Übungsblatt 9 mit Lösungen

Abgabetermin: Montag, der 08. Juli 2024 um 14:30

Hausaufgabe 3 (Entscheidbarkeit) 2+2+2 Punkte

Sei σ_{UE} wie in der Vorlesung definiert. Beweisen Sie ob die folgenden Probleme, entscheidbar oder unentscheidbar sind. Beweisen Sie außerdem ob die Probleme oder ihre Komplemente semi-entscheidbar sind.

a)

AllBew($L(\sigma_{\text{UE}})$)

Eingabe: Endliche Formelmenge $\Gamma \subseteq L(\sigma_{\text{UE}})$.

Problem: Gilt $\Gamma \models \psi$, für alle $\psi \in L(\sigma_{\text{UE}})$?

b)

ExErf($L(\sigma_{\text{UE}})$)

Eingabe: Formel $\varphi \in L(\sigma_{\text{UE}})$.

Problem: Ist $\exists x \varphi$ erfüllbar?

c)

Folg($L(\sigma_{\text{UE}})$)

Eingabe: Formel $\varphi \in L(\sigma_{\text{UE}})$.

Problem: Gilt $\forall x \varphi \vdash \exists x \varphi$?

Lösung:

a) Nach dem Vollständigkeitssatz gilt $\Gamma \models \psi$ genau dann wenn $\Gamma \vdash \psi$. Außerdem gilt das Γ erfüllbar ist genau dann wenn es widerspruchsfrei ist. Da Γ endlich ist, ist Γ widersprüchlich, genau dann wenn $\bigwedge_{\varphi \in \Gamma} \varphi$ unerfüllbar ist. Also können AllBew($L(\sigma_{\text{UE}})$) auf das Problem UErf($L(\sigma_{\text{UE}})$) reduzieren mit der Funktion $f(\Gamma) := \bigwedge_{\varphi \in \Gamma} \varphi$. Diese Funktion ist offensichtlich berechenbar und korrekt nach der obigen Beobachtung. Demnach ist das Problem AllBew($L(\sigma_{\text{UE}})$) nicht entscheidbar aber semi-entscheidbar, da σ_{UE} endlich und damit semi-entscheidbar ist. Das Komplement ist nicht semi-entscheidbar.

b) Für alle $\varphi \in L(\sigma_{\text{UE}})$ und alle $x \in \text{Var}$ gilt, dass φ erfüllbar genau dann wenn $\exists x \varphi$ erfüllbar. Also ist ExErf($L(\sigma)$) das gleiche Problem wie Erf($L(\sigma_{\text{UE}})$). Demnach ist das Problem nicht entscheidbar und nicht semi-entscheidbar. Allerdings ist das Komplement semi-entscheidbar.

c) Für alle $\varphi \in \mathrm{L}(\sigma_{\mathrm{UE}})$ gilt dass $\forall x\varphi \vdash \exists x\varphi$. Wir betrachten dafür die folgende Ableitung.

1. $\varphi\frac{x}{x} \vdash \varphi\frac{x}{x}$ (Vor)
2. $\forall x\varphi \vdash \varphi\frac{x}{x}$ ($\forall$L)
3. $\forall x\varphi \vdash \exists x\varphi$ ($\exists$R).

Das Problem ist entscheidbar und somit auch semi-entscheidbar, mit Hilfe eines Algorithmus der für alle gültigen Eingaben "ja" ausgibt.

Hausaufgabe 4 (Graphzusammenhang) 3 Punkte

Zeigen oder widerlegen Sie, dass es eine Formelmenge $\Phi \subseteq \mathrm{L}(\{E\})$ gibt, die besagt, dass ein Graph zusammenhängend ist.

Lösung:

Sei φ_{Graph} wie im Beweis vom Satz 5.48 eingeführt, die Formel die besagt dass eine Struktur ein Graph ist. Weiterhin sei $\psi''_k(x,y)$ wie in Beispiel 4.38 definiert, die Formel die besagt dass zwischen den Knoten x, y einen Pfad der Länge genau k gibt.

