelliges Funktionssymbol, dann ist $d(t) \coloneqq 1 + max\{d(s_1), \ldots, d(s_n)\}$.*

Folgendes Lemma aus einer Vorlesung von James Worrell [13] bezüglich Substitutionen ist ein Resultat, welches intuitiv klar ist und welches wir auch hin und wieder benutzen werden.

Lemma 2.8 (Übersetzungslemma)**.** *Ist σ eine Signatur, t ein σ-Term und φ eine σ-Formel, bei der keine Variable in t als gebundene Variable in φ vorkommt und ist $\mathfrak{I}$ eine Interpretation, dann gilt für eine beliebige Variable x*

$$\mathfrak{I} \models \varphi[x/t] \iff \mathfrak{I}_x^{t^{\mathfrak{I}}} \models \varphi.$$

Wurde von James Worrell bewiesen [13] .

In den späteren Entscheidungsverfahren wird der erste Schritt immer sein die freien Variablen aus einer Formel zu entfernen. Die Formelklassen, die in dieser Arbeit untersucht werden (L_2,Herbrand,Maslov) sind unter der Methodik, die nachfolgendes Lemma verwendet, nämlich das Hinzufügen von äußeren $\exists$-Quantoren, abgeschlossen.

Lemma 2.9. *Sei σ eine Signatur. Es gibt eine berechenbare Funktion*

$$\mathit{removeFreeVariables}\colon \mathit{Form}_\sigma \mapsto \mathit{Form}_\sigma,$$

die zu jeder σ-Formel φ eine σ-Formel $\psi = \mathit{removeFreeVariables}(\varphi)$ liefert, sodass ψ, φ erfüllbarkeitsäquivalent sind und ψ keine freien Variablen enthält.

Beweis. Hat φ die freien Variablen $\{x_1, \ldots, x_n\}$, dann ist $\psi = \exists x_1 \ldots \exists x_n \varphi$ trivialerweise erfüllbarkeitsäquivalent zu φ und da für den Algorithmus nur die freien Variablen von φ bestimmt werden müssen und für ψ nur Existenzquantoren hinzugefügt werden, ist *removeFreeVariables* berechenbar. □

Noch eine abschließende Bemerkung zu dem Gebrauch von Signaturen und dem damit verbundenen Symbolvorrat: Häufig werden wir entweder fordern, dass eine Signatur abzählbar unendlich viele Symbole eines Symboltyps hat (Relationen, Funktionen, Konstanten). Das wird in dem Kontext meist dafür benötigt, dass Algorithmen, die auf Formeln über einer solchen Signatur operieren, einen unendlichen Symbolvorrat benötigten, um z.B. einer Formel beliebig viele Primformeln mit verschiedenen Relationssymbolen hinzufügen zu können. Manchmal schränken wir aber auch die Signatur ein und fordern einen endlichen Symbolvorrat. Das hat in dem Kontext meist den Sinn, dass gewisse Mengen von gewissen Objekten, die in Abhängigkeit solch einer Signatur definiert sind, auch endlich sind.

Wichtig ist hierbei aber zu bemerken, dass nach dem Koinzidenzlemma 2.3 der Wahrheitswert einer Formel nur von den in der Formel vorkommenden Symbolen abhängig ist. Ob man nun $\varphi = R(x)$ über der Signatur $\sigma = (R)$ oder über einer Signatur σ' (die auch R enthält) mit unendlich vielen Relationssymbolen betrachtet, ist bezüglich der Semantik also irrelevant. Deshalb macht es in den späteren Entscheidungsalgorithmen keinen Unterschied, ob wir eine dieser Bedingungen an die Signatur gestellt haben, so dass mit dieser Signatur einfacher zu arbeiten ist. Außerdem wird nicht die Allgemeinheit eingeschränkt. Die Bedingungen werden so gewählt sein, dass es für jede Formel φ aus einer der behandelten Klassen, über einer Signatur σ, eine Signatur σ' gibt, die einerseits den Bedingungen genügt und σ' ist entweder gleich, eine Erweiterung oder eine Einschränkung von σ und φ ist eine σ'-Formel.

3 Entscheidbarkeit von L_2

Ziel dieses Kapitels ist es zu zeigen, dass die Klasse L_2 der relationalen Formeln mit zwei Variablen entscheidbar ist. Mit relationalen Formeln sind prädikatenlogische Formeln gemeint, die keine Konstantensymbole oder Funktionssymbole enthalten. Wir wollen also zeigen, dass $Sat(L_2)$ entscheidbar ist.

Das Kapitel ist in vier Teile gegliedert. Im ersten Teil zeigen wir, dass sich Formeln aus L_2 in eine gewisse Normalform bringen lassen. Im zweiten Teil zeigen wir, dass diese Normalform (und als Folgerung auch L_2) eine endliche Modelleigenschaft besitzt und geben eine explizite obere Schranke für die Modellgröße in Abhängigkeit von der Formellänge an. Im dritten Teil zeigen wir, dass wir algorithmisch überprüfen können, ob eine endliche Interpretation eine Formel erfüllt. Im letzten Teil wird aus der endlichen Modelleigenschaft, der oberen Schranke und der Überprüfbarkeit von Modellen die Entscheidbarkeit von $Sat(L_2)$ gefolgert.

Dieses Kapitel basiert auf *TCDP* [2], einem Artikel [4] von Grädel, Kolaitis und Vardi und auf einem Artikel [5] von Grädel und Otto.

Definition 3.1 (L_k-Formeln)**.** *Ist σ eine Signatur, dann bezeichnen wir die Menge aller σ-Formeln, die maximal k verschiedene Variablen und keine Konstantensymbole und keine Funktionssymbole enthalten, mit $L_k(\sigma)$. Ist σ aus dem Kontext ersichtlich, schreiben wir auch einfach nur L_k. Mit $Sen(L_k) \subseteq L_k$ bezeichnen wir die Menge aller σ-Sätze aus L_k.*

Im Folgenden beschäftigen wir uns ausschließlich mit L_2. O.B.d.A. seien die einzigen Variablen, die in L_2-Formeln vorkommen x und y.

Beispiel 3.2

Dieses Beispiel ist direkt aus *TCDP* [2] übernommen. Sei $\sigma_{GR} = (E)$ die Signatur von Graphen, wobei E ein binäres Relationssymbol ist. Folgender Satz $\varphi \in L_2$, drückt aus, dass es einen Weg im Graphen der Länge 4 gibt.

$$\varphi = \exists x \exists y (E(x,y) \wedge \exists x (E(y,x) \wedge \exists y (E(x,y) \wedge \exists x E(y,x))))$$

Man bemerke aber, dass dieser Satz nicht fordert, dass der Weg kanteneinfach sein muss. Das können wir in L_2 nicht formulieren.

Folgender Satz $\psi \in L_2$ drückt aus, dass jeder Knoten mit einer eingehenden Kante auch eine ausgehende Kante hat.

$$\psi = \forall x (\exists y (E(y,x)) \rightarrow \exists y (E(x,y)))$$

Nun zwei Beispiele für Formeln, die nicht in L_2 liegen. Sei dazu $\sigma = (R; f; c)$.

- $\forall x \forall y \forall z (R(x) \wedge \neg R(y) \wedge R(z)) \notin L_2$
- $\forall x R(f(x,c)) \notin L_2$

3.1 Entfernung der freien Variablen

In Lemma 2.9 wurde gezeigt, dass sich jede Formel φ durch Hinzufügen von äußeren $\exists$-Quantoren auf eine erfüllbarkeitsäquivalente Formel $\psi = \mathit{removeFreeVariables}(\varphi)$ reduzieren lässt. Hierdurch kommen aber keine neuen Variablen oder Konstantensymbole oder Funktionssymbole hinzu. Deshalb ist auch $\psi \in L_2$, wenn $\varphi \in L_2$. Es reicht daher aus nur Elemente aus $\mathit{Sen}(L_2)$ zu betrachten.

3.2 Reduktion auf Scott-Formeln

Ziel dieses Abschnitts ist es zu zeigen, wie beliebige Formeln aus $\mathit{Sen}(L_2)$ in eine Normalform gebracht werden können, die erfüllbarkeitsäquivalent zur Ausgangsformel ist. Dabei wird zunächst die Normalform definiert, danach der Reduktionsalgorithmus beschrieben und mit Pseudocode zusammengefasst dargestellt und abschließend wird seine Korrektheit bewiesen.

Für diesen Abschnitt bezeichne σ eine Signatur mit abzählbar unendlich vielen Relationssymbolen. Wir verlangen hier einen unendlichen Relationssymbolvorrat, da im späteren Reduktionsalgorithmus Formeln um beliebig viele Relationssymbole erweitert werden können müssen.

Definition 3.3 (Scott-Formeln)**.** *Die Menge $\Psi_{Scott} \subseteq \mathit{Sen}(L_2)$, enthält alle Formeln folgender Form:*

$$\forall x \forall y \alpha \wedge \bigwedge_{i=1}^{m} \forall x \exists y \beta_i, \ \alpha \text{ und } \beta_i \text{ sind quantorenfrei}$$

Solche Formeln nennen wir Scott-Formeln.

Satz 3.4. *Es gibt eine berechenbare Funktion $\Gamma\colon \mathit{Sen}(L_2) \mapsto \Psi_{Scott}$, sodass für alle $\psi \in \mathit{Sen}(L_2)$ gilt:*

1. $\psi \in \mathit{Sat}(L_2) \iff \Gamma(\psi) \in \mathit{Sat}(L_2)$.
2. *Für alle σ-Strukturen $\mathfrak{A}$:* $\mathfrak{A} \models \Gamma(\psi) \implies \mathfrak{A} \models \psi$.

Beweisidee: Wir ersetzen schrittweise eine Teilformel mit Quantoren in ψ durch Formeln ohne Quantoren, bis ψ quantorenfrei ist. Bei jedem Ersetzungsschritt sammeln wir Hilfsformeln, sodass am Ende das quantorenfreie ψ konjunktiert mit den gesammelten Hilsformeln, erfüllbarkeitsäquivalent zum ursprünglichen ψ ist.

Wir beschreiben im Beweis zunächst einen Algorithmus (Γ soll die Funktion sein, die von dem Algorithmus berechnet wird) und geben erste erläuternde Begründungen zu den Schritten an. Am Ende zeigen wir, dass die vom Algorithmus berechnete Funktion Γ die Eigenschaften (1) und (2) hat. Möchte man zuerst das Verfahren in Aktion sehen, kann man sich direkt das Beispiel 2 anschauen. Hier wird das Verfahren angewendet.

Beweis. Wir betrachten zunächst die am tiefsten verschachtelten Quantoren von ψ. Gemeint sind damit Teilformeln von ψ der Form $Qv\eta, Q \in \{\forall, \exists\}, v \in \{x, y\}, \eta$ ist quantorenfrei. Sei R nun ein einstelliges Relationssymbol, welches nicht in ψ vorkommt. Nun bezeichnen wir mit ψ', die Formel die entsteht, wenn jede Teilformel $Qv\eta$ in ψ durch $R(w)$ ersetzt wird. Wobei

$$w = \begin{cases} y, & \text{wenn } v = x \\ x, & \text{wenn } v = y \end{cases}$$

ψ' ist nicht direkt erfüllbarkeitsäquivalent zu ψ. Wir erstellen daher einen Satz θ, sodass $\psi' \wedge \theta$ erfüllbarkeitsäquivalent zu ψ ist. θ wird erst am Ende des Algorithmus zu ψ konjunktiert. θ soll sicherstellen, dass sich $R(w)$ semantisch äquivalent zu $Qv\eta$ verhält.

$$\theta \coloneqq \forall w(R(w) \leftrightarrow Qv\eta)$$

Zwar hat ψ' mindestens einen Quantor weniger als ψ, jedoch ist $\psi' \wedge \theta$ nicht unbedingt näher an einer Formel aus Ψ_{Scott} dran, da θ keine Scott-Formel ist. Es gibt allerdings ein θ_{Scott}, sodass $\theta_{Scott} \equiv \theta$ und $\theta_{Scott} \in \Psi_{Scott}$. Wir unterscheiden hierzu 2 Fälle.

1. $Q = \exists$:

$$\begin{aligned}\theta &= \forall w(R(w) \leftrightarrow \exists v\eta) \\ &\equiv \forall w(R(w) \rightarrow \exists v\eta) \wedge \forall w(\exists v\eta \rightarrow R(w)) \\ &\equiv \forall w \exists v(R(w) \rightarrow \eta) \wedge \forall w \forall v(\eta \rightarrow R(w)) \\ &\equiv \forall w \forall v(\eta \rightarrow R(w)) \wedge \forall w \exists v(R(w) \rightarrow \eta) = \theta_{Scott} \in \Psi_{Scott}\end{aligned}$$

2. $Q = \forall$:

$$\begin{aligned}\theta &= \forall w(R(w) \leftrightarrow \forall v\eta) \\ &\equiv \forall w(R(w) \rightarrow \forall v\eta) \wedge \forall w(\forall v\eta \rightarrow R(w)) \\ &\equiv \forall w \forall v(R(w) \rightarrow \eta) \wedge \forall w \exists v(\eta \rightarrow R(w)) = \theta_{Scott} \in \Psi_{Scott}\end{aligned}$$

Wir können nun $\psi \coloneqq \psi'$ setzen und die oben beschriebene Ersetzung wiederholen, bis ψ quantorenfrei ist. Seien θ_i, die durch den i-ten Ersetzungsschritt erzeugten Formeln zur Erfüllbarkeitsäquivalenzerhaltung und $\theta_{Scott,i}$ die zu θ_i semantisch äquivalenten Scott-Formeln. Da ψ quantorenfrei ist, kann ψ bis zu zwei freie Variablen enthalten. Wir ersetzen diese dann durch ihre universellen Quantoren wie folgt:

$$\varphi' \coloneqq \forall x \forall y \psi \wedge \bigwedge_{i=1}^{m} \theta_{Scott,i}.$$

φ' genügt noch nicht ganz einer Scott-Formel. Wir können jedoch alle Konjunktionen von $\forall\forall$-Formeln in φ' zu einer $\forall\forall$-Formel zusammenfassen und danach die Konjunktionen so umordnen, dass die $\forall\forall$-Formel vor den $\forall\exists$-Formeln steht. Die so entstandene Formel nennen wir φ und $\varphi \in \Psi_{Scott}$.

Im Folgenden ist der beschriebene Algorithmus noch einmal als Pseudocode angegeben.

Algorithmus 1: Γ_{Alg}

Eingabe : $\langle\psi\rangle, \psi \in Sen(L_2)$
Ausgabe: $\langle\varphi\rangle$, $\varphi \in \Psi_{Scott}$, φ genügt (1) und (2)

$i \leftarrow 0$
while ψ *ist nicht quantorenfrei* **do**
  $i \leftarrow i + 1$
  Wähle Teilformel von ψ der Form $Qv\eta$ wobei $Q \in \{\forall, \exists\}$, $v \in \{x, y\}$, η ist quantorenfrei
  if $v = x$ **then**
    $w \leftarrow y$
  else
    $w \leftarrow x$
  $R \leftarrow$ einstelliges Relationssymbol, das nicht in ψ vorkommt
  if $Q = \exists$ **then**
    $\theta_{Scott,i} \leftarrow \forall w \forall v(\eta \rightarrow R(w)) \wedge \forall w \exists v(R(w) \rightarrow \eta)$
  else
    $\theta_{Scott,i} \leftarrow \forall w \forall v(R(w) \rightarrow \eta) \wedge \forall w \exists v(\eta \rightarrow R(w))$
  Ersetze alle Teilformeln $Qv\eta$ von ψ durch $R(w)$
$\varphi \leftarrow \forall x \forall y \psi \wedge \bigwedge_{k=1}^{i} \theta_{Scott,k}$
Fasse die $\forall\forall$-Teilformeln von φ zusammen und wähle Reihenfolge der Konjunktionen so, dass φ eine Scott-Formel ist
return $\langle\varphi\rangle$

Wir zeigen nun, dass die von Γ_{Alg} berechnete Funktion Γ den Anforderungen (1) und (2) genügt.

Sei $\psi \in Sen(L_2)$ und $\varphi = \Gamma(\psi)$. Seien dazu $\psi^{(i)}$, die ψ's, die im i-ten Schleifendurchlauf von $\Gamma_{Alg}(\langle\psi\rangle)$ entstanden sind und sei $\psi^{(0)} := \psi$. Seien außerdem $R^{(i)}$ die einstelligen Relationen, die im i-ten Schritt produziert wurden und $F^{(i)}$, die Teilformeln, die in $\psi^{(i-1)}$ durch $R^{(i)}$ ersetzt wurden. Zuletzt sei noch $n \in \mathbb{N}_0$ die Anzahl der Ersetzungschritte. Es gilt also, dass $\psi^{(n)}$ quantorenfrei ist.

Durch die Wahl der $\theta_{Scott,i+1}$ sind $R^{(i+1)}$ und $F^{(i+1)}$ semantisch äquivalent, solange $\theta_{Scott,i+1}$ gilt. Konkreter: Ist $\mathfrak{I}$ eine σ-Interpretation dann gilt

$$\mathfrak{I} \models \theta_{Scott,i+1} \implies (\mathfrak{I} \models R^{(i+1)} \iff \mathfrak{I} \models F^{(i+1)}).$$

Da $\psi^{(i+1)}$ gerade dadurch entstanden ist, dass alle $F^{(i+1)}$ in $\psi^{(i)}$ durch $R^{(i+1)}$ ersetzt wurden, folgt sofort, dass

$$\mathfrak{I} \models \theta_{Scott,i+1} \implies (\mathfrak{I} \models \psi^{(i)} \iff \mathfrak{I} \models \psi^{(i+1)}).$$

Hieraus folgt dann direkt die Begründung, wieso die $\forall$-Quantisierung der freien Variablen im vorletzten Schritt von Γ, semantische Äquivalenz erhält. Denn es gilt:

$$\psi^{(n)} \wedge \bigwedge_{i=1}^{n} \theta_{Scott,i} \equiv \psi^{(0)} \wedge \bigwedge_{i=1}^{n} \theta_{Scott,i} \equiv \forall x \forall y \psi^{(0)} \wedge \bigwedge_{i=1}^{n} \theta_{Scott,i} \equiv \forall x \forall y \psi^{(n)} \wedge \bigwedge_{i=1}^{n} \theta_{Scott,i}.$$

Wir können nun zeigen, dass Γ die Eigenschaft (2) besitzt. Sei $\mathfrak{I}$ eine σ-Interpretation. Wir zeigen, dass wenn $\mathfrak{I}$ ein Modell von φ ist, dann ist $\mathfrak{I}$ auch ein Modell von ψ.

$$\begin{aligned} & \mathfrak{I} \models \varphi \\ \iff & \mathfrak{I} \models \psi^{(0)} \wedge \bigwedge_{k=1}^{n} \theta_{Scott,k} \\ \implies & \mathfrak{I} \models \psi^{(0)} = \psi \end{aligned}$$

Für Bedingung (1) müssen wir noch zeigen, dass $\mathfrak{I} \models \psi \implies \exists \mathfrak{I}' : \mathfrak{I}' \models \varphi$. Sei $\mathfrak{I}$ eine beliebige σ-Interpretation mit $\mathfrak{I} \models \psi$. Wir definieren uns dazu eine rekursiv definierte Folge von σ-Interpretationen $\mathfrak{I}^{(i)}$. Wir setzen $\mathfrak{I}^{(0)} \coloneqq \mathfrak{I}$ und $\mathfrak{I}^{(i+1)} \coloneqq \mathfrak{I}^{(i)}$, aber so, dass $r \in (R^{(i+1)})^{\mathfrak{I}^{(i+1)}} \iff (\mathfrak{I}^{(i)})^r_w \models F^{(i+1)}$. Hier ist $w \in \{x, y\}$ die Variable, die nicht gebunden in $F^{(i+1)}$ vorkommt. Nach Konstruktion gilt nun einerseits, dass $\mathfrak{I}^{(n)} \models \theta_{Scott,i}$ für alle i und nach dem Koinzidenzlemma gilt auch $\mathfrak{I}^{(n)} \models \psi$. Also gilt insgesamt:

$$\begin{aligned} & \mathfrak{I}^{(n)} \models \psi \wedge \mathfrak{I}^{(n)} \models \theta_{Scott,1} \wedge \ldots \wedge \mathfrak{I}^{(n)} \models \theta_{Scott,n} \\ \iff & \mathfrak{I}^{(n)} \models \psi^{(0)} \wedge \bigwedge_{k=1}^{n} \theta_{Scott,k} \\ \iff & \mathfrak{I}^{(n)} \models \varphi. \end{aligned}$$

Damit existiert zu jedem Modell $\mathfrak{I}$ von ψ ein Modell $\mathfrak{I}' = \mathfrak{I}^{(n)}$, sodass $\mathfrak{I}' \models \varphi$. Damit erfüllt Γ auch die Bedingung (1) und wir sind fertig.

□

Beispiel 3.5

Wir zeigen abschließend den Algorithmus anhand eines Beispiels. Seien dazu H, L Relationssymbole mit Stelligkeiten 1 und 2. Wir wandeln nun die Scott-Reduktion auf die Formel ψ an.

$$\psi \coloneqq \forall x(H(x) \wedge \forall y(H(y) \wedge \neg L(x, y))).$$

ψ ist nicht quantorenfrei, daher ersetzen wir $\forall y(H(y) \wedge \neg L(x, y))$ mit $R_1(x)$ und erhalten

$$\psi' \coloneqq \forall x(H(x) \wedge R_1(x)), \theta_1 \coloneqq \forall x(R_1(x) \leftrightarrow \forall y(H(y) \wedge \neg L(x, y))).$$

$\psi \coloneqq \psi'.\psi$ ist immer noch nicht quantorenfrei. Wir ersetzen diesmal $\forall x(H(x) \wedge R_1(x))$ durch $R_2(y)$ und erhalten

$$\psi' \coloneqq R_2(y), \theta_2 \coloneqq \forall y(R_2(y) \leftrightarrow \forall x(H(x) \wedge (R_1(x))).$$

$\psi \coloneqq \psi'$. ψ ist quantorenfrei, daher erstellen wir

$$\begin{aligned} \varphi &\coloneqq \forall x \forall y R_2(y) \wedge \theta_{Scott,1} \wedge \theta_{Scott,2}, \\ \theta_{Scott,1} &= \forall x \forall y(R_1(x) \rightarrow H(y) \wedge \neg L(x, y)) \wedge \forall x \exists y(H(y) \wedge \neg L(x, y) \rightarrow R_1(x)), \\ \theta_{Scott,2} &= \forall x \forall y(R_2(x) \rightarrow H(y) \wedge R_1(y)) \wedge \forall x \exists y(H(y) \wedge R_1(y) \rightarrow R_2(x)). \end{aligned}$$

Im letzten Schritt sortieren wir um und fassen die $\forall\forall$-Formeln zusammen und erhalten als Ergebnis

$$\begin{aligned}
&\varphi \coloneqq \forall x \forall y \alpha \wedge \forall x \exists y \beta_1 \wedge \forall x \exists y \beta_2, \\
&\alpha = R_2(y) \wedge (R_1(x) \rightarrow H(y) \wedge \neg L(x,y)) \wedge (R_2(x) \rightarrow H(y) \wedge R_1(y)), \\
&\beta_1 = H(y) \wedge \neg L(x,y) \rightarrow R_1(x), \\
&\beta_2 = H(y) \wedge R_1(y) \rightarrow R_2(x).
\end{aligned}$$

3.3 Endliche Modelleigenschaft der Scott-Formeln

In diesem Abschnitt zeigen wir, dass Ψ_{Scott} die endliche Modelleigenschaft besitzt und folgern daraus diese auch für $Sen(L_2)$. Wir bestimmen zusätzlich eine obere Schranke für Modelle in Abhängigkeit der Formellänge.

