Dokumentation STACK-Aufgaben

Andreas Steiger: 401-0261-G0L Analysis I HS2021

Jannik Kochert

3. Januar 2022

1 Allgemeines

Ein gutes Tutorial für STACK findet man unter `https://docs.stack-assessment.org/en/Authoring/Authoring_quick_start/`. Um sich damit vertraut zu machen, kann man einfach die ganzen Authoring Quick Start-Aufgaben machen.
Generelle Tipps für das Programmieren mit STACK:

- Generell alle Variablen mit mindestens zwei Buchstaben benennen, damit sie nicht eventuell von Maxima falsch ausgewertet werden. Man sollte auch nicht Variablen gleich wie schon belegte Ausdrücke wie "and" oder "or" oder ähnliches benenen.
- Alle Variablen sinnvoll benennen, also je nachdem, was sie tun soll.
- Hilfsvariablen benutzen, um Fehlerverfolgung bei der Erstellung der Aufgabe einfacher zu gestalten.
- Die Dokumentation von Maxima ist hier zu finden: `https://maxima.sourceforge.io/docs/manual/maxima_toc.html#SEC_Contents`. Jedoch gibt es keine direkte Suchfunktion, was das Suchen manchmal recht mühsam macht. Eine Dokumentation für STACK, wo auch gewisse interessante Dinge drinstehen, ist hier zu finden: `https://moodle-app2.let.ethz.ch/question/type/stack/doc/content/de/Erklaerung-STACK-3.pdf`

Zusätzlich dazu sollte man noch folgendes beachten:

- Im Folgenden werde ich versuchen, Aufgabenvariablen und allgemeine Variablen in kursiver Schrift (ans) und fest definierte Funktionen und konstanten aus Maxima/STACK mit normaler Schrift (ans) zu schreiben.
- Die Musterlösung kann man in "Allgemeines Feedback" unterbringen.
- Da man in der Musterlösung nichts berechnen lassen kann, muss man das in den Aufgabenvariablen bereits tun. Dies hilft, um die Musterlösung möglichst präzise zu schreiben. Diese Variablen werde ich in den folgenden Abschnitten als "Musterlösungsvariablen" bezeichnen.
- In STACK kann man sogenannte "Partial Response Trees" einstellen, mit denen man für teilweise richtige Lösungen trotzdem noch Punkte vergeben kann. Dieses Feature ist der primäre Grund, warum wir STACK eingesetzt haben. Im Folgenden werden die Partial Response Trees mit PRT abgekürzt.
- Falls man ein "teilweise richtig" für Folgefehler anzeigen lassen möchte, muss man einen PRT korrekt wie im obigen STACK-Tutorial aufsetzen und dann bei "Knoten 1 wenn FALSCH" den Score auf einen Wert > 0 setzen.
- Falls in der Aufgabenvorschau komische Zeilenumbrüche oder nichtkompilierender LaTeX-Code angezeigt werden: Man kann im Fragetext-Feld mit dem Button ganz links ein erweitertes Menü herunterfahren. Ganz unten, das fünfte Symbol von rechts namens "HTML" zeigt dann den HTML-Code an. Dort kann man dann alle Instanzen von -Tags und unnötige <pr>-Tags rauswerfen.

- In den Aufgabenhinweis schreibt man am besten die randomisierten Variablen (oder die daraus gebildete Gleichung) und die sich daraus ergebenden Lösungen rein. Dies kann man dann verwenden, wenn man in einer Aufgabe ganz oben auf "Frage-Tests und eingesetzte Varianten" klickt. Dort kann man dann unschöne oder falsche Varianten rauslöschen, die man bei der Programmierung nicht vermeiden konnte.
- STACK vereinfacht oft Ausdrücke nicht. Um das zu erzwingen, kann man die Funktion ratsimp(expr) oder fullratsimp(expr) verwenden. Falls ein Klammerausdruck explizit ausgeschrieben werden sollte, kann man expand(expr) verwenden.

Es gibt folgende bereits in Maxima implementierten Funktionen:

- Für Randomisierung gibt es einige Funktionen. rand(n) erzeugt eine beliebige ganze Zahl zwischen 0 und n. rand_with_step(a, b, c) erzeugt eine beliebige ganze Zahl zwischen a und b, mit Schrittweite c. rand_with_prohib($a, b, list$) erzeugt eine beliebige ganze Zahl zwischen a und b, ohne die Zahlen in $list$. Für $list$ eignet sich meistens [0].
- Das Unendlichkeitssymbol ∞ wird mit inf erzeugt, während $-\infty$ mit minf erzeugt wird.
- limit(a_n, n, x_0[, sign]) berechnet den Grenzwert der Folge a_n, welche von der Variablen n abhängt, zum Wert x_0. Die optionale Variable sign kann hierbei + oder − sein, je nachdem, ob man den linksseitigen oder rechtsseitigen Grenzwert berechnen will.