Angenommen, $\Phi \subseteq \mathrm{L}(\{E\})$ besagt, dass ein Graph zusammenhängend ist, dass heißt, für alle Graphen $\mathfrak{G}$ gilt

$$\mathfrak{G} \vDash \Phi \iff \mathfrak{G} \text{ ist zusammenhängend.}$$

Sei

$$\Phi' \coloneqq \Phi \cup \{\varphi_{\text{Graph}}\} \cup \left\{\neg\psi''_\ell(x,y) \;\middle|\; \ell \in \mathbb{N}\right\}.$$

Jede endliche Teilmenge von Φ' ist erfüllbar. Denn sei $\Gamma \subseteq \Phi'$ endlich. Wenn $\Gamma \cap \{\neg\psi''_\ell(x,y) \mid \ell \in \mathbb{N}\} = \emptyset$, so erfüllt jeder zusammenhängende Graph Γ. Sonst sei $\ell(\Gamma) \coloneqq \max\{\ell \in \mathbb{N} \mid \neg\chi_\ell \in \Gamma\}$ und $\mathfrak{P}_n$ der Pfad der Länge n mit Knoten $0, \dots n$ wie im Beweis vin Satz 5.49 definiert. Sei $\mathfrak{I} = (\mathfrak{P}_n, \mathfrak{b})$ eine $\{E\}$-Interpretation mit $n > \ell$ und $\mathfrak{b}(x) = 0, \mathfrak{b}(y) = n$. Dann gilt $\mathfrak{I} \models \Gamma$.

Also ist nach dem Endlichkeitssatz auch Φ' erfüllbar. Sei $\mathfrak{I} = (\mathfrak{G}, \mathfrak{b}) \models \Phi'$. Dann gilt $\mathfrak{G} \models \{\varphi_{\text{Graph}}\} \cup \Phi$, also ist $\mathfrak{G}$ ein zusammenhängender Graph. Sei $a \coloneqq \mathfrak{b}(x)$ und $b \coloneqq \mathfrak{b}(y)$ Weil $\mathfrak{I} \models \neg\chi_\ell$ für alle $\ell \in \mathbb{N}$, gibt es in $\mathfrak{G}$ keinen Weg von a nach b, denn jeder Weg zwischen zwei Knoten hat endliche Länge. Also ist $\mathfrak{G}$ nicht zusammenhängend. Dies ist ein *Widerspruch*, also kann eine solche Formelmenge nicht existieren.

Hausaufgabe 5 (Endloswortstruktur) 3+3 Punkte

Sei Σ ein endliches Alphabet. Ein *Endloswort* (auch ω-Wort genannt) ist ein unendliches Wort, also ein Wort dass zwar einen Anfang hat aber kein Ende hat. Wir stellen ein solches Wort durch eine Funktion $\omega\colon \mathbb{N} \setminus \{0\} \to \Sigma$ da und sagen das Wort hat an Stelle $i \geq 1$ den Buchstaben $\omega(i)$.
Für jedes Endloswort ω, sei $\mathfrak{A}_\omega$ die folgende σ_Σ-Struktur:

- Das Universum ist $A_\omega = \mathbb{N}$.
- Die Relation $\dot{\leq}^{\mathfrak{A}_\omega}$ ist die natürliche lineare Ordnung auf $\mathbb{N}$.
- Für jedes $a \in \Sigma$ ist $P_a \coloneqq \{i \in \mathbb{N} \mid \omega(i) = a\}$.