Sei im folgenden Abschnitt I_R eine endliche Indexmenge und $\sigma = ((R_i)_{i \in I_R})$ eine Signatur. $s\colon I_R \mapsto \mathbb{N}$ gibt die Stelligkeiten der Relationssymbole von σ an.

Definition 3.6 (Endliche Modelleigenschaft)**.** *Eine Klasse von prädikatenlogischen Formeln L besitzt die endliche Modelleigenschaft genau dann, wenn für jede Formel $\varphi \in Sat(L)$ gilt, dass φ ein endliches Modell hat. Ein Modell ist endlich, wenn sein Universum endlich ist.*

Wir wollen später konstruktiv zeigen, dass Ψ_{Scott} die endliche Modelleigenschaft besitzt. Für eine erfüllbare Scott-Formel werden wir ein endliches Modell konstruieren, indem wir zuerst das Universum angeben. Danach müssen wir die Interpretation der Relationen festlegen. Für Primformeln von $\psi \in \Psi_{Scott}$ der Form $R(x_1, \ldots, x_n)$ für ein n-stelliges Relationssymbol R, sind die Argumente alle entweder x oder y, also $\forall i : x_i \in \{x, y\}$. Es genügt also für jede Relation in unserem Modell nur festzulegen wie alle Paare aus A in Relation zueinander stehen. Dies werden wir in der Beweisführung später durch sogenannte 2-Tables angeben. Daher führen wir im Folgenden k-Tables ein, um das oben beschriebene Konzept zu präzisieren.

Definition 3.7. *Ist $\mathfrak{A}$ eine σ-Struktur mit Universum A und Relationen $R_i^{\mathfrak{A}}, i \in I_R$ und sind $a_1, \ldots, a_k \in A$ beliebig, dann bezeichne $\mathfrak{A}\big|_{\{a_1,\ldots,a_k\}} \coloneqq (\{a_1, \ldots, a_k\}; (R_i')_{i \in I_R})$ die Einschränkung von $\mathfrak{A}$ auf die Elemente $\{a_1, \ldots, a_k\}$, wobei $R_i' \subseteq \{a_1, \ldots, a_k\}^{s(i)}$ und $(x_1, \ldots, x_{s(i)}) \in R_i' \iff (x_1, \ldots, x_{s(i)}) \in R_i^{\mathfrak{A}}$.*

Definition 3.8 (k-Table)**.** *Sei $k \in \mathbb{N}$. Ein k-Table ist eine σ-Struktur, mit dem Universum $\{1, \ldots, k\}$. Ist $\mathfrak{A}$ eine σ-Struktur mit Universum A und sind $a_1, \ldots, a_k \in A$ paarweise verschieden, dann bezeichnet $T_{\mathfrak{A}}[a_1, \ldots, a_k]$ den k-Table $\mathfrak{B}$ für den die Abbildung $a_1 \mapsto 1, \ldots, a_k \mapsto k$ ein Isomorphismus von $\mathfrak{A}\big|_{\{a_1,\ldots,a_k\}}$ nach $\mathfrak{B}$ ist.*

Ein k-Table von $\mathfrak{A}$ zu einer Menge von Elementen aus dem Universum von $\mathfrak{A}$ ist informell gesprochen also nichts anderes, als die Einschränkung von $\mathfrak{A}$ auf diese Elemente zusammen mit einer standardisierten Umbenennung dieser. Daher speichert der k-Table kompakt sämtliche Informationen darüber wie diese Elemente in $\mathfrak{A}$ in Relation zueinander stehen. Außerdem ist es durch die Umbenennung leicht, verschiedene k-Tables miteinander zu vergleichen.

Beispiel 3.9

Sei $\sigma = (R_1, R_2)$ und $\mathfrak{A} = (A; R_1^{\mathfrak{A}}, R_2^{\mathfrak{A}})$ eine σ-Struktur mit $A = \{x, y, z\}, R_1^{\mathfrak{A}} = \{(x), (z)\}, R_2^{\mathfrak{A}} = \{(y, x), (z, y)\}$, dann ist $\mathfrak{A}\big|_{\{x,y\}} = (\{x, y\}; \{(x)\}, \{(y, x)\})$ und $T_{\mathfrak{A}}[x, y] = (\{1, 2\}; \{(1)\}, \{(2, 1)\})$ und $T_{\mathfrak{A}}[y, x] = (\{1, 2\}; \{(2)\}, \{(1, 2)\})$.

Wir stellen folgende allgemeine Beobachtungen über k-Tables fest:

Lemma 3.10. *Für ein fixes k ist die Menge aller k-Tables von einer σ-Struktur $\mathfrak{A}$ endlich.*

Beweis. Das Universum ist für alle k-Tables gleich und besteht aus k Elementen. Ist R ein n-stelliges Relationssymbol, dann gibt es $c(R) \coloneqq |\mathcal{P}(\{1, \ldots, k\}^n)| = 2^{k^n}$ Interpretationen dieser Relation bei einem Universum mit k Elementen. Die Anzahl der k-Tables ist dann durch $\prod_{i \in I_R} c(R_i) < \infty$ gegeben. □

Durch einen k-Table sind auch automatisch alle l-Tables, $l < k$, die durch dieselben Elemente und deren Permutationen induziert werden, bestimmt. Genauer:

Lemma 3.11. *Sei $\mathfrak{A}$ eine σ-Struktur und sind $a_1, \ldots, a_k$ paarweise verschieden und $\mathfrak{B} \coloneqq T_{\mathfrak{A}}[a_1, \ldots, a_k]$ und $l < k$. Sind $a_{i_1}, \ldots, a_{i_l}$ paarweise verschieden, dann gilt $T_{\mathfrak{A}}[a_{i_1}, \ldots, a_{i_l}] = T_{\mathfrak{B}}[i_1, \ldots, i_l]$.*

Ohne Beweis.

Sei $\varphi \in \Psi_{Scott}$. Um eine σ-Struktur $\mathfrak{A}$ hinsichtlich der Frage, ob $\mathfrak{A} \models \varphi$ zu definieren reicht es aus, das Universum A anzugeben und danach für alle Paare aus A deren 2-Tables (und damit auch indirekt deren 1-Tables) auf konsistente Art und Weise anzugeben. Denn ist $|A| \geq 3$ und gibt es ein n-stelliges Relationssymbol R in σ, $n \geq 3$, dann hat die Festlegung ob $(x_1, \ldots, x_n) \in R^{\mathfrak{A}}$ ist, keinen Einfluss auf $\mathfrak{A} \models \varphi$, insofern $\{x_1, \ldots, x_n\} \geq 3$, da es keine Teilformeln in φ mit mehr als 2 Variablen gibt.

Mit konsistenter Angabe der 2-Tables ist gemeint, dass zwei verschiedene 2-Tables, T_1, T_2, induziert durch a_1, a_2 und a_3, a_4, den gleichen 1-Table für ein $a \in A$ festlegen, wenn $a \in \{a_1, a_2\} \cap \{a_3, a_4\}$. Auch müssen 2-Tables, die durch die gleichen Elemente induziert wurden, nur in unterschiedlicher Reihenfolge, isomorph zueinander sein.

Für den späteren Beweis werden wir für jedes Modell $\mathfrak{A}$ von φ ein endliches Modell $\mathfrak{D}$, über Angabe des Universums und der 2-Tables, konstruieren. Dadurch, dass die Gleichheitsrelation in $Sen(L_2)$ erlaubt ist, kann es passieren, dass φ fordert, dass gewisse 1-Tables einzigartig sind (Einzigartig soll hier bedeuten, dass ein 1-Table von exakt einem Element aus A induziert wird). Folgender Satz illustriert dies:

$$\varphi = \forall x \forall y (P(x) \wedge P(y) \rightarrow x = y).$$

Bei der Konstruktion von $\mathfrak{D}$ müssen wir also darauf achten, dass einzigartige 1-Tables von $\mathfrak{A}$, auch in $\mathfrak{D}$ einzigartig sind. Wir behandeln diese gesondert und bezeichnen Elemente, deren 1-Table einzigartig ist, als Könige.

Definition 3.12 (Könige)**.** *Ist $\mathfrak{A}$ eine σ-Struktur mit Universum A und $a \in A$, dann ist a ein König, wenn der durch a realisierte k-Table einzigartig ist, also wenn $\forall b \neq a \colon T_{\mathfrak{A}}[a] \neq T_{\mathfrak{A}}[b]$.*

Wir können nun mit diesen Voraussetzungen die endliche Modelleigenschaft von Ψ_{Scott} zeigen und zusätzlich eine obere Schranke für Modelle angeben.

Satz 3.13. *Ψ_{Scott} besitzt die endliche Modelleigenschaft.*

Beweis. Sei $\varphi \in \Psi_{Scott}$ und $\mathfrak{A}$ eine σ-Struktur mit Universum A und $\mathfrak{A} \models \varphi$. Sei

$$\varphi =: \forall x \forall y \alpha \wedge \bigwedge_{i=1}^{m} \forall x \exists y \beta_i, \text{ wobei } \alpha \text{ und } \beta_i \text{ quantorenfrei sind.}$$

Wir können ohne Beschränkung der Allgemeinheit davon ausgehen, dass $\beta_i(x, y) \models x \neq y$ für $i = 1, \ldots, m$. Das liegt daran, dass für ein fixes i folgende semantische Äquivalenz gilt:

$$\forall x \exists y \beta_i(x, y) \equiv \forall x \exists y (x \neq y \wedge (\beta_i(x, x) \vee \beta_i(x, y))).$$

Wir führen zunächst ein paar Hilfsdefinitionen in Abhängigkeit von $\mathfrak{A}$ und φ ein, auf welche wir im Verlaufe des Beweises zurückgreifen werden.

- $P = \{T_{\mathfrak{A}}[a] \colon a \in A\}$ ist die Menge der in $\mathfrak{A}$ realisierten 1-Tables
- K ist die Menge der Könige in $\mathfrak{A}$
- für $i = 1, \ldots, m$ sind $f_i \colon A \mapsto A$ Funktionen, sodass für alle $a \in A$ gilt, dass $(\mathfrak{A}, \gamma_a) \models \beta_i(x, y)$, wobei $\gamma_a(x) \mapsto a, \gamma_a(y) \mapsto f_i(a)$
- $C \coloneqq K \cup \{f_i(k) : k \in K, i = 1, \ldots, m\}, \mathfrak{C} \coloneqq \mathfrak{A}\big|_C$, nennen wir den *Hofstaat* von $\mathfrak{A}$
- $Q \subseteq P$ ist die Menge aller 1-Tables, die durch Könige induziert werden

Wir beginnen nun damit, ein endliches Modell $\mathfrak{D}$ für φ zu konstruieren. Wir machen uns zu Nutze, dass es nur endlich viele realisierte 1-Tables in $\mathfrak{A}$ gibt. Das Universum D von $\mathfrak{D}$ setzen wir wie folgt:

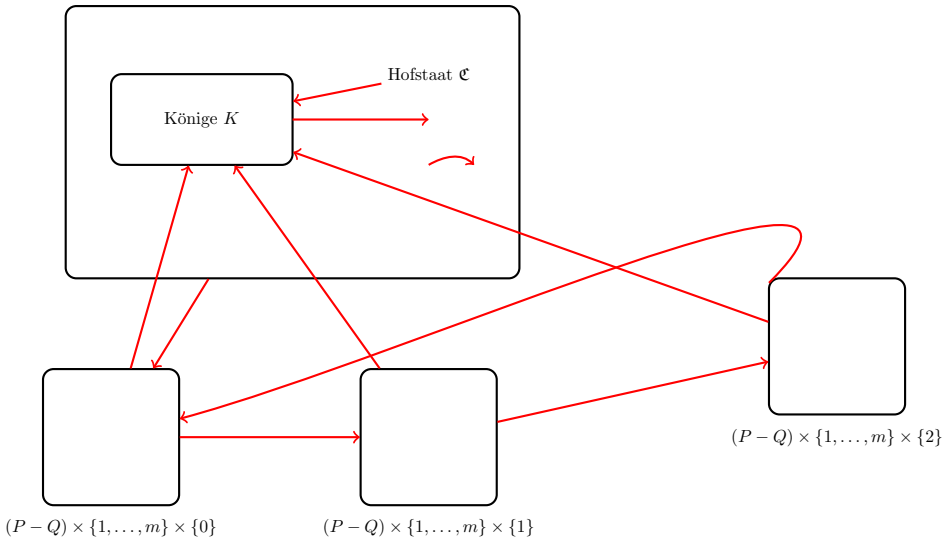
$$D \coloneqq C \cup ((P - Q) \times \{1, \ldots, m\} \times \{0, 1, 2\}).$$

Wir werden im Folgenden die Relationen von $\mathfrak{D}$ festlegen, indem wir die 2-Tables von allen Elementpaaren $a, b \in D, a \neq b$ aus $\mathfrak{D}$ festlegen. Wir sagen b ist ein Zeuge für a bezüglich β_i, falls $(\mathfrak{D}, \gamma) \models \beta_i(x, y)$ gilt, wobei $\gamma(x) = a$ und $\gamma(y) = b$. Wir müssen die Relationen also so festlegen, dass jedes $d \in D$ bezüglich allen $\beta_i, i = 1, \ldots, m$ mindestens einen Zeugen hat. Die verwendeten Metaphern helfen dabei den Beweis gleich besser zu verstehen. Die Könige kriegen eine Sonderbehandlung, indem diese ihre Zeugen ausschließlich aus dem Hofstaat beziehen.

Möchten wir beispielsweise, dass b als Zeuge für a bezüglich β_i dient, dann können wir $T_{\mathfrak{D}}[a, b] \coloneqq T_{\mathfrak{A}}[a', f_i(a')]$ für ein beliebiges $a' \in A$ setzen. Da $\mathfrak{A}$ ein Modell von φ ist, erfüllen deshalb auch a, b, die $\forall\forall$-quantifizierte Teilformel α. Die Schwierigkeit, die dabei jetzt entsteht ist, dass wir nun für b nicht a als Zeugen verwenden dürfen, da wir sonst den 2-Table doppelt definieren würden und es so zu Inkonsistenzen kommen kann.

Die folgende Abbildung zeigt, wer wem im folgenden Beweis als Zeugen dienen wird. Ein Pfeil von Menge A nach B bedeutet, dass ein Element aus A ein Element aus

B als Zeuge nutzen kann. Man bemerke, dass es im Hofstaat zwar passieren könnte, dass zwei Elemente sich gegenseitig als Zeugen benutzen, dies ist aber kein Widerspruch, da der Hofstaat $\mathfrak{C}$ bereits konsistent ist.



Wir konstruieren nun $\mathfrak{D}$ vollständig in 5 Schritten:

1. $\mathfrak{D}$ soll eine Erweiterung von $\mathfrak{C}$ sein, also $\mathfrak{D}_{|C} \coloneqq \mathfrak{C}$. Hiermit sind auch die 2-Tables und 1-Tables aller Elemente aus $\mathfrak{C}$ definiert.

2. Die restlichen 1-Tables für Elemente $b = (\mathfrak{B}, i, j) \in D - C$ setzen wir wie folgt: $T_{\mathfrak{D}}[b] \coloneqq \mathfrak{B}$. Diese 1-Tables wären implizit auch durch die folgenden Schritte mit definiert worden, erleichtern aber das Verständnis.

3. Wir stellen nun sicher, dass jedes Element $d \in D$ für jedes β_i einen Zeugen hat, indem wir die restlichen 2-Tables definieren.
   a) $d \in K$: $f(d) \in \mathfrak{C}$, $T_{\mathfrak{D}}[d, f(d)]$ ist also schon definiert und $\mathfrak{C} \models \exists y \beta_i[d]$.
   b) $d \in C - K$: Falls $f_i(d) \in \mathfrak{C}$, dann ist wie in a) $T_{\mathfrak{D}}[d, f_i(d)]$ bereits definiert. Andernfalls ist $T_{\mathfrak{A}}[f_i(d)] = \mathfrak{B} \in P - Q$. Sei dann $e = (\mathfrak{B}, i, 0) \in D$ der Zeuge für d. Wir setzen deshalb $T_{\mathfrak{D}}[d, e] \coloneqq T_{\mathfrak{A}}[d, f_i(d)]$ und dies garantiert, dass $\mathfrak{D} \models \exists y \beta_i[d]$.
   c) $d = (\mathfrak{B}, j, l) \in D - C$: Wir wählen ein $a \in A$ mit $T_{\mathfrak{A}}[a] = \mathfrak{B}$ und wählen $b = f_i(a)$. Es gibt nun zwei Möglichkeiten. Entweder $b \in K$, b ist also ein König. Dann setzen wir $T_{\mathfrak{D}}[d, b] \coloneqq T_{\mathfrak{A}}[a, b]$. Es handelt sich hierbei immer noch im eine konsistente Tablezuweisung, da d nicht der Zeuge für b sein kann, da Zeugen für b nur aus dem Hofstaat sein können.

      Andernfalls ist $b \notin K$, also ist der realisierte 1-Table von b, $T_{\mathfrak{A}}[b] = \mathfrak{B}' \in P - Q$. Wir wählen dann $e = (\mathfrak{B}', i, (l + 1) \bmod 3) \in D$ als Zeugen für d, indem wir $T_{\mathfrak{D}}[d, e] \coloneqq T_{\mathfrak{A}}[a, b]$ setzen. Dadurch, dass wir dieses Kreiszuweisungsschema für die Zeugen außerhalb des Hofstaates haben, nämlich $0 \to 1 \to 2 \to 0$, ist sichergestellt, dass es hier keine doppelten 2-Table Zuweisungen gibt.

4. Jedes Element $d \in D$ hat nun für jedes $1 \leq i \leq m$ einen Zeugen. Es gibt aber noch Paare $d, d' \in D$, für welche wir die 2-Table noch nicht festgelegt haben. Das ist in sofern noch relevant, da diese Paare alle die $\forall\forall$-Teilformel erfüllen müssen. Wir wählen dazu einfach zwei Elemente $a, a' \in A$, mit $T_{\mathfrak{A}}[a] = T_{\mathfrak{D}}[d], T_{\mathfrak{A}}[a'] = T_{\mathfrak{D}}[d']$ und setzen $T_{\mathfrak{D}}[d, d'] \coloneqq T_{\mathfrak{A}}[a, a']$.

5. Um $\mathfrak{D}$ vollständig zu definieren, müssen wir auch alle Relationen von 3 verschiedenen Elementen beschreiben. Wie oben beschrieben, haben diese aber keinen Einfluss darauf, ob $\mathfrak{D} \models \varphi$. Ist R eine mindestens 3-stellige Relation, dann setzen wir $(d_1, \ldots, d_k) \notin R^{\mathfrak{D}}$ für alle $\{d_1, \ldots, d_k\} \geq 3$.

Damit ist $\mathfrak{D}$ vollständig konstruiert und es gilt $\mathfrak{D} \models \varphi$ und $|D| < \infty$. □

Definition 3.14 (Formellänge)**.** *Eine σ-Formel φ kann als Zeichenkette über dem Zeichenvorrat $\{,\},(,\wedge,\forall,\neg,=\}$ zusätzlich zu dem Komma, den Symbolen in σ und den Variablen angesehen werden. Mit $n = |\varphi|$ bezeichnen wir die Länge dieser Zeichenkette.*

Korollar 3.15. *Ist $\varphi \in \Psi_{Scott}$ und $\mathfrak{A}$ eine σ-Struktur mit $\mathfrak{A} \models \varphi$ und $n = |\varphi|$, dann gibt es ein Modell $\mathfrak{D} \models \varphi$, dessen Universum D aus maximal $2^{O(n)}$ Elementen besteht.*

Beweis. Sei $\mathfrak{A}$ eine σ-Struktur mit $\mathfrak{A} \models \varphi$. Wir können φ auch als eine Formel über einer Signatur σ' auffassen, die nur die Relationssymbole aus φ enthält. Nach dem Koinzidenzlemma 2.3 existiert eine σ'-Struktur $\mathfrak{A}'$ mit $\mathfrak{A}' \models \varphi$. Nach vorherigem Satz können wir nun ein σ'-Modell $\mathfrak{D}'$ konstruieren. Das Universum D' ist hierbei

$$D' \coloneqq C \cup ((P - Q) \times \{1, \ldots, m\} \times \{0, 1, 2\}).$$

Die Elementanzahl in D' ist demnach endlich und durch folgenden Ausdruck begrenzt:

$$|D'| \leq |K| + m|K| + 3m(|P| - |K|) = (m+1)|K| + 3m(|P| - |K|).$$

Die Größen der Menge der Könige K und der Menge aller 1-Tables P sind beide durch $(2 * \ldots * 2 =)\ 2^n$ begrenzt, da es maximal n verschiedene Relationssymbole in φ geben kann. m ist durch die Länge von φ nach oben abzuschätzen, also $m \leq n = |\varphi|$. Insgesamt ergibt sich dann

$$\begin{aligned} |D'| &\leq (m+1)|K| + 3m(|P| - |K|) \\ &\leq (m+1)|K| + 3m|P| \\ &\leq (n+1)2^n + 3n2^n \\ &\leq 6n2^n \in 2^{O(n)}. \end{aligned}$$

Wir können $\mathfrak{D}'$ nun wieder auf ein Modell $\mathfrak{D}$ der Signatur σ erweitern, indem wir, wenn überhaupt existent, zusätzliche Relationen beliebig setzen. Nach dem Koinzidenzlemma 2.3 ist dann auch $\mathfrak{D} \models \varphi$ und $|D| \in 2^{\mathcal{O}(n)}$.