  Beispiel: limit($n^2 + n, n,$ inf) berechnet

  $$\lim_{n\to\infty} n^2 + n.$$

- Die Funktion diff(f, x) leitet die Funktion f nach der Variablen x ab.
- Die Funktion int(f, x[, a, b]) integriert die Funktion f nach der Variablen x von a bis b. Lässt man die Grenzen weg, wird einfach uneigentlich integriert.
- Die Funktion subst($x = x_0, f$) setzt für eine Funktion $f(x)$ den Punkt x_0 ein, berechnet also $f(x_0)$. Der Punkt kann dabei eine Zahl oder eine Variable sein.
- Die Funktion cabs(z) gibt den komplexen Betrag der Zahl z zurück und die Funktion carg(z) gibt das komplexe Argument der Zahl z zurück.
- Die Funktion matrix($list1, \ldots, listm$) erstellt eine $m \times n$-Matrix mit den Zeilen $list1, \ldots, listm$, wobei die Listen alle die Grösse n haben müssen.

  Beispiel: matrix($[1, 2], [3, 4]$) gibt die folgende Matrix zurück:

  $$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

  Das Zugreifen auf die Einträge funktioniert so, wie man es erwarten würde: a[1] gibt die erste Zeile zurück, a[1][1] gibt den ersten Eintrag der ersten Zeile zurück etc. (ja, Arrays starten in Maxima bei 1).

2 Erläuterungen zu bestehenden Aufgaben

Die folgenden Kommentare beziehen sich auf das STACK-Lernelement von 401-0261-G0L Analysis I (D-MAVT, D-MATL) im HS 21. Es kann hier eingesehen werden: `https://moodle-app2.let.ethz.ch/mod/quiz/view.php?id=732397`. Die Aufgabennummern in diesem Dokument entsprechen den Quizseiten.

2.1 STACK 01: Grenzwerte

Hier gibt es nicht viel zu sagen. Eine basic STACK-Aufgabe aus dem Lehrbuch mit randomisierten Aufgabenvariablen.

2.2 STACK 02: Grenzwerte von Funktionen

Diese Aufgabe ist in vier Teilaufgaben eingeteilt, und so eingestellt, dass sie alle auf einer Seite angezeigt werden. Die erste ist relativ standard.
Bei der zweiten sind alle Aufgabenvariablen nach $ta1$ Musterlösungsvariablen. Dort sieht die Musterlösung etwas anders aus, je nachdem, ob wir den linksseitigen oder rechtsseitigen Grenzwert ausrechnen müssen. Falls man den linksseitigen Grenzwert berechnen will, muss man $x \in [-\frac{\pi}{2}, 0)$ betrachten und erhält als Grenzwert -1, falls man den rechtsseitigen Grenzwert berechnen will, muss man $x \in (0, \frac{\pi}{2}]$ betrachten und erhält 1 als Grenzwert. Die Intervallgrenzen habe ich etwas unschön mit den Variablen $int1, int2$ gelöst und das Vorzeichen mit $sign2$.
Bei der dritten Aufgabe kann die Musterlösung vier verschiedene Formen haben:

$$\lim_{x\to aa}(aa-x)\cdot\tan(x) = \lim_{x\to aa}\frac{aa-x}{\cos(x)}\cdot\sin(x) \overset{x=z+aa}{=} \lim_{z\to 0}\frac{-z}{\cos(z+aa)}\cdot\sin(z+aa)$$

$$= \begin{cases} \lim_{z\to 0}\frac{-z}{-\sin(z)}\cdot\cos(z) & \text{if } aa \mod 2\pi = \frac{\pi}{2} \\ \lim_{z\to 0}\frac{-z}{\sin(z)}\cdot -\cos(z) & \text{if } aa \mod 2\pi = \frac{3\pi}{2} \\ \lim_{z\to 0}\frac{-z}{-\cos(z)}\cdot -\sin(z) & \text{if } aa \mod 2\pi = \pi \\ \lim_{z\to 0}\frac{-z}{\cos(z)}\cdot\sin(z) & \text{if } aa \mod 2\pi = 2\pi \end{cases}.$$