Eine solche Struktur nennen wie *Endloswortstruktur*.
Zeigen oder widerlegen Sie die folgenden Aussagen.

a) Es gibt eine Formel $\varphi_1 \in \mathrm{L}(\sigma_\Sigma)$, die besagt, dass eine abzählbare Struktur isomorph zu einer Endloswortstruktur ist. Also es gibt eine Formel $\varphi_1 \in \mathrm{L}(\sigma_\Sigma)$, sodass für alle abzählbaren Strukturen $\mathfrak{A}$ gilt,

$$\mathfrak{A} \models \varphi_1 \iff \text{es gibt ein Endloswort } \omega \text{ sodass } \mathfrak{A} \cong \mathfrak{A}_\omega.$$

b) Sei $a \in \Sigma$. Es gibt eine Formel $\varphi_2 \in \mathrm{L}(\sigma_\Sigma)$, die besagt, dass die Anzahl der a's des Endloswortes einer Endloswortstruktur endlich ist. Also, es gibt eine Formel $\varphi_2 \in \mathrm{L}(\sigma_\Sigma)$, sodass für alle $\mathfrak{A}_\omega$ gilt,

$$\mathfrak{A}_\omega \models \varphi_2 \iff \{i \in \mathbb{N} \mid \omega(i) = a\} \text{ ist endlich.}$$

Lösung:

a) Eine solche Formel existiert nicht. Wir nutzen die gleiche Idee wie für das Nichtstandardmodell der Arithmetik.

Angenommen eine solche Formel φ_1 existiert. Rekursiv definieren wir

$$\psi_0(x) \coloneqq \forall y(x \leq y)$$

und

$$\psi_{n+1}(x) \coloneqq \exists y(\psi_n(y) \wedge x \mathrel{\dot{\neq}} y \wedge y \mathrel{\dot{\leq}} x \wedge \forall z\,(z \leq y \vee x \leq z)),$$

also Formeln ψ_n die besagen dass x die Position n im Wort ist. Sei nun

$$\Phi \coloneqq \{\varphi_1\} \cup \{\neg\psi_n(x) \mid n \in \mathbb{N}\}.$$

Claim: Jede endliche Teilmenge $\Gamma \subseteq \Phi$ ist erfüllbar.

Sei $\mathfrak{A}_\omega$ eine Endloswortstruktur. Dann gilt $\mathfrak{A} \models \{\varphi_1\}$. Also gelte $\Gamma \cap \{\neg\psi_n(x) \mid n \in \mathbb{N}\} \neq \emptyset$ und sei $\ell(\Gamma) \coloneqq \max\{\ell \in \mathbb{N} \mid \neg\psi_\ell \in \Gamma\}$. Sei $\mathfrak{b}(x) \coloneqq \ell(\Gamma) + 1$, dann gilt $(\mathfrak{A}_\omega, \mathfrak{b}) \models \Gamma$. □

Nach dem Endlichkeitsatz ist also auch Φ erfüllbar. Nach dem Satz von Löwenheim und Skolem gibt es eine abzählbare Struktur, die Φ erfüllt. Sei $(\mathfrak{A}, \mathfrak{b}) \models \Phi$ eine solche abzählbare Struktur. Da $\mathfrak{A} \models \varphi_1$, existiert eine Funktion ω, sodass $\mathfrak{A} \cong \mathfrak{A}_\omega$. Sei π ein Isomorphismus von $\mathfrak{A}$ nach $\mathfrak{A}_\omega$. Dann gilt $\mathfrak{A}_\omega \models \psi_{\pi(\mathfrak{b}(x))}(\pi(\mathfrak{b}(x)))$. Nach dem Isomorphielemma gilt also auch $(\mathfrak{A}, \mathfrak{b}) \models \psi_{\pi(\mathfrak{b}(x))}$, dies ist ein Widerspruch zu $(\mathfrak{A}, \mathfrak{b}) \models \Phi$. Also kann eine solche Formel nicht existieren.

b) Eine solche Formel existiert. Die Idee ist, dass wenn es nur endlich viele a's gibt, dann gibt es eine endliche Position ab der es keine a's mehr gibt.

$$\varphi_2 \coloneqq \exists x \forall y \left((x \dot{\leq} y) \to \neg P_a(y) \right).$$