□

Korollar 3.16 (Satz von Mortimer)**.** *Sen(L_2) hat die endliche Modelleigenschaft und ist $\varphi \in$ Sen(L_2) und $\mathfrak{A}$ eine σ-Struktur mit $\mathfrak{A} \models \varphi$ und $n = |\varphi|$, dann gibt es ein Modell $\mathfrak{D} \models \varphi$, dessen Universum D aus maximal $2^{O(n)}$ Elementen besteht.*

Beweis. Ist $\varphi \in Sat(L_2)$ und $\Gamma(\varphi)$ die Scott-Reduktion 3.4 von φ, dann hat $\Gamma(\varphi)$ nach dem Korollar 3.15 ein Modell $\mathfrak{D}$ mit maximal $6m2^r$ Elementen, wobei r die Anzahl der verschiedenen Relationssymbole in $\Gamma(\varphi)$ ist und m die Anzahl der $\forall\exists$-Teilformeln in $\Gamma(\varphi)$. Nach dem Satz 3.4 gilt auch, dass $\mathfrak{D} \models \varphi$. Durch die Scott-Reduktion wird pro Quantor in φ eine Formel mit einem neuen Relationssymbol konjunktiert und es wird eine $\forall\exists$-Formel konjunktiert. Damit ist $m \leq q \leq n$ und $r \leq q + r' \leq n$, wobei q die Quantorenanzahl und r' die Anzahl der verschiedenen Relationssymbole in φ ist. Also hat φ ein Modell, nämlich $\mathfrak{D}$, mit $|D| \leq 6m2^r \leq 6n2^n \in 2^{O(n)}$ Elementen. □

3.4 Endliche Modellprüfung

Teil des Entscheidungsalgorithmus für L_2 wird es sein zu einem gegebenen σ-Satz φ und einer endlichen σ-Struktur $\mathfrak{A}$ zu entscheiden, ob $\mathfrak{A} \models \varphi$ gilt. Wir müssen daher zeigen, dass dies entscheidbar ist. Wir zeigen dazu, weil sich so der Entscheidungsalgorithmus leichter formulieren lässt, dass sogar das Problem, ob für eine endliche σ-Interpretation $\mathfrak{I}$ gilt, $\mathfrak{I} \models \varphi$, entscheidbar ist.

Damit wir zeigen können, dass das Problem entscheidbar ist, müssen wir zunächst Einschränkungen an Interpretationen festlegen, um sicherzustellen, dass die Menge von endlichen Interpretationen nicht überabzählbar wird. Die Entscheidbarkeit von überabzählbar großen Mengen ist nicht definiert, zumindest nicht mit dem Maschinenmodell einer Turingmaschine.

Sei für diesen Abschnitt σ daher eine Signatur mit endlich vielen Symbolen. Außerdem haben für diesen Abschnitt alle σ-Strukturen mit gleicher Universumsgröße n, alle dasselbe Universum A, nämlich $A = \{1, \ldots, n\}$. Das ist nicht problematisch, da alle anderen σ-Strukturen mit n Elementen isomorph zu einer σ-Struktur mit Universum $\{1, \ldots, n\}$ sind. Außerdem sollen Variablenbelegungen β von einer σ-Interpretation $\mathfrak{I} = (\mathfrak{A}, \beta)$ nur endlich viele Variablen x haben, sodass $\beta(x) \neq a$ für ein Platzhalterelement aus A. Durch diese Forderungen ist die Menge aller zweier Tupel von σ-Formeln und endlichen σ-Interpretationen abzählbar. Man beachte, dass es wichtig ist, dass σ nur endlich viele Symbole enthalten darf, da sonst die Mächtigkeit der Menge aller endlichen σ-Strukturen mindestens so groß ist, wie $\{0,1\}^{\mathbb{N}}$, aber $\{0,1\}^{\mathbb{N}}$ ist nach dem Lemma 2.2 überabzählbar. Wir können also die folgende Menge, die entschieden werden soll, kodieren. Sei nun

$$\mathcal{M} \coloneqq \{(\varphi, \mathfrak{I}) \colon \mathfrak{I} \models \varphi\}$$

die Menge aller Tupel von σ-Formeln φ und allen σ-Interpretationen, die ein Modell von φ sind.

Satz 3.17. *$\mathcal{M}$ ist entscheidbar.*

Beweis. Wir nutzen die rekursive Definition der Semantik der Prädikatenlogik aus und können so einen expliziten Entscheidungsalgorithmus angeben. Der folgende Algorithmus *istModell* 2 entscheidet nach Konstruktion die Menge $\mathcal{M}$.

Algorithmus 2: *istModell*

Eingabe : $\langle \varphi, \mathfrak{I} \rangle, (\varphi, \mathfrak{I}) \in \mathcal{M}$
Ausgabe: $\mathfrak{I} \models \varphi$

```
1  if φ = (s = t), s, t sind σ-Terme then
2      return s^I = t^I
3  if φ = R(t_1, ..., t_n) für ein n-stelliges Relationssymbol then
4      return R^I(t_1^I, ..., t_n^I)
5  if φ = ψ_1 ∧ ψ_2 then
6      return istModell(ψ_1, I) ∧ istModell(ψ_2, I)
7  if φ = ¬ψ then
8      return ¬istModell(ψ, I)
9  if φ = ∀xψ then
10     for a ∈ A do
11         if ¬istModell(ψ, I_x^a) then
12             return False
13     return True
```

□

3.5 Entscheidungsalgorithmus für L_2

Sei σ hier eine Signatur mit endlich vielen Symbolen.

Wir zeigen nun anhand der Resultate aus den vorherigen Kapiteln, dass L_2 entscheidbar ist. Wir geben dazu einen expliziten Entscheidungsalgorithmus an. Wir benutzen dazu den Algorithmus zur Modellprüfung *istModell* 2.

Satz 3.18. *$Sat(L_2)$ ist entscheidbar.*

Beweis. Wir gehen im Folgenden nur von Universen der Form $A = \{1, \ldots, n\}$ für Universen mit n Elemente aus, damit es nur endlich viele σ-Strukturen gibt, mit $\leq n$ Elementen im Universum. Folgender Algorithmus entscheidet $Sat(L_2)$:

Algorithmus 3: *isSat*

Eingabe : $\langle \varphi \rangle$, $\varphi \in L_2$
Ausgabe: $\varphi \in Sat(L_2)$

```
1  φ ← removeFreeVariables(φ)
2  n ← |φ|
3  for i = 1 ... 6n2^n do
4      for I ∈ {(A, id): |A| = i} do
5          if istModell(φ, I) then
6              return True
7  return False
```

Nach der Abschätzung im Satz von Mortimer 3.16, gibt es ein Modell für φ, $n = |\varphi|$ mit einem Universum mit weniger oder gleich $6n2^n$ Elementen, genau dann, wenn φ erfüllbar ist. Da der Algorithmus alle σ-Strukturen $\mathfrak{A}$ mit $|A| \leq 6n2^n$ auf $\mathfrak{A} \models \varphi$ überprüft, liefert *isSat* genau dann *True*, wenn φ erfüllbar ist.

□

4 Entscheidbare Klassen ohne Gleichheitsrelation

Im vorherigen Kapitel zur Entscheidbarkeit von L_2 hat sich ein Großteil der Beweisführung damit beschäftigt, mit der erlaubten Gleichheit in Formeln aus L_2 umzugehen. Die entscheidbaren Klassen, die ab hier nun behandelt werden, erlauben keine Gleichheit. Für dieses gesamte Kapitel gehen wir nur von Formeln ohne Gleichheitszeichen aus, ohne dies in den einzelnen Kapiteln nochmals explizit zu sagen.

4.1 Unifikation

Wir führen in diesem Abschnitt nun das Konzept der Unifikation ein. Informell gesprochen ist Unifikation, Terme durch Ersetzung von Variablen so anzupassen, dass diese gleich werden.

Unifikation wird in den folgenden Kapiteln benötigt, insbesondere ist es nötig zu wissen, dass gewisse Unifikationsprobleme entscheidbar sind. Im Kapitel zur Entscheidbarkeit der Herbrand-Formeln müssen die Terme von zwei Literalen unifiziert werden und im Kapitel zur Entscheidbarkeit der Maslov-Formeln muss bei der Robinson-Resolution ein allgemeinster Unifikator gefunden werden. Für dieses Kapitel wurde *TCDP* [2], ein Buchauszug [1] von Franz Baader und Wayne Snyder, ein Vorlesungsskript [9] von Sebastian Rudolph und ein Artikel [7] von Alberto Martelli und Ugo Montanari verwendet.

Sei ab hier σ eine fixe Signatur. Wir bezeichnen mit $Var = \{x_1, x_2, \ldots\}$ unsere abzählbare Menge an Variablen und mit T die Menge aller σ-Terme, die sich über diesen Variablen bilden lassen. Zunächst führen wir Substitutionen ein, um gleich auf Grundlage dieser Unifikatoren definieren zu können.

Definition 4.1 (Substitution)**.** *Eine Substitution $\pi\colon Var \mapsto T$ ist eine Abbildung von Variablen auf σ-Terme, wobei nur für endliche viele $x \in Var$ gilt, $\pi(x) \neq x$. Wir notieren π durch $\{x_{i_1} \mapsto \pi(x_{i_1}), \ldots, x_{i_n} \mapsto \pi(x_{i_n})\}$, wobei $x_{i_1}, \ldots, x_{i_n}$ die Variablen sind, für die gilt $\pi(x_{i_j}) \neq x_{i_j}$.*

Die endlich vielen Variablen x für die gilt $\pi(x) \neq x$, sind sozusagen die Variablen, die wir ersetzen. Die folgende Definition führt ein, auf was ein Term abgebildet wird, wenn eine Substitution auf diesen angewendet wird.

Definition 4.2. *Ist π eine Substitution und $t \in T$ ein Term, dann bezeichnet $\pi(t)$ den Term, der entsteht, wenn alle Variablen x in t gleichzeitig durch $\pi(x)$ ersetzt werden.*

> **Beispiel 4.3**
>
> Sei $\pi = \{x \mapsto f(x), y \mapsto z, z \mapsto c\}$, wobei c ein Konstantensymbol ist und f, g sind Funktionssymbole, dann ist

- $\pi(g(x,x)) = g(f(x), f(x))$
- $\pi(g(x,y)) = g(f(x), z)$
- $\pi(g(y,z)) = g(z, c)$

Die Folgende Definition führt ein, was es für eine Substitution heißt, ein Unifikator zu sein.

Definition 4.4 (Unifikator)**.** *Sind $t, s \in T$ und ist π eine Substitution, dann nennen wir π einen Unifikator von t und s, wenn $\pi(t) = \pi(s)$ gilt.*

Man bemerke, dass sich nicht jedes Paar von Termen t, s unifizieren lässt. Sind beispielsweise g, f zwei verschiedene einstellige Funktionssymbole, dann sind $t = g(x), s = f(y)$ nicht unifizierbar.

Es gibt verschiedenste Unifikationsprobleme. Eines davon ist das Problem zu einer endlichen Menge $M \subset T \times T$ von Paaren von Termen eine Substitution π zu finden, die ein Unifikator von allen Paaren $(s,t) \in M$ ist. Wir sagen dann π unifiziert M bzw. π ist ein Unifikator von M. Allgemein nennen wir Lösungen von Unifikationsproblemen, Unifikatoren. Häufig fordert man noch, dass ein Unifikator eine gewisse Eigenschaft besitzt, um mit diesem besser arbeiten zu können. Wir führen daher nun das Konzept von allgemeinsten Unifikatoren ein. Diese werden wir unter Anderem für die Robinson-Resolution im Kapitel zur Entscheidbarkeit der Maslov-Formeln benötigen.

Definition 4.5 (Allgemeinste Unifikatoren)**.** *Sind $\pi \neq \tau$ Substitutionen, dann ist π allgemeiner als τ, falls es eine Substitution θ gibt, sodass $\tau = \theta \circ \pi$. Löst π ein Unifikationsproblem, dann sagen wir π ist ein allgemeinster Unifikator, wenn für alle anderen Unifikatoren τ gilt, π allgemeiner ist als τ.*

Beispiel 4.6

Folgendes Beispiel demonstriert, dass ein allgemeinster Unifikator nicht unbedingt eindeutig ist . $\theta = \{x \mapsto y\}, \pi = \{y \mapsto x\}$ sind zwei allgemeinste verschiedene Unifikatoren, die $\{x, y\}$ unifizieren. Aber einerseits ist θ allgemeiner als π aber auch andersherum ist π allgemeiner als θ, denn $\theta = \{x \mapsto y\} \circ \pi$ und $\pi = \{y \mapsto x\} \circ \tau$.

Ohne Beweis ist aber zu bemerken, dass allgemeinste Unifikatoren jedoch bis auf Variablenumbenennungen eindeutig sind. Damit ist gemeint, dass man von jedem allgemeinsten Unifikator zu jedem Anderen, durch eine Komposition mit einer Substitution, welche nur Variablen umbenennt, gelangen kann.

Wir zeigen nun, dass die Unifikation von einer Menge von Paaren von Termen entscheidbar ist. Konkreter:

Satz 4.7. *Die Menge $\mathcal{U} = \{M \subset T \times T \colon |M| < \infty \wedge M \text{ ist unifizerbar }\}$ aller endlichen Mengen von Paaren von Termen, die unifizierbar sind, ist entscheidbar.*

Beweis. Wir geben einen Entscheidungsalgorithmus für $\mathcal{U}$ an und zeigen danach dessen Korrektheit. Dieser Algorithmus wird auch der Martelli-Montanari Algorithmus genannt [7]. Zur Sprechweise: Wir sagen eine Variable x kommt in einem Paar (u, v) vor, wenn einer der Terme u oder v, x enthält.

Algorithmus 4: *isUnifiable*

Eingabe : $\langle M \rangle$, M endlich und $M \subset T \times T$
Ausgabe: $M \in \mathcal{U}$

```
1  while ∃(s,t) ∈ M, das eine der folgenden If-Bedingungen erfüllt do
2      (s,t) ← beliebiges Paar aus M, das eine der folgenden If-Bedingungen
         erfüllt
3      //Regel (a)
4      if t ∈ Var ∧ s ∉ Var then
5          M ← (M − {(s,t)}) ∪ {(t,s)}
6          continue
7      //Regel (b)
8      if s = t then
9          M ← M − {(s,t)}
10         continue
11     //Regel (c1)
12     if (s,t) = (f(s_1,...,s_n), f(t_1,...,t_n)) then
13         M ← M − {(s,t)}
14         M ← M ∪ {(s_1,t_1),...,(s_n,t_n)}
15         continue
16     //Regel (c2)
17     if (s,t) = (f(s_1,...,s_n), g(t_1,...,t_m)), f ≠ g then
18         return False
19     //Regel (d)
20     if s ∈ Var ∧ t ≠ s∧ s kommt in einem anderen Paar in M vor then
21         if t enthält s then
22             return False
23         else
24             π ← {s ↦ t}
25             //M = {(s,t),(u_1,v_1),...,(u_n,v_n)}
26             M ← {(s,t),(π(u_1),π(v_1)),...,(π(u_n),π(v_n))}
27             continue
28 return True
```

Der Algorithmus wird im Beispiel 3, beispielhaft angewendet.
Um den Algorithmus nicht unnötig zu komplizieren nehmen wir hier der Einfachheit halber an, dass Konstantensymbole, 0-stellige Funktionssymbole sind. Ist c ein Konstantensymbol, dann würde Regel (c1) angewendet auf (c, c), dieses Paar einfach aus M entfernen und Regel (c2) angewendet auf $(c, f(s_1, \ldots, s_n))$ würde *False* liefern.

Um die Korrektheit von *isUnifiable* zu zeigen, müssen wir einerseits 1. zeigen, dass *isUnifiable* immer terminiert unabhängig davon in welcher Reihenfolge Paare gewählt werden. Andererseits müssen wir 2. zeigen, dass bei Eingabe eines M_{inp}, *isUnifiable* genau dann *True* liefert, wenn M_{inp} unifizierbar ist.

1. Idee: Wir weisen jedem Programmzustand ein Tripel von natürlichen Zahlen zu, definieren auf diesen Tripeln eine Ordnung und zeigen, dass durch jeden Schleifendurchlauf das Tripel des Programmzustands strikt kleiner wird. Die Ordnung wird so gewählt, dass es keine unendliche lange, absteigende Folge

dieser Tripel gibt. Sei dazu

$$F\colon\ \mathcal{U} \mapsto \mathbb{N}_0^3$$
$$M \mapsto (n_1, n_2, n_3),$$

n_1 = #Unaufgelöste Variablen
$= |\{x \in \mathit{Var}(M)\colon \neg\ (x \text{ kommt in exakt einem Paar } (u,v) \text{ vor und } x = u)\}|,$

n_2 =#Funktionsterme in M (Duplikate und Verschachtelte einschließlich),

$n_3 = |\{(t,t) : t \in T\}| + |\{(t,x) \in M\colon x \in \mathit{Var} \wedge t \notin \mathit{Var}\}|.$

Seien $n = (n_1, n_2, n_3), n' = (n_1', n_2', n_3') \in \mathbb{N}_0^3$. Wir definieren $n > n'$:

$$\begin{aligned} n > n' \iff & n_1 > n_1' \\ & \vee n_1 = n_1' \wedge n_2 > n_2' \\ & \vee n_1 = n_1' \wedge n_2 = n_2' \wedge n_3 > n_3'. \end{aligned}$$

Sei ab hier $M \subset T \times T$ ein Programmzustand und $n = (n_1, n_2, n_3) = F(M)$.

Durch Anwendung von den Regeln (a) oder (b) wird sicher n_3 um eins verringert und möglicherweise können n_1, n_2 verringert werden, aber sicher nicht erhöht werden.

Regel (c1) kann n_3 erhöhen, da hierdurch Paare der Form (t,t) oder (t,x) entstehen könnten. (c1) erhöht aber n_1 sicher nicht, jedoch wird n_2 sicher um 2 verringert, da zwei Funktionssymbole entfernt werden.

Regel (c2) führt zur Terminierung des Programms.

Regel (d) führt entweder dazu, dass das Programm direkt terminiert oder, dass n_2 zwar möglicherweise erhöht wird, da durch die Substitution beliebig viele Funktionssymbole in andere Paare "kopiert"werden, jedoch wird n_1 sicher dekrementiert. n_3 könnte erhöht aber auch verringert werden.

In jedem Schleifendurchlauf kann das Programm entweder sofort durch die Regeln (c2) oder (d) terminieren. Auch könnte das Programm sofort terminieren, wenn die Schleifenbedingung nicht mehr erfüllt ist. Tritt keiner dieser Fälle ein, dann gilt für das Folgezustandstripel n', dass $n > n'$. Da es keine unendlich lange, absteigende Folge von Programmzustandstripeln geben kann, muss das Programm also zwangsläufig terminieren.

2. Wir zeigen hierzu zunächst, dass die Menge der Unifikatoren der Eingabe nach jedem Schleifendurchlauf gleich bleibt. Für die Regeln (a),(b) und (c1) ist dies trivial. Sei M eine Termpaarmenge vor der Anwendung der Regel (d) und M' die Menge danach und (x,t) das ausgewählte Paar und π ein Unifikator (sofern existent) von M oder M'. Es gilt $(x,t) \in M \cap M'$. Regel (d) verändert nicht die Anzahl der Paare in M, sondern bildet jedes Paar (außer (x,t)) (u,v) durch die Substitution $\tau = \{x \mapsto t\}$ auf ein Paar (u',v') ab. Es gilt nun $\pi(x) = \pi(t) = \pi(\tau(x))$, da ja $\pi, (x,t)$ unifiziert. Daraus folgt dann, dass $\pi(u) = \pi(u')$ und $\pi(v) = \pi(v')$. Es gilt also für diese Paare $\pi(u) = \pi(v) \iff \pi(u') = \pi(v')$. Demnach unifiziert eine beliebige Substitution θ genau dann M, wenn θ M' unifiziert. Die Menge M_{end} im Schritt

bevor das Programm terminiert, hat demnach dieselben Unifikatoren wie die Eingabe M_{inp}.

Wenn der Algorithmus *True* zurück gibt, dann ist nach dem letzten Schleifendurchlauf

$$M_{end} = \{(x_1, t_1), \ldots, (x_n, t_n) | \forall i \colon x_i \in Var \wedge t_i \text{ enthält keine Variable aus } \{x_1, \ldots, x_n\}\}.$$

denn sonst hätte der Algorithmus nicht terminiert. M_{end} und damit M_{inp} ist trivialerweise unifizierbar durch $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$.

Der Algorithmus kann nur *False* durch Anwendung der Regel (c2) oder durch (d) liefern. Wurde Regel (c2) angewendet, dann gibt es ein Paar

$(f(s_1, \ldots, s_n), g(t_1, \ldots, t_m)), f \neq g$ in M_{end}. Kein Unifikator kann aus f, g machen. Wurde (d) angewendet und *False* zurückgeliefert, dann gibt es ein Paar $(x, t) \in M_{end}$ mit $x \in Var$ und $t \neq x$ ein Term, welcher die Variable x enthält. Angenommen es gäbe einen Unifikator π mit $\pi(x) = \pi(t)$. Dann wäre $\pi(x)$ ein Teilterm von $\pi(t)$. Da aber $t \neq x$ muss dann auch $\pi(x) \neq \pi(t)$ sein, was ein Widerspruch ist.

isUnifiable liefert also genau dann *True*, wenn die Eingabe $M_{inp} \in \mathcal{U}$ ist. Damit entscheidet *isUnifiable*, $\mathcal{U}$. Also ist $\mathcal{U}$ entscheidbar.

□

Beispiel 4.8

Wir zeigen den Algorithmus an zwei Beispielen. Die Menge im ersten Beispiel ist unifizierbar, die Menge im zweiten Beispiel jedoch nicht. Seien für die folgenden Beispiele x, y, z Variablen und c ein Konstantensymbol und f, g Funktionssymbole.

M	Paar	Regel	(n_1, n_2, n_3)	Ausgabe
$\{(x, y), (f(y, x), f(x, g(z)))\}$	$(f(x, y), f(x, g(z)))$	(c1)	(3,3,0)	-
$\{(x, y), (y, x), (x, g(z))\}$	(y, x)	(d)	(3,1,0)	-
$\{(y, x), (x, x), (x, g(z))\}$	(x, x)	(b)	(2,1,1)	-
$\{(y, x), (x, g(z))\}$	$(x, g(z))$	(d)	(2,1,0)	-
$\{(y, g(z)), (x, g(z))\}$	-	-	(1,2,0)	*True*

M ist also unifizierbar und ein Unifikator π lässt sich direkt aus M nach dem letzten Schleifendurchlauf ablesen: $\pi = \{y \mapsto g(z), x \mapsto g(z)\}$.

Das nächste Beispiel zeigt ein M, welches sich nicht unifizieren lässt, unabhängig von der Wahl der Paare.

M	Paar	Regel	(n_1, n_2, n_3)	Ausgabe
$\{(f(x, c), f(f(x, y), x))\}$	$f(f(x, y), x)$	(c1)	(2,4,0)	-
$\{(x, f(x, y)), (c, x)\}$	(c, x)	(a)	(2,2,1)	-
$\{(x, f(x, y)), (x, c)\}$	$(x, f(x, y))$	(d)	(2,2,0)	*False*

Wir gehen nochmal vom gleichen M aus, wählen nun aber in Schritt 3 das Tupel (x, c) aus.

M	Paar	Regel	(n_1, n_2, n_3)	Ausgabe
$\{(f(x,c), f(f(x,y),x))\}$	$f(f(x,y),x)$	(c1)	(2,4,0)	-
$\{(x, f(x,y)), (c,x)\}$	(c,x)	(a)	(2,2,1)	-
$\{(x, f(x,y)), (x,c)\}$	(x,c)	(d)	(2,2,0)	-
$\{(c, f(c,y)), (x,c)\}$	$(c, f(c,y))$	(c2)	(1,4,0)	*False*

Ein weiteres Unifikationsproblem ist es zu einer endlichen Menge $M \subset T, M = \{m_1, \ldots, m_n\}$ von Termen, eine Substitution π zu finden, sodass $\pi(m_1) = \ldots = \pi(m_n)$. Wir sagen π unifiziert M. Die Entscheidbarkeit dieses Unifikationsproblems lässt sich unmittelbar aus der Entscheidbarkeit des bereits behandelten Unifikationsproblems 4.7 folgern.