Für die ersten zwei Fälle kommt dann raus:

$$= \frac{\lim_{z\to 0}\cos(z)}{\lim_{z\to 0}\frac{\sin(z)}{z}} = \frac{1}{1} = 1.$$

Für die letzten zwei:

$$= \frac{\lim_{z\to 0} -z\cdot\sin(z)}{\lim_{z\to 0}\cos(z)} = \frac{0}{1} = 0.$$

Dies habe ich in der Musterlösung zuerst mit einer Fallunterscheidung (Variable $case$) gelöst, dann die Minuszeichen mit $sign1, sign2$ hinzugefügt und dann auf die obigen letzten Rechnungen vereinfacht. Wie man merkt, ist das alles sehr umständlich, evtl. gibt es eine bessere Art, das zu machen.
Die vierte Aufgabe ist aber wiederum STACK-technisch nicht gross anspruchsvoll und straightforward.

2.3 STACK 03: Injektive Funktionen und Wertebereich

In dieser Aufgabe geht es um die Eingabe von Intervallen. Wir haben es so gelöst, dass die linke und die rechte Grenze beide separat eingegeben werden können. So kann man bei den PRT 1 sehr einfach die Grenzen überprüfen. Im PRT 1 wird in drei Stufen überprüft:

1. Ist der von der Aufgabe vorgegebene Wert val im Intervall enthalten?
2. Ist das eingegebene Intervall ein Subintervall vom Lösungsintervall?
3. Ist das eingegebene Intervall gleich dem Lösungsintervall?

Für den PRT 1 muss noch die ganze radcan-Geschichte beachtet werden: Ich weiss nicht mehr genau, was das gemacht hat, aber es gab ohne diese gewisse Aufgabenversionen, die nicht mehr funktioniert haben.
Insbesondere ist die Antwort für die zweite Teilaufgabe, falls das erste Intervall (a, b) war, das Intervall zwischen $f(a)$ und $f(b)$. Das wird auch im PRT 2 so überprüft. Dabei muss aber jeweils beachtet werden, dass die Intervallgrenzen wechseln können, also $f(a) > f(b)$ gelten kann.
Es gibt auch die Maxima-interne Funktion oo(a, b), welche das Intervall (a, b) erzeugt. Damit kann man den PRT für das initiale Intervall einfach bestimmen, jedoch kann man (meines Wissens nach) nicht aus einem gegebenen Intervall der obigen Form die Intervallgrenzen herauslesen. Somit kann man PRTs für die zweite Teilaufgabe nicht mit dieser Funktion überprüfen, weshalb wir uns für die erste Variante entschieden haben.

Die erste Aufgabe ist von der Lösungsüberprüfung her kompletter Overkill. Das Problem ist nämlich, dass falls $aa \cdot bb < 0$ gilt, dann f drei Maxima statt nur einem Maximum hat. Das kann man aber auch ziemlich einfach hardcoden mit einer Fallunterscheidung.
Falls es dennoch jemanden interessiert, wie meine Lösung funktioniert: Der Grundgedanke ist, einfach Maxima alle Nullstellen von f auflisten zu lassen, das passiert in $zeroseq$ mit Hilfe der Funktion algsys. Das flatten ist dafür da, dass das alles in eine Liste geht, da algsys eine Liste aus Listen zurückgibt, wobei jeder dieser Listen nur ein Element beinhaltet. Dass komplexe Lösungen ignoriert werden, stellt die Funktion realonly sicher. Jedoch gibt algsys Gleichungen der Form $x = x_0$ für irgendeine Nullstelle x_0 zurück, nicht einfach die Nullstellen selber. Daher habe ich eine for-loop geschrieben, die einfach eine neue Liste $zerosun$ erstellt, und die Liste $zeroseq$ durchgeht und einfach die rechte Seite der obigen Gleichungen x_0 einfügt. Schlussendlich wird in $zeros$ noch nach der Grösse der Lösung sortiert.
Jetzt sind wir bereit, das richtige Intervall zu erzeugen. Wir gehen nämlich nun einfach die Nullstellen durch, und schauen, ob das in der Aufgabenstellung gegebene Vielfache von π kleiner als die Nullstelle ist, und erreichen dann unser Intervall, was von links vom Eintrag mit Index $index$ und von rechts vom Eintrag mit Index $index + 1$ begrenzt ist. Dabei müssen wir aber noch beachten, dass falls der Index 0 ist, das Intervall von links mit $-\infty$ begrenzt ist. Dasselbe müssen wir auch tun, falls der Index maximal ist und das Intervall von rechts mit ∞ begrenzt ist.
Der einzige Vorteil meiner Lösung ist, dass sie für jede Funktion mit endlich vielen Nullstellen funktioniert (insbesondere für alle Polynome).
Die zweite und dritte Teilaufgabe sind aber STACK-technisch wieder ganz einfach.
Achtung: In STACK sind $-\inf$ und minf nicht das gleiche! Daher gibt es im PRT 2 der zweiten Teilaufgabe den etwas komischen Workaround mit $ans3corr$.

