

Aufgabe

Im Rahmen dieser Aufgabe soll sich näher mit der logischen Programmierung beschäftigt werden. Die erfolgreiche Beschäftigung kann beispielsweise mit entsprechenden Screenshots dokumentiert werden.

Lösungsansatz und -ziel

Zur Lösung der Aufgabenstellung beschäftige ich mich näher mit der bekanntesten logischen Programmiersprache PROLOG. Hierbei steht deren Laufzeitumgebung sowie die Umsetzung des im Kurs vorgegebenen Stammbaums als Wissensbasis im Vordergrund. Anschließend soll die Wissensbasis abgefragt werden.

Somit muss eine entsprechende Basis, die als Schwerpunkt den oben genannten Stammbaum beinhaltet, erstellt werden. Mindestens Fragen zu den Eltern, Kindern und Geschwistern sollen grundsätzlich möglich sein.

Dies bildet natürlich nur einen sehr kleinen Ausschnitt der Funktionalität ab, welche mit Prolog als Sprache umsetzbar ist, steht aber beispielhaft für einen ersten Einblick in deren Möglichkeiten.

Als weiteren Schritt möchte ich die zuvor erzeugte Wissensrepräsentation in einer C# Konsolenanwendung exemplarisch verfügbar machen.

Prolog



Prolog¹ steht für *programmation en logique* (Programmieren mit Logik) und wurde maßgeblich von Alain Colmerauer entwickelt. Im Gegensatz zu objektorientierten oder funktionalen Sprachen erfolgt hier die Erstellung von Programmen durch die Definition von logischen Regeln als Wissensbasis.

Aus dieser Wissensrepräsentation werden dann Antworten zu Fragestellungen abgeleitet. Die Ausführung erfolgt durch einen Interpreter. Prolog Programme werden somit nicht kompiliert.

Die Hauptaufgabe bei der Programmierung ist sicherlich das Definieren und Festlegen der Datenbasis und Regeln als Wissensdatenbank. Hierbei wird deklarativ vorgegangen. Bei der Annahme, dass die Verarbeitung der Fakten und Regeln durch den Interpreter korrekt erfolgt, liegen Fehlerquellen somit primär in der Erstellung der Wissensdatenbank selbst.

¹ Interessanterweise lässt sich PROLOG sowohl aus dem französischen **programmation en logique**, dem deutschen **Programmieren mit Logik** und dem englischen **Programming in logic** herleiten.

Mit der Zeit ist es zur Bildung verschiedener Sprachen und Dialekte gekommen. Zudem existieren Bibliotheken, welche die Verwendung logischer Sprachen in Sprachen wie dem objektorientierten C# ermöglichen. Im folgenden wird sich jedoch primär auf Prolog bezogen.

In einem Prolog Programm sind die kleinsten Bestandteile Terme und Klauseln. Zu den Termen gehören Variablen, Konstanten und sogenannte komplexe Terme. Diese bestehen aus einem Funktor sowie Argumenten wie beispielsweise *maennlich(denethor)*. Hierbei ist *maennlich* der Funktor, *denethor* das Argument. Zudem handelt es sich bei *maennlich* um ein Prädikat.

Klauseln umfassen Fakten, Regeln und Fragen. Ein Fakt im Sinne von Prolog ist die Angabe eines Zustandes für ein bestimmtes Objekt. Im vorhergehenden Beispiel ist Fakt, dass Denethor männlich ist. Ein Beispiel für eine Regel ist *sohn(X,Y) :- maennlich(X), kind(Y,X).*. Die Bedeutung ist: X ist Sohn von Y, wenn *maennlich(X)* und *kind(Y,X)* gilt.

Weiterhin handelt es sich bei dem linken Teil der Regel um den Regelkopf, bei dem rechten Teil um den Regelrumpf. Zudem werden Regel und Namen, welche sich auf die Reihenfolgen der Variablen auswirken, vom jeweiligen Programmierer festgelegt und sollte somit im Code dokumentiert werden.

Die so entstandene Wissensbasis muss von dem Prolog Interpreter *konsolidiert* werden und steht danach für Fragen bereit.

Abschließend soll noch auf eine Besonderheit bei Prolog hingewiesen werden. So verwendet Prolog standardmäßig für Variablen Großbuchstaben und für Konstanten Kleinbuchstaben und ist hierbei sehr penibel. Jede Eingabe muss ebenso wie eine Frage mit einem Punkt abgeschlossen werden. Zudem werden logische Variablen erst zur Laufzeit an bestimmte Objekte gebunden und unterscheiden sich somit sehr zur imperativen oder objektorientierten Verwendung.