Programmieraufgabe 6 (Sequenzenregeln) 5 Punkte

- Die Abgabe der Programmieraufgabe erfolgt über **Speichern** oder **Abgabe** in VPL. Bis zur Abgabefrist könnt ihr so oft abgeben, wie ihr wollt. Wir bewerten nur die aktuellste Abgabe.
- Ihr könnt in **assignment.py** euren eigenen Code schreiben und dabei die von uns zur Verfügung gestellten Bibliotheken benutzen. Achtet allerdings darauf, keine Dateien zu löschen und die Header der Funktionen unverändert zu lassen.
- Nicht alle Importe sind möglich, manche Bibliotheken werden also einen Fehler wie z.B. `Module assignment tries to import numpy, which does not exist` liefern, wenn ihr versucht diese zu verwenden.
- Wir empfehlen, den Code mindestens einmal zu testen, mit **Ausführen** oder Strg+F11. Dies kann einige Sekunden dauern.
- Punkte und Code sind automatisch mit eurer Abgabegruppe synchronisiert.

In dieser und der folgenden Woche verifizieren wir Beweise des Sequenzenkalküls der Logik der 1. Stufe. Hierbei ist zu beachten, dass wir **nur** die Verifikation und **nicht** die Suche nach Beweisen automatisieren wollen. Für gegebene Folgen von Sequenzen inklusive angewandter Regeln überprüfen wir, ob diese unter Anwendung der Regeln gültige Ableitungen darstellen.

Schreiben Sie die Funktion `verify_axiom(sequent: Sequent) -> bool`,
welche als Eingabe ein Objekt der Klasse `Sequent` nimmt und als Wert `True` ausgibt, wenn die Sequenz mit einer Anwendung der (Vor)-Regel erhalten werden kann.

$$\frac{}{\texttt{sequent}}$$

Schreiben Sie die Funktionen
`verify_expansion(premise: Sequent, conclusion: Sequent) -> bool` und
`verify_and_left(premise: Sequent, conclusion: Sequent) -> bool`,
welche als Eingabe zwei Objekte der Klasse `Sequent` nehmen und als Wert `True` ausgeben, wenn die Sequenz `conclusion` mit einer Anwendung der (Erw)-Regel bzw. der $(\wedge L)$-Regel auf die Sequenz `premise` erhalten werden kann.

$$\frac{\texttt{premise}}{\texttt{conclusion}}$$

Schreiben Sie die Funktion `verify_and_right(first_premise: Sequent, second_premise: Sequent, conclusion: Sequent) -> bool`,
welche als Eingabe drei Objekte der Klasse `Sequent` nimmt und als Wert `True` ausgibt, wenn die Sequenz `conclusion` mit einer Anwendung der $(\wedge R)$-Regel auf die Sequenzen `first_premise` und `second_premise` erhalten werden kann.

$$\frac{\texttt{first\_premise}\quad\texttt{second\_premise}}{\texttt{conclusion}}$$

Wir implementieren in der nächsten Woche nur noch zwei weitere Regeln. Auf alle anderen Regeln werden wir bei den Beweisen, die wir verifizieren, verzichten.

Lösung:

- Die Abgabe der Programmieraufgabe erfolgt über **Speichern** oder **Abgabe** in VPL. Bis zur Abgabefrist könnt ihr so oft abgeben, wie ihr wollt. Wir bewerten nur die aktuellste Abgabe.
- Ihr könnt in **assignment.py** euren eigenen Code schreiben und dabei die von uns zur Verfügung gestellten Bibliotheken benutzen. Achtet allerdings darauf, keine Dateien zu löschen und die Header der Funktionen unverändert zu lassen.
- Nicht alle Importe sind möglich, manche Bibliotheken werden also einen Fehler wie z.B. `Module assignment tries to import numpy, which does not exist` liefern, wenn ihr versucht diese zu verwenden.
- Wir empfehlen, den Code mindestens einmal zu testen, mit **Ausführen** oder Strg+F11. Dies kann einige Sekunden dauern.
- Punkte und Code sind automatisch mit eurer Abgabegruppe synchronisiert.

Das Codegerüst von Blatt 10 enthält vollständige Implementierungen der Regeln.