Korollar 4.9. *Die Menge $\mathcal{V} = \{M \subset T \colon M$ ist endlich und M ist unifizierbar$\}$ ist entscheidbar.*

Beweis. Ist $M = \{m_1, \ldots, m_n\} \subset T$ eine endliche Menge an Termen, dann berechne $F(M) \coloneqq \{(m_1, m_2), (m_2, m_3), \ldots, (m_n, m_1)\} \subset T \times T$. $F(M)$ hat die gleichen Unifikatoren wie M, denn für eine Substitution π gilt:

$$\pi(m_1) = \pi(m_2), \pi(m_2) = \pi(m_3), \ldots, \pi(m_n) = \pi(m_1) \iff \pi(m_1) = \ldots = \pi(m_n).$$

Um also zu überprüfen, ob M unifizierbar ist, bildet man $F(M)$ und überprüft, ob $F(M)$ unifizierbar ist, was nach Satz 4.7 entscheidbar ist.

□

Das nächste Korollar zeigt, dass sich der Unifikationsalgorithmus *isUnifiable* 4 leicht modifizieren lässt, um, falls existent, einen allgemeinsten Unifikator zu bestimmen.

Korollar 4.10. *Ist $M \subset T \times T$ endlich und unifizierbar und M_{end} die Instanz von M nach dem letzten Schleifendurchlauf von isUnifiable, dann hat M_{end} die Form*

$$M_{end} = \{(x_1, t_1), \ldots, (x_n, t_n) | \forall i \colon x_i \in Var \wedge t_i \text{ enthält keine Variable aus } \{x_1, \ldots, x_n\}\}.$$

und die Substitution $\pi = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ ist ein allgemeinster Unifikator von M.

Beweis. Hätte M_{end} nicht diese Form, dann hätte *isUnifiable* noch eine der Regeln (a)-(d) auf M_{end} anwenden können, M_{end} könnte dann also nicht die Instanz von M nach dem letzten Schleifendurchlauf sein.

π ist offensichtlich ein Unifikator von M_{end}. Nach dem Korrektheitsbeweis des Algorithmus *isUnifiable* 4, verändern die Schleifendurchläufe nicht die Menge der Unifikatoren von M. Damit ist π also auch ein Unifikator von M.

Ist $\tau \neq \pi$ ein weiterer Unifikator von M, dann ist τ auch ein Unifikator von M_{end} und es gilt sogar, dass $\tau = \tau \circ \pi$, denn

$$\tau(\pi(x_i)) = \tau(\pi(t_i)) = \tau(t_i) = \tau(x_i) \text{ für alle } i = 1 \ldots n.$$

damit ist π also ein allgemeinster Unifikator von M. □

Das Konzept der Unifikation lässt sich erweitern, indem man nicht mehr nur das Ziel hat, Terme zu unifizieren, sondern verschiedenste Klassen von Objekten. Gleich bleibt, dass eine Substitution bei dem jeweiligen Objekt Variablen durch Terme ersetzt. Ist beispielsweise, die Rede von einer Substitution π angewendet auf eine relationale Primformel $P(t_1, \ldots, t_n)$, dann ist $\pi(P(t_1, \ldots, t_n)) = P(\pi(t_1), \ldots, \pi(t_n))$. Alle Unifikationsprobleme von verschiedensten Objekten, die in dieser Arbeit auftauchen lassen sich auf triviale Weise auf die hier behandelten Unifikationsprobleme reduzieren, sodass wir darauf nicht weiter eingehen.

4.2 Skolemisierung

Die Skolemisierung ist ein Verfahren um eine beliebige Formeln auf die Skolem-Normalform umzuformen. Die Skolem-Normalform ist wie die Pränexnormalform, nur dass keine $\exists$-Quantoren im Quantorenpräfix erlaubt sind. Diese Umformung, bzw. Skolemisierung, erhält dabei die Erfüllbarkeitsäquivalenz. Die Konvertierung ist in sofern interessant, da diese Form leichter zu handhaben ist. Insbesondere werden wir uns im Beweis zur Entscheidbarkeit der Herbrand und Maslov-Formeln das Prinzip der Skolemisierung zu Nutze machen.

Dieser Abschnitt basiert auf Vorlesungen zur Skolemisierung von Evgeny Kruglov und Christoph Weidenbach [3] und von James Worrell [13].

Sei im Folgenden σ eine fixe Signatur mit abzählbar unendlich vielen Konstanten-/Funktionssymbolen.

Definition 4.11 (Skolem-Normalform)**.** *Eine σ-Formel φ in Pränexnormalform ist in der Skolem-Normalform, wenn der Quantorenpräfix von φ die Form $\forall x_1 \dots \forall x_k$ hat und die x_i paarweise verschieden sind.*

Beispiel 4.12

Seien R, H Relationssymbole und x, y, z Variablen.
$\varphi = \forall x \forall y R(x) \wedge \neg H(x, y)$ ist in der Skolem-Normalform,
$\psi = \exists x \forall y R(x) \wedge \neg H(x, y)$ hingegen nicht.

Der folgende Satz ist Kern, dieses Abschnitts. In diesem Satz wird gezeigt, dass sich jede Formel skolemisieren lässt. Den Algorithmus, der in diesem Beweis angegeben wird, werden wir später des Öfteren benutzen. Dieser Algorithmus wird beispielhaft im Beispiel 5 angewendet.

Satz 4.13 (Skolemisierbarkeit)**.** *Es gibt eine berechenbare Funktion*

$$\mathit{skolemize}\colon \mathit{Form}_\sigma \mapsto \mathit{Form}_\sigma,$$

sodass für alle φ gilt, dass φ genau dann erfüllbar ist, wenn skolemize(φ) erfüllbar ist und skolemize(φ) ist in Skolem-Normalform.

Beweis. Sei φ eine beliebige σ-Formel. Es ist aus dem Logik Skript [11] bekannt, dass sich φ auf eine erfüllbarkeitsäquivalente Formel in Pränexnormalform umformen lässt. Sei also

$$\varphi \coloneqq \mathit{Praenexnormalform}(\varphi) = Q_1 x_1 \dots Q_n x_n \psi.$$

Die nächsten Schritte werden alle die Pränexnormalform erhalten.

Wir können im nächsten Schritt sicherstellen, dass $x_1, \dots, x_n$ paarweise verschieden sind. Denn angenommen $x_i = x_j$ für $i > j$, dann kommt x_j nicht als freie Variable in der Formel vor, die Q_j quantisiert. Nach dem Lemma 2.4 kann $Q_j x_j$ durch $Q_j x'_j$ ersetzt werden, wobei x'_j eine neue Variable ist.

Im nächsten Schritt können wir nun φ so umformen, dass die führenden Quantoren $Q_1, \dots, Q_k, k \leq n$, $\forall$-Quantoren sind. Solange φ die Form $\exists x \psi$ hat, formen wir wie folgt um:

$$\varphi \coloneqq \psi[x/c],$$

wobei c ein neues Konstantensymbol ist. Dies nennen wir äußere Skolemisierung. Die so erhaltene Formel ist trivialerweise erfüllbarkeitsäquivalent.

Im letzten Schritt müssen wir noch die restlichen $\exists$-Quantoren aus φ entfernen. Informell gesprochen erreichen wir das, indem wir Variablen, die an einen $\exists$-Quantor gebunden sind, vor welchem nur $\forall$-Quantoren kommen, durch eine Auswahlfunktion f zu ersetzen, die als Argumente die gebundenen Variablen der $\forall$-Quantoren enthält. Dies nennen wir innere Skolemisierung. Etwas konkreter:

Enthält φ einen $\exists$-Quantor, dann hat φ folgende Form.

$$\varphi = \forall x_1 \ldots \forall x_k \exists y \psi, k \geq 1.$$

Es werden dann folgende Schritte auf φ ausgeführt, solange φ einen $\exists$-Quantor enthält.

1. Wähle k-stelliges Funktionssymbol f, das nicht in ψ vorkommt.
2. $\varphi \coloneqq \forall x_1 \ldots \forall x_k \psi[y/f(x_1, \ldots, x_k)]$.

φ ist nun in Skolem-Normalform, da keine Existenzquantoren mehr im Quantorenpräfix vorkommen. Wir müssen noch zeigen, dass auch die innere Skolemisierung die Erfüllbarkeitsäquivalenz gewährleistet.
Sei dazu $F = \forall x_1 \ldots \forall x_k \exists y \psi$ und $F' = \forall x_1 \ldots \forall x_k \psi[y/f(x_1, \ldots, x_k)]$. Dabei ist f ein n-stelliges Funktionssymbol, das nicht in F vorkommt. Wir zeigen, dass F erfüllbar ist gdw. F' erfüllbar ist.
" $\Longrightarrow$ ": Sei $\mathfrak{I}$ eine σ-Interpretation mit $\mathfrak{I} \models F$ und $a_1, \ldots, a_k \in A$ beliebig. Dann $\exists b \in A$: $\mathfrak{I}^{a_1,\ldots,a_k,b}_{x_1,\ldots,x_k,y} \models \psi$. Für eine beliebige Interpretation $\tilde{\mathfrak{I}}$, die gleich $\mathfrak{I}$ ist, abgesehen von $f^{\tilde{\mathfrak{I}}}$, gilt nach dem Koinzidenzlemma $\tilde{\mathfrak{I}}^{a_1,\ldots,a_k,b}_{x_1,\ldots,x_k,y} \models \psi$. Ist $f^{\tilde{\mathfrak{I}}}(a_1, \ldots, a_k) = b$, dann gilt durch das Übersetzungslemma 2.8 $\tilde{\mathfrak{I}}^{a_1,\ldots,a_k}_{x_1,\ldots,x_k} \models \psi[y/f(x_1, \ldots, x_k)]$. Da $a_1, \ldots, a_k$ beliebig waren, lässt sich also so ein $\mathfrak{I}'$ konstruieren mit $\mathfrak{I}' \models F'$, indem $\mathfrak{I}'$ gleich $\mathfrak{I}$ gesetzt wird, abgesehen von $f^{\mathfrak{I}}$. Für alle $a_1, \ldots, a_k \in A$ setzt man nun $f^{\mathfrak{I}'}(a_1, \ldots, a_k) = b$, für ein b mit $\mathfrak{I}^{a_1,\ldots,a_k,b}_{x_1,\ldots,x_k,y} \models \psi$.
" $\Longleftarrow$ ": Sei $\mathfrak{I}$ eine σ-Interpretation mit $\mathfrak{I} \models F'$. Dann gilt für alle $a_1, \ldots, a_k \in A$, dass $\mathfrak{I}^{a_1,\ldots,a_k}_{x_1,\ldots,x_k} \models \psi[y/f(x_1, \ldots, x_k)]$. Nach dem Übersetzungslemma 2.8 ist das äquivalent dazu, dass $\mathfrak{I}^{a_1,\ldots,a_k,b}_{x_1,\ldots,x_k,y} \models \psi$, $b = f^{\mathfrak{I}}(a_1, \ldots, a_k)$. Für alle $a_1, \ldots, a_k \in A$ gibt es also ein $b = f^{\mathfrak{I}}(a_1, \ldots, a_k) \in A$, mit $\mathfrak{I}^{a_1,\ldots,a_k,b}_{x_1,\ldots,x_k,y} \models \psi$. Daraus folgt, dass $\mathfrak{I} \models \forall x_1 \ldots \forall x_k \exists y \psi = F$.

Noch eine Anmerkung zu dem verwendeten Verfahren, die eine andere Blickweise auf das Verfahren gibt: Im Prinzip sind die innere und die äußere Skolemisierung äquivalent, sofern man Konstanten als 0-stellige Funktionen betrachtet.

Wir fassen nun das beschriebene Verfahren zur Umwandlung einer σ-Formel in eine Skolem-Normalform mittels Pseudocode zusammen. *skolemize* ist dann genau die Funktion, die durch den Algorithmus $skolemize_{Alg}$ berechnet wird.

Algorithmus 5: $skolemize_{Alg}$

Eingabe : $\langle\varphi\rangle$, $\varphi \in Form_\sigma$
Ausgabe: $\langle\varphi\rangle$ in Skolem-Normalform

```
1  φ ← Praenexnormalform(φ)
2  //φ = Q_1 x_1 ... Q_k ψ, ψ ist quantorenfrei
3  while ∃i > j : x_i = x_j do
4  |   x'_j ← neue Variable
5  |   Ersetze äußerstes Q_j x_j in φ durch Q_j x'_j
6  while φ = ∃xψ do
7  |   c ← neues Konstantensymbol
8  |   φ ← ψ[x/c]
9  while φ hat einen ∃-Quantor do
10 |   //φ = ∀x_1 ... ∀x_k ∃y ψ
11 |   f ← neues k-stelliges Funktionssymbol
12 |   φ ← ∀x_1 ... ∀x_k ψ[y/f(x_1, ..., x_n)]
13 return φ
```

Die von $skolemize_{Alg}$ berechnete Funktion *skolemize* ist damit eine berechenbare Funktion, die eine beliebige σ-Formel φ auf ihre Skolem-Normalform bringt. Da jeder Schritt von $skolemize_{Alg}$ Erfüllbarkeitsäquivalenz erhält, ist φ erfüllbarkeitsäquivalent zu $skolemize(\varphi)$. Somit erfüllt *skolemize* die geforderten Eigenschaften.

□

Beispiel 4.14

Wir wollen mittels des Verfahrens die σ-Formel

$$\varphi \coloneqq \exists y_1 \forall x_1 \exists y_2 \forall x_2 (R(y_1, x_1, y_2, x_2)),$$

skolemisieren. R ist dabei ein Relationssymbol. φ ist bereits in Pränexnormalform und alle Quantoren binden unterschiedliche Variablen. φ hat aber einen führenden $\exists$-Quantor. Wir führen daher eine Konstante c ein und ersetzen y_1 durch c. Wir erhalten dadurch

$$\varphi \coloneqq \forall x_1 \exists y_2 \forall x_2 (R(c, x_1, y_2, x_2)).$$

φ hat nun einen führenden $\forall$-Quantor, aber immer noch einen $\exists$-Quantor. Wir führen daher f als neues, einstelliges Funktionssymbol ein und ersetzen y_2 durch $f(x_1)$. Dadurch erhalten wir

$$\varphi \coloneqq \forall x_1 \forall x_2 (R(c, x_1, f(x_1), x_2)).$$

φ hat keinen $\exists$-Quantor mehr und damit ist das Verfahren beendet. φ ist jetzt in Skolem-Normalform.

Beispiel 4.15

Wir möchten anhand dieses Beispiels demonstrieren, dass *skolemize* zwar immer die Erfüllbarkeitsäquivalenz garantiert, aber im Allgemeinen nicht semantische Äquivalenz. In der Beweisführung haben wir unter Anderem gezeigt , dass für alle

φ und alle Interpretationen $\mathfrak{I}$ gilt, dass

$$\mathfrak{I} \models \mathit{skolemize}(\varphi) \implies \mathfrak{I} \models \varphi.$$

Die Rückrichtung gilt aber im Allgemeinen nicht. Sei dazu $\varphi = \exists x R(x)$, wobei R ein Relationssymbol ist. Dann ist $\mathit{skolemize}(\varphi) = R(c)$ für ein Konstantensymbol c. Es gilt z.B. für $\mathfrak{A}$ mit Universum $A = \{1, 2\}$ und $c^{\mathfrak{A}} = 1$ $(1), \notin R^{\mathfrak{A}}, (2) \in R^{\mathfrak{A}}$, zwar dass $\mathfrak{A} \models \varphi$ aber auch $\mathfrak{A} \not\models \mathit{skolemize}(\varphi)$.

4.3 Herbrand-Theorie

In diesem Kapitel werden Definitionen und Resultate aus der Herbrand-Theorie eingeführt und gezeigt. Diese Begriffe und Resultate werden für die Kapitel der Entscheidbarkeit der Herbrand-Formeln und der Maslov-Formeln benötigt.

Im Kapitel zur Entscheidbarkeit von Herbrand-Formeln werden wir, falls eine Formel erfüllbar ist, in Lemma 4.28 ein explizites Modell für diese konstruieren. Dieses Modell wird ein sogenanntes *Herbrand-Modell* sein. Im Kapitel zur Entscheidbarkeit der Maslov-Formeln ist zusätzlich der Begriff einer *Herbrand-Basis* wichtig. Das wohl wichtigste Resultat dieses Abschnitts ist das Herbrand-Theorem 4.20, was besagt, dass ein Satz in Skolem-Normalform genau dann erfüllbar ist, wenn dieser ein Herbrand-Modell besitzt.

Dieser Abschnitt basiert auf *TCDP* [2] und auf Vorlesungen zur Herbrand-Theorie von James Worrell [12] und von Javier Esparza und Uwe Schöning [10]. Außerdem stammt die Idee zur Definition des Herbrand-Universums, aus dem deutschen Wikipediaartikel zum Herbrand-Universum [6]. Diese Definition ist aber äquivalent zu den anderen gängigen Definitionen.

Sei für dieses Kapitel σ eine fixe Signatur mit mindestens einem Konstantensymbol.

Wir definieren nun zuerst den Begriff des Herbrand-Universums. Das Herbrand-Universum eines Satzes ist informell gesprochen die Menge aller Terme, die sich aus Zeichen (ohne Variablen) aus dem Satz bilden lassen.

Definition 4.16 (Herbrand-Universum)**.** *Ist φ ein σ-Satz in Skolem-Normalform, dann bezeichnen wir mit $\mathcal{H}_\varphi$ das Herbrand-Universum zu φ und es ist wie folgt induktiv definiert.*

1. *Kommt ein Konstantensymbol c in φ vor, dann ist $c \in \mathcal{H}_0$. Kommen keine Konstantensymbole in φ vor, dann ist $\mathcal{H}_0 \coloneqq \{a\}$, a ist Konstantensymbol aus σ.*
2. *$\mathcal{H}_{k+1} \coloneqq \mathcal{H}_k \cup G$. G ist die Menge aller Terme, die sich durch einmalige Anwendung der in φ vorkommenden Funktionssymbole, angewendet auf die Elemente in $\mathcal{H}_k$, bilden lassen.*
3. $\mathcal{H}_\varphi \coloneqq \bigcup_{k \geq 0} \mathcal{H}_k$.

Beispiel 4.17

Im Folgenden ein paar Beispiele für Herbrand Universen. c bezeichnet ein Kon-

stantensymbol, P ist ein Relationssymbol und f, g sind Funktionssymbole.

- Ist $\varphi = \forall x P(x)$, dann ist $\mathcal{H}_\varphi = \{a\}$
- Ist $\varphi = \forall x P(f(f(x))$, dann ist $\mathcal{H}_\varphi = \{a, f(a), f(f(a)), \ldots\}$
- Ist $\varphi = \forall x \forall y P(c, x, h(g(y)))$, dann ist $\mathcal{H}_\varphi = \{c, h(c), g(c), h(g(c)), h(h(c)), g(h(c)), g(g(c)), \ldots\}$

Mit Hilfe des Herbrand-Universums können wir nun Herbrand-Strukturen definieren. Die Idee hinter einer Herbrand-Struktur ist es, Symbole durch sich selbst zu interpretieren. Im auf diese Definition folgenden Lemma werden wir sehen, dass in einer Herbrand-Struktur jeder Term (also eine Verknüpfung von Symbolen) tatsächlich durch sich selbst interpretiert wird.

Definition 4.18 (Herbrand-Struktur)**.** *Ist φ ein σ-Satz und $\mathfrak{A}$ eine σ-Struktur, dann sagen wir, dass $\mathfrak{A}$ eine Herbrand-Struktur zu φ ist, wenn $\mathfrak{A}$ folgende Eigenschaften erfüllt.*

1. *Das Universum A ist das Herbrand-Universum zu φ, also $A = \mathcal{H}_\varphi$.*
2. *Für jedes Konstantensymbol c aus φ, gilt $c^{\mathfrak{A}} = c$. Enthält φ keine Konstantensymbole, dann ist $a^{\mathfrak{A}} = a$.*
3. *Für jedes n-stellige Funktionssymbol f aus φ gilt: $f^{\mathfrak{A}}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$ für alle $t_1, \ldots, t_n \in \mathcal{H}_\varphi$.*

Gilt zusätzlich $\mathfrak{A} \models \varphi$, dann sagen wir $\mathfrak{A}$ ist ein Herbrand-Modell von φ.

Folgendes Lemma zeigt die Stärke der Definition einer Herbrand-Struktur, indem tatsächlich jeder Term durch sich selbst interpretiert wird.

Lemma 4.19. *Ist φ ein σ-Satz und $\mathfrak{A}$ eine Herbrand-Struktur und $t \in \mathcal{H}_\varphi$, dann gilt $t^{\mathfrak{A}} = t$.*

Beweis. Wir zeigen die Behauptung durch Induktion über die Tiefe eines Terms. Sei dazu φ ein beliebiger σ-Satz, $\mathfrak{A}$ eine Herbrand-Struktur zu φ und $t \in \mathcal{H}_\varphi$.

1. Induktionsanfang $n = 0$: $t = c$ für ein Konstantensymbol c. Nach Definition von $\mathfrak{A}$ gilt $t^{\mathfrak{A}} = c^{\mathfrak{A}} = c = t$.
2. Induktionsschritt $n \rightsquigarrow n + 1$: $t = f(s_1, \ldots, s_n)$ für ein n-stelliges Funktionssymbol f. Es gilt

$$\begin{aligned} t^{\mathfrak{A}} &= f^{\mathfrak{A}}(s_1^{\mathfrak{A}}, \ldots, s_n^{\mathfrak{A}}) \\ &= f^{\mathfrak{A}}(s_1, \ldots, s_n) && \text{(Induktionsannahme)} \\ &= f(s_1, \ldots, s_n) && \text{(Nach Definition)} \\ &= t. \end{aligned}$$

□

Das folgende Theorem ist das zentrale Theorem in der Herbrand-Theorie. Dieses Theorem erlaubt es sich in den folgenden Kapiteln bei der Suche nach Modellen auf Herbrand-Strukturen zu beschränken.

