

Logische Programmierung & deduktive Datenbanken — Übungsblatt 12 (Adventure in Prolog) —

Ihre Lösung zu Hausaufgabe a) geben Sie bitte über die Übungs-Plattform in StudIP ab. Einsendeschluss ist der 20. Juli, 24⁰⁰. Wenn Sie die Aufgabe bis zum 14. Juli, 10⁰⁰, abgeben, können wir uns Ihr Programm noch in der Übung anschauen.

Hausaufgaben

- a) In dieser Aufgabe sollen Sie ein ganz kleines Textadventure-Spiel erstellen.

Textadventure-Spiele sind eine Art Bücher, bei denen der Leser/Spieler den Ablauf der Handlung beeinflusst. Es wird ihm/ihr jeweils die aktuelle Situation beschrieben, z.B.:

„Du bist auf einem Waldweg, der hier eine Biegung macht. Im Norden befindet sich eine Lichtung, während im Osten der Weg tiefer in den Wald hineinführt. Du hörst das Summen vieler Bienen.“

Der Spieler kann nun Kommandos/Spielzüge eingeben, wie z.B. „(ich) gehe nach Norden“. Tatsächlich wird dieses Kommando meist durch „n“ abgekürzt (das ist für den Spieler wie den Programmierer bequemer). Natürlich kann der Spieler nur in Richtungen gehen, die der Programmierer/Autor vorgesehen hat: Auf das Kommando „s“ müsste in der obigen Situation geantwortet werden „dort ist der Wald zu dicht“ (oder einfach „das geht nicht“). So ein Spiel ist häufig auf eine recht kleine Zahl von Orten/Räumen beschränkt, an denen sich der Spieler befinden kann (kommerzielle Spiele aus den Achtzigern etwa 100).

Das bloße Herumgehen auf der vom Programmierer entworfenen Karte und das Entdecken neuer Orte würde aber schnell langweilig. Ein anderes wichtiges Kommando ist es, Sachen zu untersuchen oder näher zu betrachten. Dadurch bekommt der Spieler häufig zusätzliche Informationen oder findet versteckte Gegenstände. Im Beispiel könnte er durch „Betrachte Bienen“ ein Astloch mit Honigwaben finden. Schließlich ist es noch wichtig, Gegenstände nehmen zu können, z.B. „Nimm Honig“. Natürlich würden die Bienen das verhindern.

Hier gilt es also eine echte Aufgabe zu lösen, wie man nun doch an den Honig herankommt. Zum Beispiel könnte sich auf der Lichtung eine Hütte befinden, und in dieser vielleicht eine Pfeife, deren Rauch die Bienen abschreckt. Der Spieler muss sich also in einer bestimmten Reihenfolge bestimmte Gegenstände verschaffen und dabei eine

gewisse Phantasie entwickeln. Solche Gegenstände können ihm dann auch neue Wege öffnen. So würde der Honig vielleicht einen Bären besänftigen, der den Eingang zu einer Höhle versperrt. In der Höhle befindet sich z.B. ein Schatz, der das Ziel des Spieles darstellt.

Sie können sich natürlich auch eine andere Geschichte ausdenken. Ihr Spiel sollte mindestens 3 „Räume“ und einen Gegenstand haben, und neben den Kommandos für die vier Himmelsrichtungen auch noch mindestens zwei weitere Kommandos („Verben“) „verstehen“ (z.B. „nimm Gegenstand“, und „betrachte Gegenstand“). Es ist nicht verlangt, dass Ihr Spiel eine sinnvolle Geschichte enthält — es reicht, wenn man in der kleinen Spielwelt herumgehen kann und den Gegenstand nehmen und betrachten kann.

Sie können natürlich die Grammatik für Textadventure-Befehle aus Kapitel 7 ausprobieren und ggf. etwas erweitern oder modifizieren. Sie müssen dazu auch den Scanner am Ende des Kapitels programmieren, um Eingabezeichen bis zum Zeilenende einzulesen und daraus Worte zu machen (Prolog Atome), die dann von der Grammatik verarbeitet werden können. Alternativ können Sie mit der Grammatik auch bis auf die Zeichen-Ebene herunter gehen.

Die Ausgaben für die Kommandos dürfen recht einfach sein, aber vielleicht haben Sie ja auch Spass daran, eine einfache Spielwelt zu implementieren.

Versuchen Sie, Ihr Programm möglichst übersichtlich zu strukturieren.