2.4 STACK 04: Komplexe Zahlen

Diese Aufgabe ist zur Abwechslung mal sehr einfach gehalten.
Die PRTs 5 und 6 berechnen einfach auf Grund den Eingaben $ans1, ans3$ und $ans2, ans4$ die Werte $ans = \frac{ans1}{ans3}$ und $ans = ans2 - ans4$ und überprüft die Eingaben $ans5$ und $ans6$ auf Übereinstimmung mit den beiden ans. Bei PRT 6 muss noch beachtet werden, dass man $\bmod 2\pi$ rechnen muss, da ansonsten Werte $> \pi$ oder $\leq -\pi$ rauskommen können.

2.5 STACK 05: Komplexe Zahlen II

Wieder eine einfache Aufgabe. Die Variablen $eq2r$ und $eq2d$ sind wieder reine Musterlösungsvariablen. Das simp : false ist aber essentiell an der Stelle, weil ansonsten in der Musterlösung die Werte direkt in kartesische Koordinaten umgerechnet werden.
Beide PRTs überprüfen in drei Schritten die Lösungseingabe:

1. Ist die Eingabe eine gültige, nichtleere Menge?
2. Ist die Eingabe eine Teilmenge der Lösungsmenge?
3. Ist die Eingabe gleich der Teilmenge?

2.6 STACK 06: Lineare Ersatzfunktion

STACK-technisch gibt es nicht viel zu sagen. Ich habe aber sehr viele Hilfsvariablen gebraucht, damit die Formeln nicht zu lange und unübersichtlich werden.

2.7 STACK 07: Grössenordnungen

Hier benutzen wir erstmals den Wahr/Falsch-Eingabetyp, welcher genau für solche Aufgaben sehr nützlich ist. Die Aufgabenvariablen $limg2$ und $poly12$ sind Musterlösungsvariablen.

2.8 STACK 08: Ebene Kurven

Hier lernen wir nochmal einen neuen Eingabetyp kennen, nämlich Matrix. In der Eingabe muss nicht eingegeben werden, wie gross sie sein soll, sie übernimmt einfach die Dimensionen von $ta1$. Vektoren setzen wir einfach als $m \times 1$-Matrizen um.
Die PRTs 2 und 3 nehmen einfach den in $ans1$ gegebenen Vektor und benutzen seine Einträge, um daraus die Lösung zu berechnen.
Achtung: Um z.B. auf den zweiten Eintrag im Vektor zuzugreifen, muss man $ans1[2]$**[1]** eingeben, weil Maxima nur den Datentyp Matrix, aber nicht Vektor kennt ($ans1[2]$ ist nicht der Eintrag selbst, sondern eine Liste mit einem Element drin).

2.9 STACK 09: Krümmung und Evolute

STACK-technisch gibt es nichts neues. Die Aufgabe ist von den Lösungen her nicht die schönste, sollte aber zum Ausrechnen noch gehen. *nvechelper* ist eine Musterlösungsvariable.
Von den PRTs ist nur PRT 4 interessant, dieser nimmt einfach alle vorherigen Eingaben $ans1, ans2, ans3$ und berechnet daraus die Lösung.

2.10 STACK 10: Integrieren mit Substitution

Wieder nichts neues.
Bei den PRTs wurde einfach für die Integrationsgrenzen ein PRT gemacht, der beide nacheinander auf Richtigkeit überprüft.

2.11 STACK 11: Bogenlänge

Technisch gesehen sind alle Variablen zwischen $ableitung1$ und *sol* Musterlösungsvariablen. Diese Aufgabe ist ein Import aus einem vorherigen Semester, weshalb sie stilistisch etwas anders aussieht als die vorherigen Aufgaben.

2.12 STACK 12: Volumen und Schwerpunkt

Alle Variablen unter $ta4$ sind Musterlösungsvariablen.
Die PRTs 2 und 3 benutzen $ans1$, um daraus die Lösung zu errechnen, PRT 4 benutzt $ans2$ und $ans3$.