Satz 4.20 (Theorem von Herbrand)**.** *Ist φ ein σ-Satz in Skolem-Normalform, dann ist φ genau dann erfüllbar, wenn φ ein Herbrand-Modell besitzt.*

Beweis. " $\Longrightarrow$ ": Sei $\mathfrak{B}$ ein beliebiges Modell von φ. Wir konstruieren nun ein Herbrand-Modell $\mathcal{H}$ von φ. Nach Definition muss das Universum von $\mathcal{H}$, das Herbrand-Universum $\mathcal{H}_\varphi$ sein und alle in φ vorkommenden Konstantensymbole oder Funktionssymbole (bzw. die funktionalen Terme) werden durch sich selbst interpretiert. Wir müssen also nur noch die Interpretation der Relationssymbole in φ festlegen. Alle nicht in φ vorkommenden Konstantensymbole, Relationssymbole oder Funktionssymbole können beliebig interpretiert werden. Ist R ein $n-$stelliges, in φ vorkommendes Relationssymbol, dann setzen wir für alle $t_1, \ldots, t_n \in \mathcal{H}_\varphi$

$$(t_1, \ldots, t_n) \in R^{\mathfrak{A}} \iff (t_1^{\mathfrak{B}}, \ldots, t_n^{\mathfrak{B}}) \in R^{\mathfrak{B}}.$$

Behauptung: $\mathcal{H} \models \varphi$. Wir zeigen dies über Induktion der in φ vorkommenden Quantoren:

1. Induktionsanfang $n = 0$: Da φ nach Voraussetzung ein Satz ist, kommen in φ nur Primformeln ohne freie Variablen und deren Konnektoren vor. Ist $R(t_1, \ldots, t_n)$ ein n-stelliges Relationssymbol in φ, dann gilt wegen unser Definition von $R^{\mathcal{H}}$, $\mathcal{H} \models R(t_1, \ldots, t_n) \iff \mathfrak{B} \models R(t_1, \ldots, t_n)$. Also muss dann auch $\mathcal{H} \models \varphi \iff \mathfrak{B} \models \varphi$ gelten.

2. Induktionsschritt $n \rightsquigarrow n + 1$: $\varphi = \forall x \psi$. Ist $t \in \mathcal{H}_\varphi$, dann gilt $\mathfrak{B} \models \psi[t^{\mathfrak{B}}]$. Daraus folgt durch das Übersetzungslemma 2.8 $\mathfrak{B} \models \psi[x/t]$. $\psi[x/t]$ enthält nun keine freien Variablen. Außerdem gilt, dass $\mathcal{H}_\varphi = \mathcal{H}_{\psi[x/t]}$, also ist, wie oben beschrieben, die aus $\psi[x/t]$ konstruierte Herbrand-Struktur gleich $\mathcal{H}$. Aus der Induktionsvoraussetzung folgt damit, dass $\mathcal{H} \models \psi[x/t]$. Daraus folgt dann wieder durch das Übersetzungslemma, nur in andere Richtung, dass $\mathcal{H} \models \psi[t^{\mathcal{H}}], t^{\mathcal{H}} = t$. Da $t \in \mathcal{H}_\varphi$ beliebig gewählt wurde, gilt dann also $\mathcal{H} \models \varphi$.

" $\Longleftarrow$ ": Trivial. □

Als nächstes führen wir den Begriff einer Herbrand-Basis ein. Das Konzept der Herbrand-Basis ist zentral im Kapitel zur Entscheidbarkeit von Maslov-Formeln.

Definition 4.21 (Herbrand-Basis)**.** *Ist φ ein σ-Satz, dann bezeichnet*

$$HB_\varphi := \{P(t_1, \ldots, t_n) \mid t_i \in \mathcal{H}_\varphi \text{ für } i=1 \ldots n, P \text{ ist } n\text{-stelliges Relationssymbol in } \varphi\}$$

die Herbrand-Basis von φ.

Möchte man zu einem Satz φ eine Herbrand-Struktur $\mathfrak{A}$ spezifizieren, dann muss man nur die nicht in φ vorkommenden, Konstanten- und Funktionssymbole interpretieren und alle Relationssymbole. Die Interpretation der in φ vorkommenden Relationssymbole lässt sich hierbei elegant durch eine Teilmenge $\mathcal{A} \subseteq HB_\varphi$ angeben. Ist nämlich P ein n-stelliges, in φ vorkommendes Relationssymbol, dann setzt man für $t_1, \ldots, t_n \in \mathcal{H}_\varphi$:

$$(t_1, \ldots, t_n) \in P^{\mathfrak{A}} \iff P(t_1, \ldots, t_n) \in \mathcal{A}.$$

Die nächste Definition ist die der Herbrand-Expansion. Die Folgenden Sätze basieren auf dieser Definition und liefern uns ein wichtiges Erfüllbarkeitskriterium für Sätze in Skolem-Normalform.

Definition 4.22 (Herbrand-Expansion)**.** *Ist φ ein σ-Satz in Skolem-Normalform und ist ψ der quantorenfreie Teil von φ mit den freien Variablen $x_1, \ldots, x_n$, dann bezeichnet*

$$E(\varphi) \coloneqq \{\psi[x_1/t_1] \ldots [x_n/t_n] \mid t_1, \ldots, t_n \in \mathcal{H}_\varphi\},$$

die Herbrand-Expansion von φ.

Dadurch, dass keine Variablen in den Formeln aus $E(\varphi)$ vorkommen und verschiedene Primformeln $P(t_1, \ldots, t_n), R(s_1, \ldots, s_l)$ unabhängig von einander entweder durch *wahr* oder $falsch$ bzw. $\mathfrak{A} \models P(t_1, \ldots, t_n)$ oder $\mathfrak{A} \not\models P(t_1, \ldots, t_n)$, interpretiert werden können, kann $E(\varphi)$ auch als eine Menge von aussagenlogischen Formeln angesehen werden, wobei die Primformeln die aussagenlogischen Variablen sind. Das nächste Lemma liefert uns ein Erfüllbarkeitskriterium, indem die Erfüllbarkeit eines Satzes in Skolem-Normalform auf die Erfüllbarkeit einer Menge an aussagenlogischen Formeln reduziert wird.

Lemma 4.23 (Gödel-Herbrand-Skolem Theorem)**.** *Ist φ ein σ-Satz in Skolem-Normalform und $E(\varphi)$ seine Herbrand-Expansion, dann ist φ genau dann erfüllbar, wenn $E(\varphi)$ erfüllbar ist.*

Beweis. Sei φ ein beliebiger Satz in Skolem-Normalform und $E(\varphi)$ seine Herbrand-Expansion. Mit ψ bezeichnen wir den quantorenfreien Teil von φ mit $x_1, \ldots, x_n$ die freien Variablen von ψ. Durch das Herbrand-Theorem 4.20 ist φ genau dann erfüllbar, wenn φ ein Herbrand-Modell besitzt. Sei deshalb $\mathfrak{A}$ eine beliebige Herbrand-Struktur.

$$\begin{aligned}
&\mathfrak{A} \models \varphi \\
\iff &\forall t_1, \ldots, t_n \in \mathcal{H}_\varphi \colon \mathfrak{A} \models \psi[t_1, \ldots, t_n] \\
\iff &\forall t_1, \ldots, t_n \in \mathcal{H}_\varphi \colon \mathfrak{A} \models \psi[x_1/t_1] \ldots [x_n/t_n] \qquad \text{(Übersetzungslemma 2.8)} \\
\iff &\forall G \in E(\varphi) \colon \mathfrak{A} \models G \\
\iff &\mathfrak{A} \models E(\varphi)
\end{aligned}$$

□

Das nächste Lemma ist auch als Endlichkeitssatz bekannt und besagt, dass Unerfüllbarkeit von Formelmengen nicht erst durch Unendlichkeit entstehen kann. Wir benötigten dieses Lemma als Hilfe, um eine noch stärkere Aussage als die im vorherigen Lemma 4.23 formulieren zu können.

Lemma 4.24 (Endlichkeitssatz)**.** *Ist Φ eine Formelmenge über einer Signatur σ, dann ist Φ genau dann unerfüllbar, wenn es eine endliche Teilmenge $\Phi_0 \subseteq \Phi$ gibt, die unerfüllbar ist.*

Hier ohne Beweis. Genaueres im Logik Skript [11].

Wir können dieses Lemma nun dazu nutzen, um eine noch stärkere Aussage zu zeigen, als die des Gödel-Herbrand-Skolem Theorem 4.23. Diese Aussage wird später für die Entscheidbarkeit von Maslov-Formeln von Relevanz sein.

Satz 4.25. *Ist φ ein σ-Satz in Skolem-Normalform und $E(\varphi)$ seine Herbrand-Expansion, dann ist φ genau dann unerfüllbar, wenn es eine endliche Teilmenge von $E(\varphi)$ gibt, die unerfüllbar ist.*

Beweis. Das folgt direkt aus dem Gödel-Herbrand-Skolem Theorem 4.23 und dem Endlichkeitssatz 4.24. φ ist genau dann unerfüllbar wenn $E(\varphi)$ unerfüllbar ist und $E(\varphi)$ ist genau dann unerfüllbar, wenn es eine endliche Teilmenge $E_0 \subseteq E(\varphi)$ gibt, die unerfüllbar ist. □

4.4 Entscheidbarkeit der Herbrand-Formeln

Dieses Kapitel hat das Ziel die Entscheidbarkeit der Herbrand-Formeln zu zeigen. Herbrand-Formeln sind σ-Formeln in Pränexnormalform, dessen quantorenfreier Teil eine Konjunktion aus Primformeln und negierten Primformeln ist. Dieses Kapitel basiert auf dem Kapitel zu Herbrand-Formeln aus *TCDP* [2].

Definition 4.26 (Herbrand-Formeln)**.** *Ist σ eine Signatur, dann bezeichnen wir mit $Form_{HB}(\sigma)$ die Menge aller σ-Formeln, mit Elementen φ der Form*

$$\varphi = Q_1 x_1 \dots Q_k x_k \bigwedge_{i=1}^{m} \alpha_i \wedge \bigwedge_{i=1}^{n} \neg\beta_i;\ \alpha_i, \beta_i \text{ sind Primformeln und } Q_i \in \{\exists, \forall\} \text{ für alle } i.$$

Ist σ aus dem Kontext ersichtlich, schreiben wir nur $Form_{HB}$ und mit $Sen(Form_{HB}) \subseteq Form_{HB}$ bezeichnen wir die Menge aller Herbrand-Sätze.

> **Beispiel 4.27**
>
> Im Folgenden ein paar Beispiele für Herbrand-Formeln und Formeln, die keine Herbrand-Formeln sind. c ist ein Konstantensymbol, R, H sind Relationssymbole und f ist ein Funktionssymbol.
>
> - $\forall x \exists y(R(x, c) \wedge \neg H(f(x, y))) \in Form_{HB}$
> - $\forall x(H(x) \wedge \exists y(R(x, y)) \notin Form_{HB}$
> - $\exists x(\neg(R(x) \wedge R(x))) \notin Form_{HB}$

Das Vorgehen im späteren Beweis wird es sein, Herbrand-Formeln zunächst zu skolemisieren und dann die skolemisierte Herbrand-Formel auf ein Erfüllbarkeitskriterium hin zu überprüfen. Dieses Kriterium wird in Lemma 4.28 formuliert. Es reduziert die Entscheidbarkeit auf ein Unifikationsproblem. Dieses Kapitel basiert daher auf den Kapiteln zur Unifikation und zur Herbrand-Theorie. Es ist insbesondere wichtig zu wissen, dass die Unifikation von Paaren von Termen ein entscheidbares Problem ist.

Für dieses Kapitel sei σ eine fixe Signatur mit abzählbar unendlich vielen Funktionssymbolen und Konstantensymbolen.

4.4.1 Entfernung der freien Variablen

Wir können in den folgenden Abschnitten wieder ohne Beschränkung der Allgemeinheit von σ-Sätzen ausgehen, da $Form_{HB}$ abgeschlossen ist unter Hinzufügung von äußeren $\exists$-Quantoren. Die Abbildung auf erfüllbarkeitsäquivalente Sätze *removeFreeVariables* 2.9 fügt nämlich nur äußere $\exists$-Quantoren hinzu.

4.4.2 Entscheidungsalgorithmus für Herbrand-Formeln

Ziel dieses Abschnittes ist nun mit Hilfe der Resultate aus den vorherigen Kapiteln zu zeigen, dass die Erfüllbarkeit von Herbrand-Formeln auf ein Unifikationsproblem reduzierbar ist.

Lemma 4.28. *Ein $\varphi \in Sen(Form_{HB})$ in Skolem-Normalform mit verschiedenen Variablennamen in verschiedenen Primformeln ist genau dann erfüllbar, wenn es in φ keine entgegengesetzten Literale $P(s_1, \ldots, s_k)$, $\neg P(t_1, \ldots, t,_k)$ gibt, für die die Menge $\{(s_1, t_1), \ldots, (s_k, t_k)\}$ unifizierbar ist.*

Beweis. " $\Longleftarrow$ " : Wir zeigen hierzu konstruktiv, dass φ ein Herbrand-Modell besitzt. Wir konstruieren nun dieses Herbrand-Modell $\mathfrak{A}$ mit Universum $\mathcal{H}_\varphi$. Sei R ein n-stelliges Relationssymbol, das l-mal in nicht negierter Form in φ vorkommt, dann bezeichne $R(s_{i_1}, \ldots, s_{i_n})$ das i-te, $i \leq l$, Vorkommen von R in φ. Wir interpretieren R dann wie folgt:

$$R^{\mathfrak{A}} \coloneqq \{(t_1, \ldots, t_n) \in \mathcal{H}_\varphi^n \mid \exists \pi, i \colon \pi \text{ unifiziert } \{(t_1, s_{i_1}), \ldots, (t_n, s_{i_n})\}\ \}.$$

Wir müssen nun zeigen, dass auch tatsächlich $\mathfrak{A} \models \varphi$ gilt. Ist R wieder ein n-stelliges Relationssymbol. Nach Definition von $R^{\mathfrak{A}}$ sind die Vorkommen von R in φ in nicht negierter Form immer erfüllt. Existiert $R(u_1, \ldots, u_n)$ aber auch in negierter Form in φ, dann muss für alle Terme $(t_1, \ldots, t_n) \in \mathcal{H}_\varphi^n$, für die $\{(t_1, u_1), \ldots, (t_n, u_n)\}$ unifizierbar ist, gelten, dass $(t_1, \ldots, t_n) \notin R^{\mathfrak{A}}$. Das ist aber gerade dadurch gewährleistet, dass sich nach Annahme keine entgegengesetzten Literale unifizieren lassen. Das lässt sich wie folgt begründen: Da Terme t aus $\mathcal{H}_\varphi$ keine Variablen enthalten, gilt für jede Substitution π, dass $\pi(t) = t$. Angenommen es gibt ein π_1, π_2 und ein i, sodass

$$t_1 = \pi_1(s_{i_1}), \ldots, t_n = \pi_1(s_{i_n}) \text{ und } t_1 = \pi_2(u_1), \ldots, t_n = \pi_2(u_n).$$

Dann folgt daraus sofort, dass

$$\pi_1(s_{i_1}) = \pi_2(u_1), \ldots, \pi_1(s_{i_n}) = \pi_2(u_n).$$

Da es nach Annahme keine Variable gibt, die in s_{i_j} und u_m vorkommt, für beliebige j, m, können die Substitutionen π_1, π_2 zu einer Substitution π erweitert werden, die ein Unifikator von $\{(s_{i_1}, u_1), \ldots, (s_{i_n}, u_n)\}$ ist. Das ist aber ein Widerspruch dazu ist, dass sich die Terme zweier entgegengesetzter Literale (hier $P(s_{i_1}, \ldots, s_{i_n})$, $\neg P(u_1, \ldots, u_n)$) nicht unifizieren lassen.

" $\Longrightarrow$ " : Nun die Rückrichtung. Es reicht nach dem Herbrand-Theorem 4.20 zu zeigen, dass φ kein Herbrand-Modell besitzt, wenn es zwei entgegengesetzte, unifizierbare Literale gibt. Gibt es zwei entgegengesetzte Literale $P(s_1, \ldots, s_n)$, $\neg P(t_1, \ldots, t_n)$, für die $\{(s_1, t_1), \ldots, (s_n, t_n)\}$ unifizierbar ist, dann gibt es einen Unifikator π, sodass $\pi(s_i), \pi(t_i) \in \mathcal{H}_\varphi$ ist, für alle i. Für jede Herbrand-Struktur $\mathfrak{A}$ gilt aber

$$(\pi(s_1)^{\mathfrak{A}}, \ldots, \pi(s_n)^{\mathfrak{A}}) = (\pi(s_1), \ldots, \pi(s_n)) = (\pi(t_1), \ldots, \pi(t_n)) = (\pi(t_1)^{\mathfrak{A}}, \ldots, \pi(t_n)^{\mathfrak{A}}).$$

und es kann aber nicht gleichzeitig gelten, dass einerseits $(\pi(s_1)^{\mathfrak{A}}, \ldots, \pi(s_n)^{\mathfrak{A}}) \in P^{\mathfrak{A}}$ und $(\pi(s_1)^{\mathfrak{A}}, \ldots, \pi(s_n)^{\mathfrak{A}}) \notin P^{\mathfrak{A}}$. Daher muss gelten $\mathfrak{A} \not\models \varphi$.

□

Satz 4.29. *Sat*($Form_{HB}$) *ist entscheidbar.*

Beweis. Wir beschreiben einen Entscheidungsalgorithmus und fassen diesen am Ende in Pseudocode nochmals zusammen: Sei $\varphi \in Form_{HB}$ beliebig. Wir können zunächst φ zu einem erfüllbarkeitsäquivalenten Satz in Skolem-Normalform umformen. Wir setzen dazu

$$\varphi := skolemize(removeFreeVariables(\varphi)).$$

φ hat nun die Form

$$\forall x_1 \dots \forall x_k \bigwedge_{i=1}^{m} \alpha_i \wedge \bigwedge_{i=1}^{n} \neg\beta_i;\ \alpha_i, \beta_i \text{ sind Primformeln für alle } i.$$

Wir können jetzt die Variablen in den Primformeln so umbenennen, dass keine zwei Primformeln, eine gleiche Variable enthalten. Sei im Folgenden $\forall\tilde{x}$ eine Abkürzung für $\forall x_1 \dots \forall x_n$, und $\tilde{x}$ eine Abkürzung für $x_1, \dots, x_n$ für ein n. Durch eine Umformung der Form

$$\forall\tilde{x}(\psi \wedge \psi') \mapsto \forall\tilde{x}\forall\tilde{y}(\psi \wedge \psi'[\tilde{x}/\tilde{y}]),$$

können wir sukzessiv die Variablen in einzelnen Primformeln so umbenennen, dass diese in keiner anderen Primformel in φ vorkommen. Dabei wäre ψ pro Schritt eine noch nicht umbenannte Primformel und ψ' der restliche quantorenfreie Teil von φ. Diese Umformung erhält die Skolem-Normalform und sogar semantische Äquivalenz, denn

$$\forall\tilde{x}(\psi \wedge \psi') \equiv \forall\tilde{x}\psi \wedge \forall\tilde{x}\psi' \equiv \forall\tilde{x}\psi \wedge \forall\tilde{y}\psi'[\tilde{x}/\tilde{y}] \equiv \forall\tilde{x}\forall\tilde{y}(\psi \wedge \psi'[\tilde{x}/\tilde{y}]).$$

Jetzt ist gewährleistet, dass zwei verschiedene Primformeln in φ keine gemeinsame Variable enthalten, sodass φ nun genau die Vorbedingungen erfüllt, die im Erfüllbarkeitskriterium für Herbrand-Sätze 4.28 gefordert sind. φ ist somit genau dann in *Sat*($Form_{HB}$), wenn es keine zwei entgegengesetzten Literale $P, \neg P$ in φ gibt, dessen Terme unifizierbar sind. Alle beschriebenen Schritte sind berechenbar, also ist *Sat*($Form_{HB}$) entscheidbar.

Das Verfahren fassen wir nun noch einmal mit Pseudocode zusammen.

Algorithmus 6: *isSat*

Eingabe : $\langle\varphi\rangle$, $\varphi \in Form_{HB}$
Ausgabe: $\varphi \in Sat(Form_{HB})$

```
1 φ ← skolemize(removeFreeVariables(φ))
2 Forme φ so um, dass verschiedene Primformeln keine gemeinsamen
    Variablen enthalten
3 for Primformelpaare P(s_1,...,s_n), ¬P(t_1,...,t_n) in φ do
4 |   if isUnifiable({(s_1,t_1),...,(s_n,t_n)}) then
5 |   |   return False
6 return True
```

□

4.5 Entscheidbarkeit der Maslov-Formeln

Ziel dieses Kapitels ist es die Entscheidbarkeit der sogenannten Maslov-Formeln zu zeigen. Wir werden dabei die aus der Aussagenlogik bekannte Resolution auf die Prädikatenlogik erweitern. Danach werden wir das Konzept von semantischen Bäumen einführen und wichtige Verknüpfungen zwischen der Resolution und den semantischen Bäumen herstellen. Insbesondere zeigen wir, wie sich die Vollständigkeit einer Resolutionsstrategie mit Hilfe von semantischen Bäumen zeigen lässt. Zum Ende führen wir eine spezielle Resolutionsstrategie ein, die M-Resolution, und zeigen, wieder durch semantische Bäume, dass die M-Resolution auch vollständig ist. Die Entscheidbarkeit der Maslov-Formeln folgt dann sofort aus der Vollständigkeit der M-Resolution. Dieses Kapitel basiert auf der Beweisführung aus *TCDP* [2]. Die Idee zur Notation der Robinson-Resolution stammt aus dem deutschen Wikipedia-Artikel zur Resolution [8].

Definition 4.30 (Krom-Formeln)**.** *Sei σ eine Signatur. KROM(σ) bezeichnet die Menge aller σ-Formeln φ in Pränexnormalform, deren quantorenfreier Teil ψ eine Konjunktion von veroderten Primformeln und negierten Primformeln ist, also*

$$\psi = \bigwedge_{i=1}^{m} (\alpha_i \vee \beta_i);\ \alpha_i, \beta_i \text{ sind Primformeln oder negierte Primformeln.}$$

Ist σ aus dem Kontext klar, dann schreiben wir nur KROM.

Definition 4.31 (Maslov-Formeln)**.** *Sei σ eine Signatur. $Form_M(\sigma)$ (oder auch Maslov-Formeln genannt) ist die Menge aller Formeln φ, die folgende Eigenschaften erfüllen.*

1. *$\varphi \in KROM$.*
2. *φ enthält keine Funktionssymbole.*
3. *Der Quantorenpräfix von φ hat die Form $\exists^*\forall^*\exists^*$.*

Ist σ aus dem Kontext klar, dann schreiben wir nur $Form_M$. Mit $Sen(Form_M) \subseteq Form_M$ bezeichnen wir die Menge aller Maslov-Sätze.

Beispiel 4.32

Im Folgenden ein paar Beispiele für Maslov-Formeln und Formeln, die keine Maslov-Formeln sind.

- $\forall x(R(x) \vee \neg L(x)) \in Form_M$
- $\exists x \exists y(H(f(x), y) \vee \neg H(x, y)) \notin Form_M$
- $\forall x \exists y \forall z(L(x, y, z) \vee R(x)) \notin Form_M$
- $\forall x R(x) \notin Form_M$

Sei ab hier σ eine fixe Signatur mit mindestens einem Konstantensymbol.