Zur Wiederholung

- b) Was würden Sie in einer mündlichen Prüfung auf die folgenden fortgeschrittenen Fragen zu magischen Mengen antworten?
- Was ist der Zweck der „Supplementary Predicates“?
 - Wann ist ein Programm „rektifiziert“? Wofür wird diese Bedingung gebraucht?
 - Was ist die Beziehung zwischen magischen Fakten, die aus dem magisch transformierten Programm herleitbar sind, und dem SLD-Baum (für das ursprüngliche Programm)?
 - Inwiefern kann man sagen, dass aus dem magisch transformierten Programm nur solche nicht-magischen Fakten herleitbar sind, die auch bei der SLD-Resolution bewiesen werden?
 - Was ist ein Vorteil der Bottom-Up Auswertung des magisch transformierten Programms gegenüber der SLD-Resolution?
 - Was ist umgekehrt ein Vorteil der SLD-Resolution gegenüber der Bottom-Up Auswertung des magisch transformierten Programms?
 - Nennen Sie einige Probleme der „Magische Mengen“ Methode.
 - Was ist die Grundidee der „SLDMagic“ Methode?

- c) Was würden Sie in einer mündlichen Prüfung auf die folgenden Fragen zur Negation antworten?
- Wie kann man sehen, dass der Operator „not“ (bzw. „\+“) in der logischen Programmierung nicht die klassische Negation ist?
 - Warum macht es Sinn, in dieser Art der Wissensrepräsentation nicht die klassische Negation aus der Logik zu verwenden? Zu welchen Schwierigkeiten würde das führen?
 - Machen Sie ein Beispiel, das zeigt, dass wir es in der logischen Programmierung mit nichtmonotoner Logik zu tun haben.
 - Wie funktioniert SLDNF-Resolution?
 - Was ist das Problem, wenn negierte Literale noch ungebundene Variablen enthalten, wenn sie mit SLDNF-Resolution ausgewertet werden?
 - Was ist ein „kausales Modell“ (engl. „Supported Model“)?
 - Was ist die Grundidee von „Clark’s Completion“?
 - Welches Problem soll die Stratifizierung lösen?
 - Wann ist ein Programm stratifiziert?
 - Was ist das perfekte Modell eines stratifizierten logischen Programms?
 - Gibt es praktische Anwendungen für nicht stratifizierte Programme?
 - Nennen Sie eine Semantik für nicht stratifizierte Programme. Wie wird in dieser Semantik das Problem „ $p \text{ :- not } p.$ “ gelöst?
 - Nennen Sie einige Programmtransformationen auf variablenfreien logischen Programmen, die für sinnvolle Semantiken die Menge der intendierten Modelle nicht ändern sollten (also Äquivalenztransformationen sein sollten).

Zum Selbststudium

g) Schauen Sie sich die folgenden Webseiten an. Sie brauchen für diese Aufgabe nichts abzugeben. Ziel ist, dass Sie einen Eindruck davon gewinnen, was es im Internet zum Thema dieser Vorlesung gibt, und dabei für sich nützliche Quellen entdecken.

- Es sei nochmals auf das Prolog-Tutorial „Adventure in Prolog“ von Dennis Merritt verwiesen:

[<https://www.amzi.com/AdventureInProlog/a1start.php>]

- Das Soufflé System ist eine schnelle Datalog-Implementierung (auch mit paralleler Auswertung):

[<https://souffle-lang.github.io/>]

Zumindest auf Linux- und Mac-Rechnern lässt es sich recht einfach installieren. Es gibt auch eine Online-Version, bei der man nichts installieren muss:

[<http://plang1.it.usyd.edu.au:8000/>]

Folien zur Einführung in die Sprache von Soufflé gibt es hier:

[<https://courses.cs.washington.edu/courses/cse344/18sp/SouffleLanguage.pdf>]

Hier gibt es ein kurzes und recht aktuelles Youtube-Video von einem Vortrag von Bernhard Scholz:

[<https://www.youtube.com/watch?v=Qp3zfM-JSx8>]

- Ich habe mich mit der Negation in der logischen Programmierung beschäftigt, und dazu eine ganze Reihe Veröffentlichungen. In der Vorlesung werden wir z.B. über die Wohlfundierte Semantik (WFS) sprechen. Der folgende Artikel enthält einen Vorschlag zur Berechnung:

Stefan Brass, Jürgen Dix, Burkhardt Freitag, Ulrich Zukowski: Transformation-Based Bottom-Up Computation of the Well-Founded Model. *Theory and Practice of Logic Programming* 1:5, Sept. 2001, pp. 497–538.

[https://www.researchgate.net/publication/1955167_Transformation-Based_Bottom-Up_Computation_of_the_Well-Founded_Model]

Die offizielle Quelle (bei Cambridge University Press) ist:

[<https://doi.org/10.1017/S147106840100103X>]