4.5.1 Robinson-Resolution

Aus der Veranstaltung *Logik und Formale System* [11] ist die Ableitungsregel Resolution bekannt, mit welcher die Erfüllbarkeit von Klauselmengen in der Aussagenlogik widerlegt werden kann. Zur Erinnerung: Sind C und D Klauseln und $p \in C, \neg p \in D$, dann lässt sich durch die Resolutionsregel die Klausel $R = (C - \{p\}) \cup (D - \{\neg p\})$ ableiten und wir notierten das wie folgt:

$$\frac{C \quad D}{(C - \{p\}) \cup (D - \{\neg p\})} \ .$$

Können wir durch wiederholtes Anwenden der Resolutionsregel die leere Klausel ableiten, dann folgt daraus, dass unsere Klauselmenge unerfüllbar ist. Tatsächlich wurde sogar gezeigt, dass das Resolutionskalkül vollständig ist, also eine endliche Klauselmenge Γ genau dann unerfüllbar ist, wenn sich aus Γ die leere Klausel ableiten lässt.

Das Prinzip dieser Resolution lässt sich auf prädikatenlogische Klauseln erweitern. Mit einer Klausel C bezeichnen wir eine endliche Menge an Literalen. Die Formel ψ zu der eine Klauselmenge Γ korrespondiert, ist die Konjunktion aller Klauseln in Γ (Jede Klausel wird als Disjunktion ihrer Literale geschrieben), umschlossen von $\forall$-Quantoren für jede Variable in $Var(\Gamma)$. Wir können daher über Γ wie über eine Formel sprechen. Wir sagen zum Beispiel Γ ist (un)erfüllbar, wenn ψ (un)erfüllbar ist.

Ist C eine Klausel und π eine Substitution, dann soll $\pi(C)$, die Klausel sein, die entsteht, wenn π auf alle Literale von C angewendet wird. Eine Substitution π angewendet auf eine Primformel $P(t_1, \ldots, t_n)$, ist $P(\pi(t_1), \ldots, \pi(t_n))$.

Definition 4.33 (Resolution)**.** *Sind C, D und E Klauseln, dann nennen wir E einen $\mathcal{R}$-Resolvent oder auch Robinson-Resolvent von C und D, wenn Folgendes gilt:*

1. *Es gibt Substitutionen π, τ, die nur Variablen auf Variablen abbilden (Umbenennen), sodass $\pi(C)$ und $\tau(D)$ keine gemeinsamen Variablen haben.*
2. *Es gibt $\emptyset \neq A \subseteq \pi(C), \emptyset \neq B \subseteq \tau(D)$, sodass $(\neg A \cup B)$ unifizierbar ist. $\neg A$ ist die Menge an Literalen, die entsteht, wenn die Literale in A negiert werden. Eine doppelt negierte Primformel, wird unnegiert.*

   Sei θ ein allgemeinster Unifikator von $\neg A \cup B$.
3. *$E = \theta((\pi(C) - A) \cup (\tau(D) - B))$.*

Wir notieren dies analog zur Resolution in der Aussagenlogik wie folgt

$$\frac{C \quad D}{E} \ \pi, \tau, \theta,$$

oder wenn $\pi, \tau = id$ mit

$$\frac{C \quad D}{E} \ \theta.$$

Wir sagen dann, dass $A \subseteq C$ und $B \subseteq D$ faktorisiert wurden.

Ist Γ eine Klauselmenge, dann bezeichnen wir mit $\Gamma \subseteq \mathcal{R}(\Gamma)$ die Menge aller $\mathcal{R}$-Resolventen, die sich aus Klauseln aus Γ bilden lassen, zusätzlich zu Γ.

Beispiel 4.34

Im Folgenden ein einfaches Beispiel für Robinson-Resolution. L, R, P sind Relationssymbole.

$$\frac{L(x), P(x,x), P(f(y),x) \quad \neg P(u,v), H(u)}{L(f(y)), H(f(y))}\ \theta,$$

$\theta = \{x \mapsto f(y), u \mapsto f(y), v \mapsto f(y)\}$.
Nun noch ein Beispiel, welches zeigt, weshalb wir die Variablenumbenennung fordern:

$$\frac{L(x), H(f(x)) \quad \neg H(x)}{L(x)}\ id, \{x \mapsto y\}, \{y \mapsto f(x)\}.$$

Ohne die vorherige Umbenennung, hätten $H(f(x))$ und $H(x)$ nicht unifiziert werden können.

Lemma 4.35 (Korrektheit Resolution)**.** *Sind C, D Klauseln und $E \in \mathcal{R}(\{C, D\})$ und ist $\mathfrak{A}$ eine σ-Struktur mit $\mathfrak{A} \models C, D$, dann folgt daraus, dass $\mathfrak{A} \models E$.*

Beweis. Seien π, τ, θ die Substitutionen aus der Definition der Resolution 4.33. Da die Variablen in den Klauseln $\forall$-quantifiziert sind, gilt $\mathfrak{A} \models \theta(\pi(C)), \theta(\tau(D))$. Seien $C' := \theta(\pi(C)), D' := \theta(\tau(D))$, $x_1, \ldots, x_r$ die Variablen in C, D und $y_1, \ldots, y_s$ die Variablen in C', D'. Mit C_i, D_i und C'_i, D'_i werden die i-ten Literale von C, D bzw. C', D' bezeichnet. Sei L das Literal, welches durch die Unifizierung mit θ faktorisiert wurde, also z.B. $L = C'_i$ für gewisse i und $\neg L = D'_j$ für gewisse j. Es gilt nun

$$\begin{aligned}
&\mathfrak{A} \models \forall x_1, \ldots, x_r (C_1 \vee \ldots \vee C_n) \wedge (D_1 \vee \ldots \vee D_m) \\
\Longrightarrow\ &\mathfrak{A} \models \forall y_1, \ldots, y_s (C'_1 \vee \ldots \vee C'_l \vee L \vee \ldots \vee L) \wedge \\
&\qquad (D'_1 \vee \ldots \vee D'_k \vee \neg L \vee \ldots \vee \neg L) \qquad (\textit{nach } \pi, \tau, \theta) \\
\Longrightarrow\ &\mathfrak{A} \models \forall y_1, \ldots, y_s (C'_1 \vee \ldots \vee C'_l \vee L) \wedge (D'_1 \vee \ldots \vee D'_k \vee \neg L) \\
\Longrightarrow\ &\mathfrak{A} \models \forall y_1, \ldots, y_s (\neg L \to C'_1 \vee \ldots \vee C'_l) \wedge (L \to D'_1 \vee \ldots \vee D'_k) \\
\Longrightarrow\ &\mathfrak{A} \models \{C'_1, \ldots, C'_l, D'_1, \ldots, D'_k\} = E.
\end{aligned}$$

□

Da gerade gezeigt wurde, dass die $\mathcal{R}$-Resolution korrekt ist, kann die Unerfüllbarkeit einer Klauselmenge wie in der Aussagenlogik gezeigt werden, indem man aus dieser die leere Klausel ableitet. Dieser etwas schwammige Begriff 'ableiten kann' präzisieren wir nun durch den Begriff der Resolutionsstrategie. Etwas 'ableiten zu können' bedeutet dann nichts anderes, als dass die leere Klausel durch eine endlich-fache Anwendung einer Resolutionsstrategie entsteht.

Definition 4.36 (Resolutionsstrategie)**.** *Eine Resolutionsstrategie $F \colon \mathcal{C} \mapsto \mathcal{C}$ ist eine Abbildung von einer Klasse von Klauselmengen auf sich selbst. Dabei ist die Abbildung von $\Gamma \in \mathcal{C}$ zu einer Klauselmenge $F(\Gamma) \supseteq \Gamma$ eine Menge von $\mathcal{R}$-Resolventen oder von Instanzen von $\mathcal{R}$-Resolventen von Klauselpaaren aus Γ. I ist eine Instanz eines $\mathcal{R}$-Resolventen E von Γ, wenn es eine Substitution π gibt, sodass $I = \pi(E)$.*

Beispiel 4.37

- Robinson-Resolution $\mathcal{R}(\Gamma)$ ist eine Resolutionsstrategie bezüglich der Klasse aller Klauseln
- Die M-Resolution $\mathcal{R}_M(\Gamma)$, die wir später einführen werden, ist eine Resolutionsstrategie bezüglich der Klasse aller M-Klauseln

Wir bemerken, dass nach Definition jede Resolutionsstrategie entweder $\mathcal{R}$-Resolventen oder Instanzen dieser liefert. Aus der Korrektheit von $\mathcal{R}$-Resolution folgt damit, dass wir mit beliebigen Resolutionsstrategien aus einer Klauselmenge Γ nicht die leere Klausel ableiten können, wenn Γ erfüllbar ist. Das Schwierige ist es also die Vollständigkeit von einer bestimmten Resolutionsstrategien zeigen, insbesondere für die uns am Ende wichtige M-Resolution.

4.5.2 Semantische Bäume

Wir führen nun das Konzept von semantischen Bäumen ein. Die einzelnen Definitionen und Erkenntnisse dieses Abschnitts verfolgen das Ziel, das Lemma 4.45 zeigen zu können. Dies ist das Werkzeug, welches wir benutzen werden, um am Ende die Vollständigkeit der M-Resolution zu zeigen.

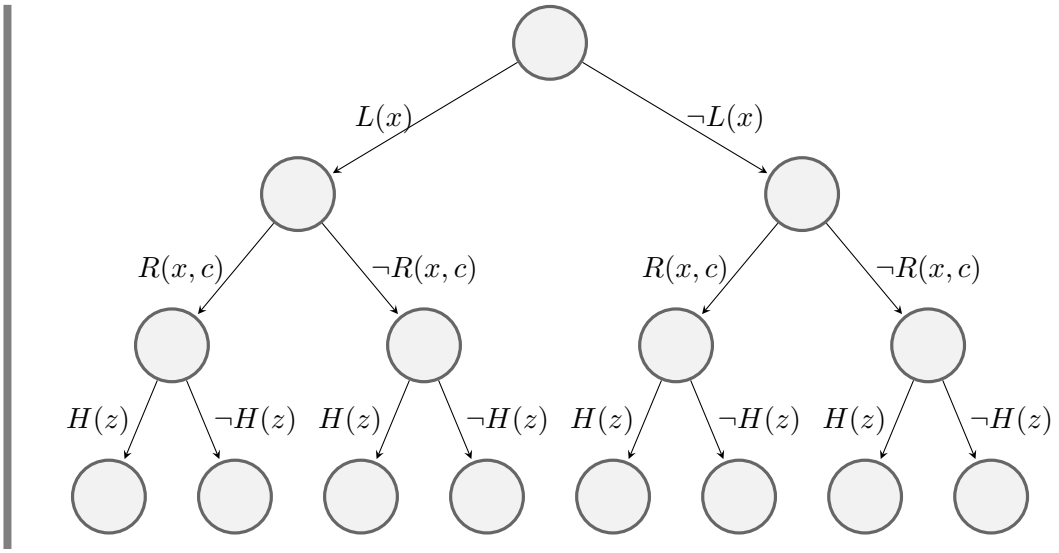
Definition 4.38 (Semantische Bäume)**.** *Sei K eine (möglicherweise abzählbar unendliche) Menge an Primformeln. Ein Binärbaum T, ist ein semantischer Baum, basierend auf K, wenn es eine fixe Nummerierung von den Elementen in K gibt, $\alpha_1, \alpha_2, \ldots \in K$, sodass Folgendes gilt:*

1. *Der Wurzelknoten hat zwei ausgehende Kanten, mit der Markierung α_1 und $\neg\alpha_1$.*
2. *Hat ein Knoten v eine eingehende Kante mit der Markierung α_i oder $\neg\alpha_i$ und ist $i < |K|$, dann hat v zwei ausgehende Kanten mit den Markierungen α_{i+1} und $\neg\alpha_{i+1}$. Ist $i = |K|$, dann hat v keine ausgehende Kanten.*

Wir bemerken, dass nach dieser Definition semantische Bäume vollständig sind, das heißt jeder Knoten hat entweder keine oder 2 ausgehende Kanten. Um das Konzept besser zu verstehen sollte man die Blickweise einnehmen, dass man Literale als aussagenlogische Variablen betrachtet und Pfade zu Blättern (möglicherweise unendliche lange Pfade, falls K unendlich groß ist) für vollständige Belegungen der Variablen stehen. Damit enthält ein semantischer Baum durch die Gesamtheit dieser Pfade, alle Variablenbelegungen.

Beispiel 4.39

Seien L, R, H Relationssymbole und c ein Konstantensymbol. Seien noch $K = \{L(f(x)), R(x, c), H(z)\}$ und $\alpha_1 = L(f(x)), \alpha_2 = R(x, c), \alpha_3 = H(z)$, dann ist folgender Binärbaum T ein semantischer Baum basierend auf K.



Definition 4.40. *Sei* Γ *eine Klauselmenge und* T *ein semantischer Baum, basierend auf einer Teilmenge der Herbrand-Basis* HB_Γ *von* Γ*. Wir führen nun folgende aufeinander aufbauende Begriffe ein.*

1. *Eine Klausel* $C \in \Gamma$ *scheitert an einem Knoten* v *in* T*, wenn es eine Substitution* π *der Variablen* $x_1, \ldots, x_r$ *in* C *durch Elemente des Herbrand-Universums* $\mathcal{H}_\Gamma$ *gibt, sodass alle Literale in* $\pi(C)$ *auf dem Weg von der Wurzel von* T *zu* v *in negierter Form als Kantenmarkierung auftreten.*

2. *Ein Knoten* v *von* T *ist ein Ausfallknoten bezüglich* Γ*, wenn es eine Klausel* $C \in \Gamma$ *gibt, die in* v *scheitert und keine Klausel an einem Vorgängerknoten von* v *scheitert.*

3. *Ein Knoten* v *von* T *dessen Nachfolger beide Ausfallknoten sind, heißt Entscheidungsknoten bezüglich* Γ*.*

4. T *ist abgeschlossen bezüglich* Γ*, wenn jeder Zweig von* T *einen Ausfallknoten besitzt.*

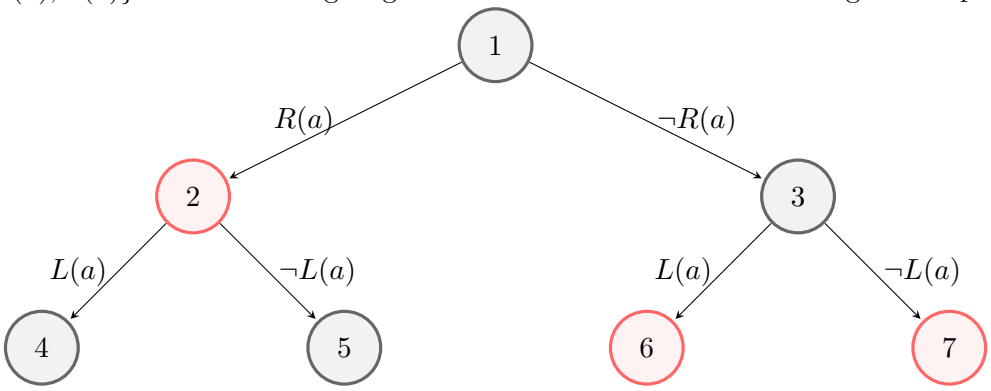
Nach dem Herbrand-Theorem 4.20 ist bekannt, dass ein Satz in Skolem-Normalform genau dann erfüllbar ist, wenn der Satz ein Herbrand-Modell besitzt. Die zu einer Klauselmenge Γ korrespondierende Formel ψ ist ein Satz in Skolem-Normalform, so dass Γ genau dann erfüllbar ist, wenn Γ ein Herbrand-Modell hat. Es reicht also aus uns auf Herbrand-Modelle zu beschränken bzw. auf semantische Bäume bezüglich einer Herbrand-Basis. Ein Herbrand-Modell $\mathfrak{A}$ ist hinsichtlich $\mathfrak{A} \models \Gamma$ oder $\mathfrak{A} \not\models \Gamma$ dadurch spezifiziert, wie die einzelnen Elemente der Herbrand-Basis HB_Γ interpretiert werden, also ob für ein $P \in HB_\Gamma$ gilt $\mathfrak{A} \models P$ oder $\mathfrak{A} \not\models P$. Jede Kante eines semantischen Baums T basierend auf einer Teilmenge von HB_Γ interpretiert ein Element der Herbrand-Basis durch seine Kantenmarkierung. Damit kodiert ein Pfad in T eine Menge von Interpretationen, die durch die Kantenmarkierungen entlang des Pfades festgelegt sind. Diese Menge wird kleiner je länger der Pfad ist, da dann mehr Elemente spezifiziert sind. Dass T abgeschlossen ist, kann so interpretiert werden, dass egal welchen Pfad man wählt, also egal welche Interpretationen man wählt, alle diese Interpretationen Γ bereits schon nicht erfüllen, oder der Pfad immer verlängert

werden kann, sodass alle Interpretationen, die dieser längere Pfad kodiert, allesamt Γ nicht erfüllen. Das bedeutet also, dass wenn es einen abgeschlossenen semantischen Baum zu Γ gibt, Γ unerfüllbar ist.

Beispiel 4.41

Wir veranschaulichen kurz die Begriffe anhand eines kleinen Beispiels. Seien dazu R, L Relationssymbole und a ein Konstantensymbol.

Sei $\Gamma = \{\{\neg R(x)\}, \{R(x), L(x)\}, \{\neg L(x)\}\}$, dann ist $\mathcal{H}_\Gamma = \{a\}$ und $HB_\Gamma = \{R(a), L(a)\}$. Die Abbildung zeigt einen semantischen Baum T bezüglich HB_Γ.



Knoten 2 ist ein Ausfallknoten, da $\{\neg R(x)\}$ dort scheitert. Knoten 6 ist ein Ausfallknoten, da dort $\{\neg L(x)\}$ scheitert. Auch Knoten 7 ist ein Ausfallknoten, da dort $\{R(x), L(x)\}$ scheitert. Damit ist Knoten 3 ein Entscheidungsknoten und T ist abgeschlossen unter Γ.

Wie bereits erwähnt ist es das Ziel ein Vollständigkeitskriterium für Resolutionsstrategien zu formulieren, damit wir dieses Kriterium nutzen können, um die Vollständigkeit von M-Resolution zu zeigen. Wir werden dabei wie folgt vorgehen. In der nächsten Definition führen wir die Vollständigkeit von Resolutionsstrategien bezüglich semantischer Bäume ein. Mit Lemma 4.45 werden wir dann zeigen, dass Vollständigkeit bezüglich semantischen Bäumen, Vollständigkeit impliziert und haben damit unser Kriterium formuliert.

Definition 4.42. *Wir sagen eine Resolutionsstrategie $F\colon \mathcal{C} \mapsto \mathcal{C}$ ist vollständig im Bezug auf semantische Bäume, wenn es für jede unerfüllbare Klauselmenge $\Gamma \in \mathcal{C}$ einen semantischen Baum T basierend auf einer endlichen Teilmenge von HB_Γ gibt, sodass*

1. *T ist abgeschlossen unter Γ.*
2. *Für alle Klauselpaare $C, D \in \Gamma$, die beide an unterschiedlichen Kinderknoten eines Entscheidungsknotens v scheitern, gibt es ein $E \in F(\Gamma)$, sodass E an v scheitert.*

Dass die 1. Bedingung aus obiger Definition zumindest immer erfüllt werden kann, zeigt das nächste Lemma. Diese Bedingung ist insbesondere für den Beweis von dem späteren Lemma 4.45 von essenzieller Bedeutung, da die Beweisidee dort nur bei endlichen Bäumen funktioniert.

Lemma 4.43. *Ist Γ eine unerfüllbare Klauselmenge, dann gibt es eine endliche Teilmenge $K \subseteq HB_\Gamma$, sodass jeder semantische Baum, der auf K basiert, abgeschlossen ist bezüglich Γ.*

Beweis. Ist Γ unerfüllbar, dann gibt es nach Lemma 4.25 eine endliche Teilmenge E_0 von der Herbrand-Expansion $E(\Gamma)$, die unerfüllbar ist. Wir setzen nun

$$K \coloneqq \bigcup_{G \in E_0} \{R(t_1, \ldots, t_n) \in HB_\Gamma \mid R(t_1, \ldots, t_n) \text{ kommt in } G \text{ vor}\}.$$

Wir zeigen, dass jeder semantische Baum, basierend auf K abgeschlossen ist bezüglich Γ. Sei dazu T ein semantischer Baum basierend auf K. Da K endlich ist, ist auch T endlich. Angenommen T wäre nicht abgeschlossen. Dann gibt es einen Pfad mit Literalen $L_1, \ldots, L_n$, sodass keine Klausel aus Γ an einem Knoten auf diesem Pfad scheitert. Seien $R_1, \ldots, R_n$ die entsprechenden Primformeln zu den Literalen. Wählen wir nun eine Herbrand-Struktur $\mathfrak{A}$ von E_0, die die Primformeln wie folgt interpretiert:

$$\mathfrak{A} \models R_i \iff R_i = L_i \text{ (also } \mathfrak{A} \not\models R_i \iff L_i = \neg R_i).$$

Dann muss $\mathfrak{A} \models E_0$ gelten, denn jede Klausel in E_0 wird von $\mathfrak{A}$ erfüllt, da es sonst eine Klausel aus Γ geben würden, die auf dem Pfad $L_1, \ldots, L_n$ scheitert. Dass $\mathfrak{A} \models E_0$ gilt ist aber ein Widerspruch dazu, dass E_0 unerfüllbar ist. Also darf es keinen Pfad in T geben, auf dem keine Klausel aus Γ scheitert. Folglich ist T bezüglich Γ abgeschlossen.

□

Wir können nun durch vorheriges Lemma einen weiteren wichtigen Satz zeigen, welcher ebenfalls für das Vollständigkeitskriterium in Lemma 4.45 benötigt wird.

Satz 4.44. *$\mathcal{R}$-Resolution ist vollständig im Bezug auf semantische Bäume.*

Beweis. Sei Γ eine unerfüllbare Klauselmenge. Nach Lemma 4.43 gibt einen semantischen Baum T basierend auf einer endlichen Teilmenge von HB_Γ, der abgeschlossen ist bezüglich Γ. Sei nun v ein Entscheidungsknoten und v_1, v_2 seine Kinder und C, D zwei Klauseln, die an v_1 bzw. v_2 scheitern. Sei $L_\mathcal{H}$ das Literal der Kante $v \to v_1$ und $\neg L_\mathcal{H}$ das der Kante $v \to v_2$. O.B.d.A. können wir davon ausgehen, dass C, D bereits keine gemeinsamen Variablen enthalten, da wir die Variablen bei der $\mathcal{R}$-Resolution umbenennen dürfen bzw. müssen. Da v_1, v_2 Ausfallknoten sind und C, D deshalb an keinem Vorgängerknoten scheitern, muss C ein Literal L_C enthalten, sodass es eine Substitution π_1 gibt, sodass alle Literale in $\pi_1(C)$ negiert auf dem Weg zu v_1 vorkommen und $\pi_1(L_C) = \neg L_\mathcal{H}$. D muss ein Literal L_D enthalten, sodass es eine Substitution π_2 gibt sodass alle Literale in $\pi_2(D)$ auf dem Weg zu v_2 negiert vorkommen und $\pi_2(L_D) = \neg(\neg L_\mathcal{H})$. Sei P das Relationssymbol zu $L_D, L_C, L_\mathcal{H}$. Damit ist $\{\neg L_C, L_D\}$ unifizierbar und wir können nun mindestens eine Klausel $E' \in \mathcal{R}(\{C, D\})$ resolvieren, sodass Literale mit dem Relationssymbol P faktorisiert wurden. E' wird weiterhin spätestens in v_1 und v_2 scheitern, da bei der $\mathcal{R}$-Resolution ein allgemeinster Unifikator gewählt wurde. Es kann aber nun immer noch sein, dass E' erst in v_1, v_2 scheitert, aber noch nicht bereits in v. Das liegt daran, dass es noch Literale mit Relationssymbol P geben könnte. Wir wählen daher $E \in \mathcal{R}(\{C, D\})$ so, dass einerseits L_C, L_D faktorisiert werden und E in v_1, v_2 scheitert aber andererseits auch so, dass E eine minimale Anzahl an Literalen enthält. Nun wird E bereits sicher in v scheitern, da wenn E erst in v_1, v_2 scheitern würde, dann gäbe es wie oben beschrieben wieder L_C, L_D, sodass $\{\neg L_C, L_D\}$ unifizierbar ist. Diese hätte man aber auch direkt mit der Resolution zu E entfernen können und damit wäre E nicht minimal. □

Wir können nun durch folgendes Lemma endlich den Bogen zurück zur Vollständigkeit von Resolutionsstrategien schlagen. Dieses Lemma ist ein wichtiges Werkzeug um die Vollständigkeit von Resolutionsstrategien zu zeigen. Wir werden dieses Lemma auch am Ende benutzen, um die Vollständigkeit der M-Resolution zu zeigen.

Lemma 4.45. *Sei $F\colon \mathcal{C} \mapsto \mathcal{C}$ eine Resolutionsstrategie, welche vollständig im Bezug auf semantische Bäume ist. Dann kann aus jeder unerfüllbaren Klauselmenge $\Gamma \in \mathcal{C}$ die leere Klausel abgeleitet werden.*

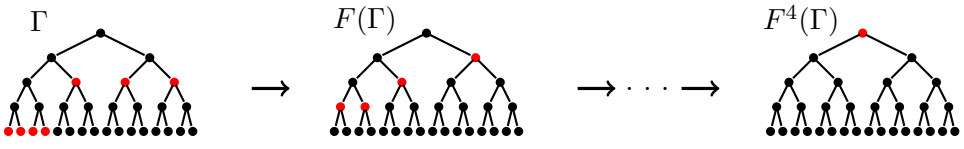
Beweis. Sei $F\colon \mathcal{C} \mapsto \mathcal{C}$ eine vollständige Resolutionsstrategie bezüglich semantischer Bäume und $\Gamma \in \mathcal{C}$ eine unerfüllbare Klauselmenge. Sei T dann ein semantischer Baum wie aus der Definition zu vollständigen Resolutionsstrategien 4.42. Da T endlich ist, gibt es auch sicher nur endlich viele Ausfallknoten bezüglich Γ in T.

Wir verwenden daher vollständige Induktion über die Anzahl der Ausfallknoten.

1. Induktionsanfang $n = 0, n = 1$: T kann nicht 0 Ausfallknoten bezüglich Γ haben, da Γ sonst erfüllbar wäre. Hat T nur einen Ausfallknoten bezüglich Γ, dann muss dieser Ausfallknoten die Wurzel von T sein, weil T sonst mindestens zwei Ausfallknoten hätte, nämlich einen im linken Zweig der Wurzel und einen im rechten Zweig, da T abgeschlossen ist bezüglich Γ. Da die Wurzel ein Ausfallknoten ist, muss Γ bereits die leere Klausel enthalten und wir sind fertig.

2. Induktionsschritt $n \rightsquigarrow n + 1$: Gibt es mindestens zwei Ausfallknoten in T bezüglich Γ, dann muss es mindestens einen Entscheidungsknoten in T bezüglich Γ geben. Wäre dies nämlich nicht der Fall dann kann für jeden Knoten v ein Nachfolger v' gewählt werden, sodass v' kein Ausfallknoten ist. Somit gäbe es einen Pfad bis zu einem Blatt ohne Ausfallknoten, was aber ein Widerspruch dazu ist, dass T bezüglich Γ nach Annahme abgeschlossen ist. Seien nun C, D Klauseln, die an den Kindern eines Entscheidungsknotens v bezüglich Γ scheitern. Dann muss es nach Definition einer vollständigen Resolutionsstrategie eine Klausel $E \in F(\Gamma)$ geben, sodass E an v scheitert. Bezüglich $\Gamma \cup \{E\}$ sind damit die Kinder von v keine Entscheidungsknoten mehr und entweder ist v selbst nun ein Entscheidungsknoten oder ein Vorgänger von v ist ein Entscheidungsknoten. Das heißt die Anzahl der Ausfallknoten in T bezüglich $\Gamma \cup \{E\}$ ist um mindestens eins kleiner als bezüglich Γ. Nun ist $\Gamma \subseteq \Gamma \cup \{E\} \subseteq F(\Gamma)$, das heißt durch die Resolutionsstrategie werden keine Klauseln entfernt, also ist T auch abgeschlossen bezüglich $F(\Gamma)$. Und da das Hinzufügen von Klauseln, die Anzahl der Ausfallknoten nicht erhöhen kann (Ausfallknoten bleiben Ausfallknoten oder ein Vorgänger wird stattdessen ein Ausfallknoten), muss T bezüglich $F(\Gamma)$ strikt weniger Ausfallknoten haben, als bezüglich Γ. Nach Induktionsannahme muss nun die leere Klausel aus $F(\Gamma)$ ableitbar sein, also auch aus Γ.

Folgende Abbildung veranschaulicht die Intuition dahinter und zeigt, wie die Ausfallknoten (in rot) durch Anwendung einer vollständigen Resolutionsstrategie in Richtung Wurzel wandern.



□

Man kann sich die Anwendung einer vollständigen Resolutionsstrategie so vorstellen, dass es bei einer unerfüllbaren Klauselmenge Γ einen semantischen Baum basierend auf einer endlichen Teilmenge der Herbrand-Basis von Γ gibt, der abgeschlossen ist. Wir müssen diesen konkreten Baum nicht kennen, aber wir wissen nun, dass jeder Schritt der Resolutionsstrategie die Ausfallknoten in diesem Baum "nach oben schiebt", sodass wir irgendwann bei der Wurzel ankommen werden, bzw. die leere Klausel ableiten werden.

4.5.3 M-Resolution

Wir bereits erwähnt lassen sich skolemisierte Maslov-Formeln als Klauselmengen auffassen. Wir möchten in den nächsten Definitionen die Klauseln von skolemisierten Maslov-Formeln etwas genauer charakterisieren, da diese gewisse, für uns später nützliche Eigenschaften besitzen. Wir werden diesen Klauseltyp, M-Klauseln bzw. Maslov-Klauseln nennen.

Um eine Maslov-Klausel präzise definieren zu können, brauchen wir zunächst ein paar Hilfsdefinitionen, die wir nun einführen werden.

Definition 4.46 (Kongruenz)**.** *Wir nennen zwei Terme $f(s_1, \ldots, s_n), g(t_1, \ldots, t_n)$ kongruent zueinander, wenn es eine Permutation $\pi \in \mathcal{S}_n$ gibt, sodass $(s_1, \ldots, s_n) = (t_{\pi(1)}, \ldots, t_{\pi(n)})$.*

Definition 4.47. *Die Tiefe eines Literals L ist die maximale Tiefe aller Terme in L und die Tiefe einer Klausel, ist die maximale Tiefe aller seiner Literale.*

Definition 4.48 (Uniformität)**.** *Ist C eine Klausel und $L(s_1, \ldots, s_n)$ ein Literal von C, dann sind $s_1, \ldots, s_n$ die Argumente von L. Wir sagen C ist uniform, wenn entweder keine Funktionssymbole in C vorkommen oder es ein funktionales Argument $t = f(u_1, \ldots, u_k)$ eines Literals in C gibt, sodass alle anderen Argumente von Literalen entweder kongruent zu t sind oder Konstantensymbole sind oder ein Argument von t sind. Ein Literal L ist uniform, wenn $\{L\}$ uniform ist.*

Beispiel 4.49

Im Folgenden wird die Uniformität von Klauseln an Beispielen gezeigt. Seien dazu R, L, H Relationssymbole und f, g Funktionssymbole und c ein Konstantensymbol.

- $C = \{\neg R(x), L(c, y)\}$ ist uniform, da keine Funktionssymbole vorkommen
- $C = \{R(f(g(x), y, z)), L(c, h(z, y, g(x))), H(g(x))\}$ ist auch uniform. Wählen wir $t = f(g(x), y, z)$, dann ist c ein Konstantensymbol, $h(z, y, g(x))$ ist kongruent zu t und $g(x)$ in $H(g(x))$ ist ein Argument von t

Wir können dank dieser Hilfsdefinitionen nun die Maslov-Klausel definieren.

Definition 4.50 (M-Klauseln)**.** *Mit M (oder auch Maslov-Klauseln) bezeichnen wir die Menge von Klauseln, deren Elemente C folgende Eigenschaften erfüllen.*

1. *C hat ≤ 2 Literale.*
2. *C hat eine Tiefe ≤ 1.*
3. *C ist uniform.*

Wir gehen im finalen Beweis zur Entscheidbarkeit der Maslov-Formeln 4.5.5 genauer darauf ein, aber M-Klauseln sind für uns in sofern relevant, dass alle Klauseln eines skolemisierten Maslov-Satzes M-Klauseln sind. Wir definieren nun die M-Resolutionsstrategie, die genau auf den M-Klauseln operiert. Wir werden später zeigen, dass die M-Resolution vollständig ist und wir mit Hilfe dieser Maslov-Formeln entscheiden können.

Definition 4.51 (M-Resolution)**.** *Mit $\mathcal{R}_M \colon \mathcal{P}(M) \mapsto \mathcal{P}(M)$ bezeichnen wir die M-Resolutionsstrategie. Ist $\Gamma \subseteq M$, dann gilt für jedes $G \in \mathcal{R}_M(\Gamma)$, dass $G = \pi(E)$, wobei $E \in \mathcal{R}(\Gamma)$ und π ist eine Substitution.*

Es stellt sich die Frage weshalb wir eine neue Resolutionsstrategie nur für die M-Klauseln einführen, wenn die $\mathcal{R}$-Resolution doch bereits vollständig und korrekt ist. Das hat den Grund, dass die Menge aller durch $\mathcal{R}$-Resolution ableitbaren Klauseln einer initialen M-Klauselmenge nicht unbedingt endlich sein muss. Es ist also nicht garantiert, dass man nur endlich viele Ableitungsschritte benötigt, um alle ableitbaren Klauseln, abzuleiten. Ein Beispiel dafür ist folgende Ableitung, die wiederholt angewendet ein beliebig tief verschachteltes Literal resolvieren kann:

$$\frac{R(f(x)), H(x) \quad \neg H(f(y)), H(y)}{R(f(f(y))), H(y)} \{x \mapsto f(y)\}.$$

Wir bemerken aber, dass die Menge M über einer Signatur mit endlich vielen Symbolen, bis auf die Umbenennung von Variablen, endlich ist. Die Strategie $\mathcal{R}_M$ muss daher nur endlich oft auf eine beliebige Klauselmenge $\Gamma \subseteq M$ angewendet werden, um alle mit dieser Strategie ableitbaren Klauseln zu erhalten. Wir können also immer in endlich vielen Schritten überprüfen, ob die leere Klausel aus Γ mit $\mathcal{R}_M$ ableitbar ist.

Wir müssen jetzt nur noch zeigen, dass die M-Resolution vollständig im Bezug auf semantische Bäume ist. Damit können wir genau dann aus den Klauseln eines Maslov-Satzes in Skolem-Normalform die leere Klausel ableiten, wenn die Formel unerfüllbar ist.

Satz 4.52. *$\mathcal{R}_M$-Resolution ist vollständig im Bezug auf semantische Bäume.*

Beweis. Sei $\Gamma \subseteq M$ eine unerfüllbare Klauselmenge. Nach Lemma 4.43 gibt es eine endliche Teilmenge $HB_0 \subseteq HB_\Gamma$, sodass alle semantischen Bäume basierend auf HB_0 abgeschlossen sind. Jeder semantische Baum, der auf einer Obermenge von HB_0 basiert, ist dann erst recht abgeschlossen. Sei nun $\alpha_1, \alpha_2, \ldots$ eine Enumeration von HB_Γ für die gilt

1. Tiefere Literale kommen vor weniger tiefen.
2. Innerhalb von Literalen einer Tiefe kommen zuerst die uniformen Literale.

Da HB_0 endlich ist, gibt es ein $i \in \mathbb{N}$, sodass $HB_0 \subseteq \alpha \coloneqq \{\alpha_1, \ldots, \alpha_i\}$. Sei nun T der semantische Baum basierend auf $\alpha_1, \ldots, \alpha_i$ in dieser Reihenfolge. T ist abgeschlossen.

Sei v ein Entscheidungsknoten von T bezüglich Γ und C, D Klauseln (o.B.d.A. ohne gemeinsame Variablen), die an den Kinderknoten v_1, v_2 von v scheitern. Da $\mathcal{R}$-Resolution vollständig ist im Bezug auf semantische Bäume, gibt es eine Klausel

$E \in \mathcal{R}(\Gamma)$, welche in v scheitert. Da bei der Resolution aus jeweils C, D mindestens ein Literal faktorisiert wurde, hat E maximal zwei Literale. Indem wir zwischen 3 Fällen unterscheiden zeigen wir nun, dass es eine Substitution π gibt, sodass $\pi(E) \in \mathcal{R}_M(\Gamma)$, also sichergestellt werden kann dass, $\pi(E)$ eine Tiefe ≤ 1 hat und uniform ist.

1. E hat die Tiefe 0: Dann ist E sofort nach Definition ein M-Resolvent und $\pi = id$.

2. E hat die Tiefe 1: Sei P die Primformel auf den Kantenmarkierungen der Kanten $v \to v_1, v_2$. P muss ein Funktionssymbol enthalten, da in T zuerst weniger tiefe Literale kommen und E ein Funktionssymbol enthält und E vor v_1, v_2 scheitert. Dadurch, dass die Argumente von P nicht nur Konstantensymbole sein können, müssen bei der Resolution in der Menge, die unifiziert wurde, die Literale mindestens ein Funktionssymbol enthalten. Da C, D nun aber uniform sind, gibt es deshalb einen Funktionsterm t, so dass alle anderen Funktionsterme in E, kongruent zu t sind.

   Gilt bereits $E \in M$, dann setzen wir $\pi = id$ und sind fertig. Wir beschäftigen uns nun mit dem Fall, dass $E \notin M$ und konstruieren ein π, sodass $\pi(E) \in M$. Da E an v scheitert, gibt es eine Substitution τ, sodass alle Literale in $\tau(E)$ negiert auf dem Weg zu v vorkommen. Entweder $\tau(E)$ ist uniform oder wir konstruieren eine Substitution θ, sodass $\theta(E)$ uniform ist (durch $\theta(t)$) und auch alle Literale in $\theta(E)$ auf dem Weg zu v negiert vorkommen. Da E nicht uniform ist, gibt es Variablen $y_1, \ldots, y_k$, die keine Argumente von t sind und nur als Argumente von Primformeln vorkommen. Wir setzen $\theta(y_i) = c$ für ein fixes Konstantensymbol aus dem Herbrand-Universum, falls $\tau(y_i)$ weder kongruent zu $\tau(t)$, noch konstant, noch ein Argument von $\tau(t)$ ist und für die restlichen Variablen $\theta(x) = \tau(x)$. Nun ist $\theta(E)$ uniform und alle Literale in $\theta(E)$ kommen auf Grund der Reihenfolge α in negierter Form auf dem Weg zu v vor. Ist nämlich ein Literal L aus $\tau(E)$ nicht uniform, dann ist jedoch das zugehörige Literal L' in $\theta(E)$ uniform und kommt auf dem Weg zu v vor (sogar vor L). Ist ein Literal $L \in \tau(E)$ uniform, dann ist das zugehörige Literal L' in $\theta(E)$ entweder gleich L oder $d(L') < d(L)$ und damit kommt L' auch vor L auf dem Weg zu v.

   Sei ab hier θ die Substitution, sodass $\theta(E)$ uniform ist und alle Literale negiert auf dem Weg zu v vorkommen. Wir können θ noch nicht als unsere abschließende Substitution wählen, da möglicherweise die Tiefe von $\theta(E) > 1$ ist. Wir setzen π daher wie folgt. $\pi(y) = \theta(y)$, falls $\theta(y)$ ein Konstantensymbol ist. Ist $s = t$ oder ist s ein Argument von t, dann setzen wir $\pi(y) = s$, falls $\theta(y) = \theta(s)$. Alle anderen Variablen werden auf sich selbst abgebildet. Damit ist $\pi(E)$ uniform und in M. Auch scheitert $\pi(E)$ in v, denn $\theta(\pi(E)) = \theta(E)$. Damit ist $\theta(E) \in M$ der gesuchte Resolvent.

3. E hat eine Tiefe > 1: Wir zeigen, dass dies nicht möglich ist. Sei θ der allgemeinste Unifikator, der bei der Resolution von C, D verwendet wurde. Angenommen es wurden zwei Literale der Tiefe 1 faktorisiert, dann kann für keine Variable x, $\theta(x)$ ein funktionaler Term sein, da die Literale uniform sind. Der Anstieg der Tiefe (C, D müssen beide Tiefe 1 haben) kann also nur dadurch

passiert sein, dass entweder C oder D Literale γ, β hat, wobei γ keine Funktionsterme hat, β jedoch schon und γ faktorisiert wird und β nicht. Hier ein Beispiel für diesen Fall:

$$\frac{H(x), R(f(x)) \quad \neg H(f(y))}{R(f(f(y))} \ \{x \mapsto f(y)\}.$$

Auf Grund der Uniformität von C, D, beinhaltet β alle Variablen von γ. Da $d(\beta) > d(\gamma)$ folgt, dass $d(\theta(\beta)) > d(\theta(\gamma))$. Sei nun π eine Substitution, sodass alle Literale in $\pi(E)$ negiert auf dem Weg zu v vorkommen und die Kantenmarkierungen $v \mapsto v_1$ und $v \mapsto v_2$, $\pi(\theta(\gamma))$ bzw. $\neg\pi(\theta(\gamma))$ sind (π existiert, weil θ ein allgemeinster Unifikator ist). Da $\theta(\beta) \in E$ ist auch $\pi(\theta(\beta)) \in \pi(E)$, also ist $d(\pi(E)) > d(\pi(\theta(\gamma)))$. Das ist aber ein Widerspruch dazu, dass nach der Reihenfolge α weniger tiefe Literale zuerst kommen, denn E scheitert an v oder davor, also tauchen die Literale von $\pi(E)$ nicht erst nach v als Kantenmarkierung auf, $\pi(\theta(\gamma))$ jedoch schon. Folglich kann die Tiefe $d(E)$ nicht > 1 sein.

□

Korollar 4.53. *M-Resolution ist vollständig und korrekt.*

Beweis. M-Resolution ist vollständig im Bezug auf semantische Bäume, also ist M-Resolution vollständig. Die Korrektheit von M-Resolution folgt direkt aus dem Lemma 4.35 zur Korrektheit von $\mathcal{R}$-Resolution. □

4.5.4 Entfernung der freien Variablen

Auch die Maslov-Klasse ist abgeschlossen unter Hinzufügung von äußeren $\exists$-Quantoren. Wir können also mit der Abbildung *removeFreeVariables* 2.9 jede Maslov-Formel
auf einen erfüllbarkeitsäquivalenten Maslov-Satz reduzieren.

4.5.5 Entscheidungsalgorithmus für Maslov-Formeln

Wir können nun die Resultate aus den vorherigen Abschnitten nutzen, um die Entscheidbarkeit der Maslov-Formeln zu zeigen. Sei ab hier σ eine Signatur mit abzählbar unendlich vielen Konstanten und Funktionssymbolen. Wir brauchen diesen Symbolvorrat für die Skolemisierung.

Satz 4.54 (Theorem von Maslov)**.** *Sat*($Form_M$) *ist entscheidbar.*

Beweis. Wir betrachten eine beliebige Formel $\tilde{\varphi} \in Form_M$. Wir können aus φ die freien Variablen entfernen und dann skolemisieren, um einen erfüllbarkeitsäquivalenten Satz in Skolem-Normalform zu erhalten. Sei

$$\varphi := skolemize(removeFreeVariables(\tilde{\varphi})).$$

Wir fassen φ als Klauselmenge Γ auf.

Dadurch, dass Maslov-Formeln nach Definition keine Funktionssymbole enthalten, entstehen bei der Skolemisierung höchstens Terme mit einer Tiefe ≤ 1. Außerdem hängen die bei der Skolemisierung hinzugefügten Skolem-Funktionen alle von denselben Argumenten ab. Folglich ist $\Gamma \subseteq M$.

Im vorherigen Abschnitt haben wir die Vollständigkeit und Korrektheit von M-Resolution gezeigt. Das heißt wir können aus Γ mit M-Resolution genau dann die leere Klausel ableiten, wenn Γ bzw. φ bzw. $\tilde{\varphi}$ unerfüllbar ist.

Da wir im Beweis zur Vollständigkeit der M-Resolution 4.45 nicht über den Zeichenvorrat von Γ hinausgehen, abgesehen von einem Konstantensymbol, wenn Γ keins enthält, können wir uns auf M-Klauseln beschränken über Symbole in Γ. Diese Menge an M-Klauseln ist dann endlich, abgesehen von Umbenennungen von Variablen. Wir können daher immer durch eine endlich-fache Anwendung der M-Resolutionsstrategie auf Γ, alle möglichen, aus Γ ableitbaren M-Klauseln, ableiten.

Der Entscheidungsalgorithmus für $Sat(Form_M)$ besteht also nur darin bei Eingabe φ die freien Variablen aus φ zu entfernen und dann φ zu skolemisieren, danach alle ableitbaren M-Klauseln aus Γ (Die Klauselmenge von φ) zu bestimmen und abschließend zu überprüfen, ob die leere Klausel abgeleitet wurde. All diese Schritte sind berechenbar, also ist $Sat(Form_M)$ berechenbar. Im Folgenden ist dieser Algorithmus nochmals mit Pseudocode zusammengefasst. Um die angesprochene Endlichkeit zu garantieren, sollen mit $\mathcal{R}_M(\Gamma)$ nur Klauseln mit Zeichenvorrat aus Γ resolviert werden (oder mit einem zusätzlichem Konstantensymbole, wenn es in Γ keine gibt) und die Resolventen sollen so gebildet werden, dass die Variablenbenennung standardisiert ist.

Algorithmus 7: *isSat*

Eingabe : $\langle\varphi\rangle$, $\varphi \in Form_M$
Ausgabe: $\varphi \in Sat(Form_M)$

```
1 φ ← skolemize(removeFreeVariables(φ))
2 //Sei Γ die Klauselmenge von φ
3 while Γ ≠ R_M(Γ) do
4 |   Γ ← R_M(Γ)
5 if leere Klausel ∈ Γ then
6 |   return False
7 else
8 |   return True
```

□

5 Implementierung

Um das theoretisch vermittelte Wissen zur Entscheidbarkeit der Maslov,Herbrand und L_2 Formeln anwendbar zu machen, wurden die 3 jeweiligen Entscheidungsalgorithmen in Python implementiert. In diesem Kapitel wird die Implementierung dieser kurz beschrieben und für konkrete Formeln, durch Anwendung der Algorithmen, entschieden, ob diese erfüllbar sind.

Die gewählten Beispiele sind allesamt sehr klein gehalten, um das Ergebnis als Mensch noch nachvollziehen zu können. Des Weiteren sind die Laufzeiten allesamt exponentiell oder gar für den Algorithmus für L_2 sogar doppelt exponentiell, so dass die Anwendung der Algorithmen auf größere Formeln in der Praxis nicht sinnvoll erscheint.

Das Kapitel gliedert sich wie folgt. Wir zeigen zunächst, welche Repräsentation für prädikatenlogische Formeln gewählt wurde und geben einen Überblick über wichtige verwendete Hilfsfunktionen. Danach stellen wir die Module vor, in denen die jeweiligen Entscheidungsalgorithmen implementiert wurden und geben jeweils Beispiele für erfüllbare und nicht erfüllbare Formeln, die der jeweilige Algorithmus entscheidet.

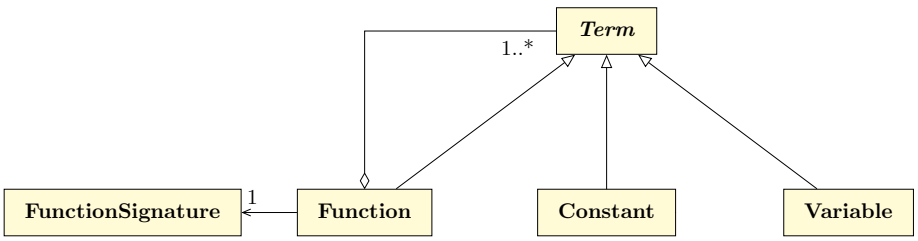
Dieses Kapitel hat nicht den Anspruch die Implementierung im Detail zu erläutern. In diesem Kapitel wird lediglich eine grobe Übersicht über die implementierten Module und deren Funktionalitäten gegeben. Wer sich für die Implementierungsdetails interessiert, kann diese direkt im Code nachlesen, da dieser ausreichend dokumentiert ist.

Alle Beispiele, die im Folgenden gezeigt werden, sind in dem Modul **Examples.py** implementiert. Um die Ergebnisse tatsächlich zu sehen, muss das Modul nur ausgeführt werden.

5.1 Repräsentation

Die Repräsentation einer prädikatenlogischer Formel auf rein syntaktischer Ebene ist durch das Modul **Formula.py** implementiert.

Um eine prädikatenlogische Formeln repräsentieren zu können, müssen wir zunächst Terme repräsentieren können. Dies ist durch die Klasse **Term** umgesetzt, die die Unterklassen **Function**,**Variable**,**Constant** besitzt, welche jeweils funktionale Terme, Variablen oder Konstantensymbole repräsentieren. Dabei ist die Repräsentation strikt nach der Definition eines Terms umgesetzt, indem wir **Term** als einen rekursiven Datentyp implementiert haben. Folgendes UML-Diagramm zeigt dies.

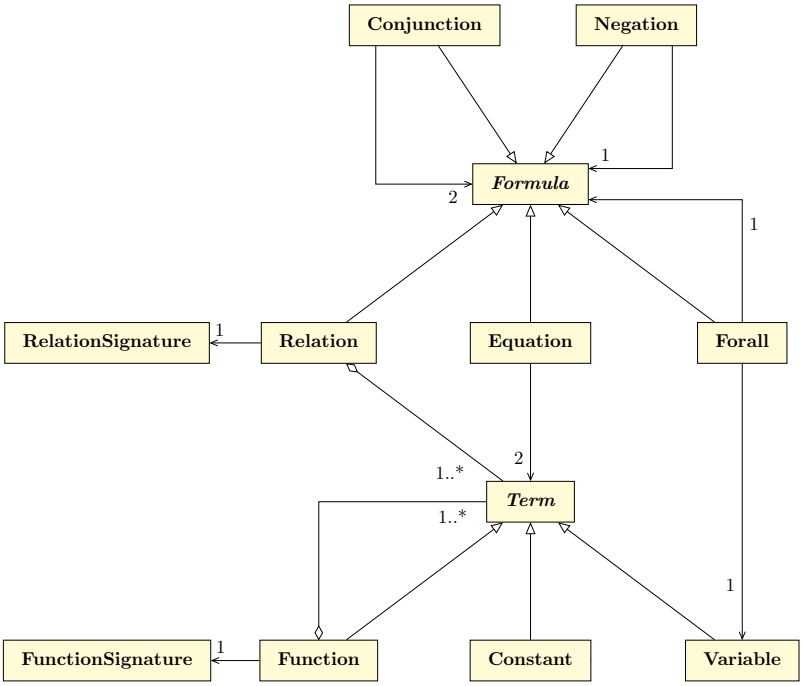


Ein Term implementiert folgende Methoden:

1. **getLength**: Liefert die Länge des Terms, wenn dieser als String repräsentiert würde.
2. **getVariables**: Liefert alle Variablen, die in diesem Term vorkommen.
3. **getFunctions**: Liefert aller Funktionsterme, insbesondere alle verschachtelten, die in diesem Term vorkommen.
4. **getConstants**: Liefert alle Konstantensymbole, die in diesem Term vorkommen.
5. **applySubstitution**: Liefert den Term der entsteht, wenn eine gegebene Substitution auf diesen angewendet wird.
6. **getDepth**: Liefert die Tiefe dieses Terms.

Jede Instanz von **Function** speichert eine Instanz der Klasse **FunctionSignature** als Attribut, welche das zugehörige Funktionssymbol und die Stelligkeit abspeichert.

Außerdem wurden für die verschiedenen Term-Klassen die Methoden **eq** und **hash** überschrieben, sodass wir mit Termen wie mit primitiven Datentypen umgehen können.
Die Repräsentation einer Formel ist durch die Klasse **Formula** implementiert. **Formula** hat die Unterklassen **Forall,Conjunction,Negation,Relation,Equation**, welche jeweils Formeln mit äußerem ∀-Quantor, Konjunktionen, Negationen, relationale Primformeln oder Gleichheitsformeln, repräsentieren. Wie bei den Termen ist auch der Datentyp **Formula** rekursiv, strikt nach der Definition einer prädikatenlogischen Formel, implementiert. Folgendes UML-Diagramm zeigt dies.



Jede Instanz von **Relation** hat als ein Attribut eine Instanz der Klasse **RelationSignature**, welche sowohl das zugehörige Relationssymbol als auch die Stelligkeit speichert.

Eine Formel implementiert folgende Methoden:

1. **getLength**: Liefert die Länge der Formel, wenn man diese als String repräsentiert.
2. **getFreeVariables**: Liefert die Menge aller freien Variablen, die in der Formel vorkommen.
3. **getVariables**: Liefert die Menge aller Variablen, die in der Formel vorkommen.
4. **getFunctions**: Liefert die Menge aller Funktionsterme, die in der Formel vorkommen.
5. **getFunctionSignatures**: Liefert die Menge aller Funktionssignaturen, die in der Formel vorkommen.
6. **getConstants**: Liefert die Menge aller Konstantenterme, die in der Formel vorkommen.
7. **containsEquality**: Liefert, ob die Formel ein Gleichheitszeichen enthält.
8. **getRelations**: Liefert die Menge aller Teilformeln dieser Formel, die relationale Primformeln sind.
9. **getRelationSignatures**: Liefert die Menge aller Relationssignaturen (Instanzen der Klasse **RelationSignature**, die Relationssymbol und Stelligkeit speichert), die in der Formeln vorkommen.
10. **getLiterals**: Liefert die Menge aller Literale, die in der Formel vorkommen.
11. **applySubstitution**: Liefert die Formel, die entsteht, wenn eine gegebene Substitution auf diese Formel angewendet wird.
12. **isPrimeForm**: Liefert, ob diese Formel eine Primformel ist.
13. **isLiteral**: Liefert, ob diese Formel ein Literal ist.

Auch hier sind die Methoden **eq** und **hash** überschrieben, sodass Formeln wie primitive Datentypen behandelt werden können.

5.2 Hilfsmethoden

Allgemeine Funktionalitäten bezüglich Termen und Formeln, die in mehreren Entscheidungsalgorithmen benötigt werden, sind in **utility.py** implementiert. Die wichtigsten Methoden sind im Folgenden explizit angegeben.

1. **removeFreeVariables**: Liefert die Formel, die entsteht, wenn zu einer gegebenen Formel die freien Variablen entfernt werden, indem die Gesamtformel durch zusätzlich $\exists$-Quantoren umschlossen wird.
2. **skolemize**: Liefert die Formel, die entsteht, wenn eine gegebene Formel skolemisiert wird (Implementiert $skolemize_{Alg}$ 5).
3. **getMostGeneralUnifier**: Liefert zu einer Menge von Paaren von Termen, einen allgemeinsten Unifikator (Implementiert *isUnifiable* 4).

5.3 L_2 Entscheidungsalgorithmus Implementierung

Der Entscheidungsalgorithmus für L_2 ist in dem Modul **L2Decision.py** implementiert. Die konkrete Methode, die entscheidet, ob ein eine gegebene Formel aus L_2 erfüllbar ist, heißt **isSatL2**. Es wurde der in Pseudocode beschriebene Algorithmus *isSat* 3 bezüglich L_2 Formeln implementiert.

Das Modul enthält die Klassen **FiniteStructure** und **FiniteInterpretation**, welche eine endliche Struktur bzw. endliche Interpretation repräsentieren. **FiniteInterpretation** hat unter Anderem die Methode **models**, welche zu einer gegebenen Formel entscheidet, ob diese durch die Interpretation erfüllt wird.

Der gesamte Entscheidungsalgorithmus für L_2, ist in **isSatL2** umgesetzt. Die Methode wirft einen Fehler, wenn die eingegebene Formel nicht in L_2, sonst werden alle möglichen Interpretationen bis zu einer gewissen Grenze (auch manuell festlegbar) iteriert und überprüft, ob diese die eingegebene Formel erfüllen.

isSatL2 wurde beispielhaft auf folgende Formeln angewendet. In **Examples.py** sind diese Beispiele implementiert (in gleicher Reihenfolge. Die Nummerierung ist auch im Code zu sehen). R, E sind hier Relationssymbole.

Nummer	Formel	Ausgabe
1	$\forall x \forall y \forall z R(x, y, z)$	Nicht in L_2
2	$\exists x \exists y (E(x, y) \wedge \exists x (E(y, x) \wedge \exists y (E(x, y) \wedge \exists x E(y, x))))$	Erfüllbar
3	$R(x) \wedge \neg R(y)$	Erfüllbar
4	$R(x) \wedge \neg R(x)$	Nicht erfüllbar
5	$(\exists x R(x)) \wedge (\forall x \neg R(x))$	Nicht erfüllbar

Anmerkung zu 4,5: Da durch die doppelt exponentielle Laufzeit von **isSatL2** hier nicht alle Strukturen bis zu der im Satz von Mortimer 3.16 bestimmten Grenze iteriert werden können (zumindest nicht in unserer Lebenszeit), wurde der Suchraum auf Strukturen bis zu einer Universumsgröße von 10 beschränkt.

5.4 Herbrand Entscheidungsalgorithmus Implementierung

Der Entscheidungsalgorithmus für die Herbrand-Formeln ist in dem Modul **HerbrandDecision.py** implementiert. Die konkrete Methode, die Herbrand-Formeln entscheidet, heißt **isSatHerbrand**. Es wurde der in Pseudocode beschriebene Algorithmus *isSat* 6 bezüglich Herbrand-Formeln, implementiert.

Die für **isSatHerbrand** wichtigen Hilfsmethoden, die auch in diesem Modul implementiert sind, sind die Folgenden:

1. **isHerbrand**: Prüft, ob eine gegebene Formel eine Herbrand-Formel ist. **isSatHerbrand** wirft nämlich einen Fehler, falls die eingegebene Formel keine Herbrand-Formel ist.

2. **renameLiterals**: Wandelt eine gegebene Herbrand-Formel so um, dass alle Literale verschiedene Variablennamen besitzen. Liefert diese transformierte Formel dann zurück.

isSatHerbrand wurde beispielhaft auf folgende Formeln angewendet. In **Examples.py** sind diese Beispiele implementiert (in gleicher Reihenfolge. Die Nummerierung ist auch im Code zu sehen). R, H sind hier Relationssymbole, c, d sind Konstantensymbole und f, g sind Funktionssymbole.

Nummer	Formel	Ausgabe
1	$\exists x \neg \forall y R(x, y)$	Nicht in $Form_{HB}$
2	$\forall x(H(x, c) \wedge R(x) \wedge \neg R(x))$	Nicht erfüllbar
3	$\forall x \forall y(\neg R(x, g(c, d)) \wedge H(x, y) \wedge R(f(c), y))$	Nicht erfüllbar
4	$\forall x(R(f(x)) \wedge \neg R(g(x)))$	Erfüllbar

5.5 Maslov Entscheidungsalgorithmus Implementierung

Der Entscheidungsalgorithmus für die Maslov-Formeln ist in dem Modul **Maslov-Decision.py** implementiert. Die konkrete Methode, die Maslov-Formeln entscheidet, heißt **isSatMaslov**. Es wurde der in Pseudocode beschriebene Algorithmus *isSat* 7 bezüglich Maslov-Formeln implementiert.

Das Modul enthält folgende Klassen, die für die Methode **isSatMaslov** benötigt werden.

1. **Clause**: Diese Klasse repräsentiert eine Klausel, also eine Menge von Literalen. Diese Klasse implementiert unter anderem die Methode **isMaslovClause**, die liefert, ob es sich bei der Klausel um eine M-Klausel handelt.

2. **R_Resolution**: Diese Klasse ist eine Sammlung von Methoden bezüglich der Robinson-Resolution. Die Methode **getRResolvents** liefert alle Robinson-Resolventen (bis auf Umbenennung) zu zwei Klauseln.

Außerhalb einer Klasse, aber innerhalb von **MaslovDecision.py** gibt es noch unter Anderem die Methode **isMaslovForm**, die für eine gegebene Formel, entscheidet, ob diese eine Maslov-Formel ist. Eine weitere wichtige Methode ist **getMResolvents**, die bis auf Umbenennung von Variablen alle M-Klauseln zu zwei gegebenen Maslov-Klauseln liefert. Die Methode **isSatMaslov** ist die vollständige Implementierung von *isSat* 7, wobei die Methode noch einen Fehler wirft, falls die eingegebene Formel keine Maslov-Formel ist.

isSatMaslov wurde beispielhaft auf folgende Formeln angewendet. In **Examples.py** sind diese Beispiele implementiert, in gleicher Reihenfolge. Die Nummerierung ist auch im Code zu sehen. L, R, T, Q sind hier Relationssymbole und c ist ein Konstantensymbol.

Nummer	Formel	Ausgabe
1	$\forall x \exists y \forall z(L(x, y, z) \wedge R(x))$	Nicht in $Form_M$
2	$((R(x, c) \vee R(x, c)) \wedge (\neg R(x, c) \vee \neg R(x, c))$	Nicht erfüllbar
3	$\forall x \exists y((T(y) \vee T(y)) \wedge (Q(x) \vee Q(x)) \wedge (\neg Q(x) \vee \neg T(y)))$	Nicht erfüllbar
4	$\forall x \exists y((\neg Q(x) \vee T(y)) \wedge (Q(x) \neg T(y)))$	Erfüllbar

6 Zusammenfassung und Ausblick

Wir haben im Verlaufe dieser Arbeit die Entscheidbarkeit von 3 verschiedenen Klassen der Prädikatenlogik gezeigt.

Dabei haben wir zunächst die Entscheidbarkeit von L_2 gezeigt, der Klasse mit 2 Variablen ohne Konstantensymbole und Funktionssymbole aber dafür mit Gleichheit. In der Beweisführung haben wir gezeigt, dass L_2 die endliche Modelleigenschaft besitzt, indem wir gezeigt haben, dass sich Formeln aus L_2 auf die Scott-Normalform reduzieren lassen können. Für Formeln in der Scott-Normalform haben wir dann ein konkretes endliches Modell durch geschicktes Setzen von 2-Tables definiert, falls diese erfüllbar waren.

Eine weitere Klasse für welche die Entscheidbarkeit gezeigt wurde, ist die der Herbrand-Formeln. Das sind Formeln in Pränexnormalform deren quantorenfreier Teil eine Konjunktion von Literalen ist. Im Gegensatz zu L_2 haben wir hier aber keine Gleichheit zugelassen. Um die Entscheidbarkeit zu zeigen haben wir zunächst die Entscheidbarkeit von einem speziellen Unifikationsproblem gezeigt. Wir konnten dann die Entscheidbarkeit von Herbrand-Formeln auf dieses Unifikationsproblem zurückführen.

Zuletzt haben wir die Entscheidbarkeit der Maslov-Klasse gezeigt. Das sind Formeln in Pränexnormalform, ohne Funktionssymbole, deren quantorenfreier Teil eine Konjunktion von Krom-Klauseln ist. Der Quantorenpräfix hat die Form $\exists^*\forall^*\exists^*$. Auch hier haben wir wieder die Gleichheit ausgeschlossen. Die Beweisführung hier war etwas aufwändiger. Wir haben dafür Resolution für prädikatenlogische Klauseln eingeführt und mit Hilfe von semantischen Bäumen gezeigt, dass Resolution für eine gewisse Klauselklasse, die M-Klauseln, vollständig ist. Da die Klauseln eines skolemisierten Maslov-Satzes gerade M-Klauseln sind, und die Menge der M-Klauseln über einem endlichen Symbolvorrat bis auf Variablenumbenennung endlich ist, konnten wir durch endlich viele Resolutionen feststellen, ob die leere Klausel ableitbar ist, also ob die Formel erfüllbar ist.

In dieser Arbeit wurden die Klassen nur auf die Entscheidbarkeit hin untersucht, es bietet sich aber auch an die Klassen bezüglich ihrer Komplexität zu untersuchen. Es lässt sich zum Beispiel zeigen, dass L_2, NEXPTIME vollständig ist [4]. Auch kann gezeigt werden, dass das Problem der Entscheidbarkeit der Maslov-Formeln, $\mathrm{DTime}(2^{\mathcal{O}(n)})$ vollständig ist [2]. Es lässt auch zeigen, dass das hier behandelte Unifikationsproblem P-vollständig ist und damit dann auch die Entscheidbarkeit der Herbrand-Formeln [2].

Neben den entscheidbaren Klassen, die hier behandelt wurden gibt es noch interessante weitere. Zum Beispiel ist die Aanderaa-Lewis Klasse entscheidbar. Das ist die Klasse der Formeln in Pränexnormalform ohne Gleichheit, Konstanten- oder Funktionssymbole, deren quantorenfreier Teil eine Konjunktion von Krom-Klauseln ist und der Quantorenpräfix die Form $\forall\exists\forall$ besitzt. Die Entscheidbarkeit kann hier mittels linearer diophantischer Gleichungen gezeigt werden [2]. Ein weiteres Beispiel für eine entscheidbare Klasse, ist die Löwenheim-Klasse. Das sind Formeln in

Pränexnormalform mit beliebigem Quantorenpräfix mit ausschließlich unären Relationssymbolen, keinen Konstanten oder Funktionssymbolen. Gleichheit ist erlaubt. Die Entscheidbarkeit der Löwenheim-Klasse wird darüber gezeigt, dass diese Klasse die endliche Modelleigenschaft besitzt [2].

Es lassen sich aber auch die hier gezeigten Klassen teilweise noch erweitern. So können wir durch einen im Kern gleichen, aber aufwändigeren, Beweis zeigen, dass L_2 erweitert um Konstantensymbole ebenfalls noch entscheidbar ist.

Literatur

[1] Franz Baader, Wayne Snyder, Paliath Narendran, Manfred Schmidt-Schauß und Klaus U. Schulz. "Unification Theory". In: *Handbook of Automated Reasoning (in 2 volumes)*. Hrsg. von John Alan Robinson und Andrei Voronkov. Elsevier und MIT Press, 2001, S. 445–532. DOI: 10.1016/b978-044450813-3/50010-2. URL: https://doi.org/10.1016/b978-044450813-3/50010-2.

[2] Egon Börger, Erich Grädel und Yuri Gurevich. *The Classical Decision Problem*. Perspectives in Mathematical Logic. Springer, 1997.

[3] Evgeny Kruglov und Christoph Weidenbach. *Normal Forms and Skolemization (Traditional)*. Vorlesungsskript. Max Planck Institut Informatik. URL: http://rg1-teaching.mpi-inf.mpg.de/autrea-ss10/script/lecture10.pdf.

[4] Erich Grädel, Phokion G. Kolaitis und Moshe Y. Vardi. "On the decision problem for two-variable first-order logic". In: *Bull. Symb. Log.* 3.1 (1997), S. 53–69. DOI: 10.2307/421196. URL: https://doi.org/10.2307/421196.

[5] Erich Grädel und Martin Otto. "On Logics with Two Variables". In: *Theor. Comput. Sci.* 224.1-2 (1999), S. 73–113. DOI: 10.1016/S0304-3975(98)00308-9. URL: https://doi.org/10.1016/S0304-3975(98)00308-9.

[6] *Herbrand-Universum*. Zuletzt zugegriffen am 16. Juni 2021. 2021. URL: https://de.wikipedia.org/wiki/Herbrand-Universum.

[7] Alberto Martelli und Ugo Montanari. "An Efficient Unification Algorithm". In: *ACM Trans. Program. Lang. Syst.* 4.2 (1982), S. 258–282. DOI: 10.1145/357162.357169. URL: https://doi.org/10.1145/357162.357169.

[8] *Resolution (Logik)*. Zuletzt zugegriffen am 16. Juni 2021. 2021. URL: https://de.wikipedia.org/wiki/Resolution_(Logik).

[9] Sebastian Rudolph. *Foundations of Logic Programming - Unification*. Vorlesungsskript. TU Dresden, 2019. URL: https://www.inf.tu-dresden.de/content/institutes/ki/cl/study/winter09/flp/slides/2.pdf.

[10] Javier Esparza und Uwe Schöning. *Herbrand Theory*. Vorlesungsskript. Technische Universität München. URL: https://www21.in.tum.de/teaching/logik/SS16/Slides/herbrand.pdf.

[11] Heribert Vollmer und Thorsten Kluge. *Logik und formale Systeme*. Vorlesungsskript. 2019.

[12] James Worrell. *Herbrand's Theorem and Ground Resolution*. Vorlesungsskript. Oxford University. URL: https://www.cs.ox.ac.uk/people/james.worrell/lecture11-2015.pdfbaa.

[13] James Worrell. *Normal Forms for First-Order Logic*. Vorlesungsskript. Oxford University. URL: https://www.cs.ox.ac.uk/people/james.worrell/lecture10-2015.pdf.