Bei der Ausgabe erfolgt in der Regel immer nur jeweils eine Ausgabe. Die nächste Ausgabe muss mit der Eingabe eines Punktes und Enter oder betätigen der Spacetaste angefordert werden. Existieren keine weiteren Ergebnisse, so wird false ausgegeben.

Installation der Laufzeitumgebung

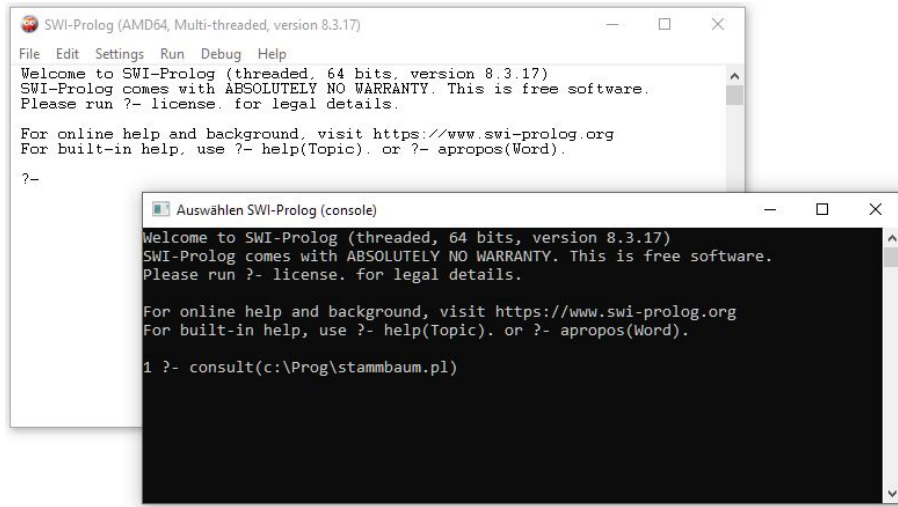
Zur Umsetzung soll SWI Prolog² Verwendung finden. Im Menü unter *Download/SWL-Prolog* erfolgt eine Verzweigung hin zu einer Downloadseite. Hier kann die für ein Betriebssystem entsprechende Version heruntergeladen und installiert werden. Hierbei existieren Versionen für Windows, Linux sowie Mac. Weiterhin existiert eine Reihe an Erweiterungen³.

Nach der Installation ist eine Laufzeitumgebung zur Ausführung von Prolog Programmen verfügbar und die Erweiterung *“.pl”* wurde für Prologdateien registriert. Zudem verfügt das

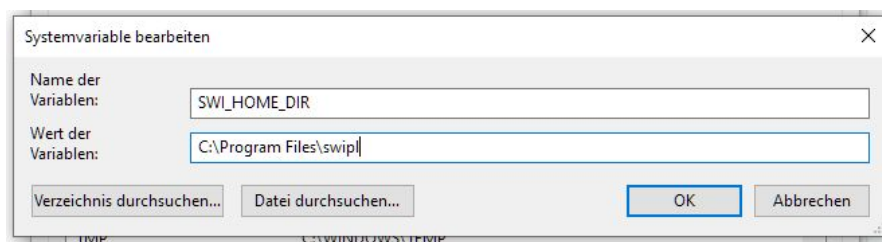
² <https://www.swi-prolog.org/>

³ <https://www.swi-prolog.org/pack/list>

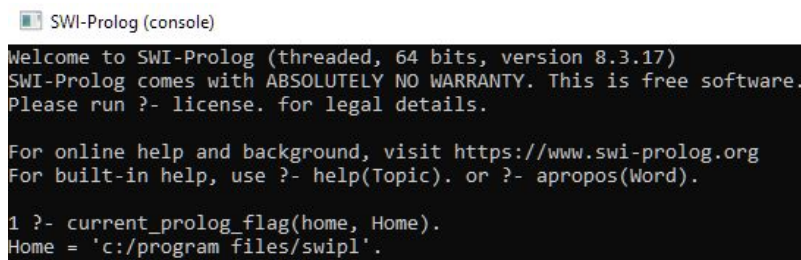
Zielsystem nun über einen sehr einfachen Editor sowie einer Prolog-Console zur Abfrage der Datenbasis:



Als weiterer Schritt empfiehlt sich, die Systemvariablen um die Variable *SWI_HOME_DIR* zu erweitern, wie in der folgenden Abbildung zu sehen ist.



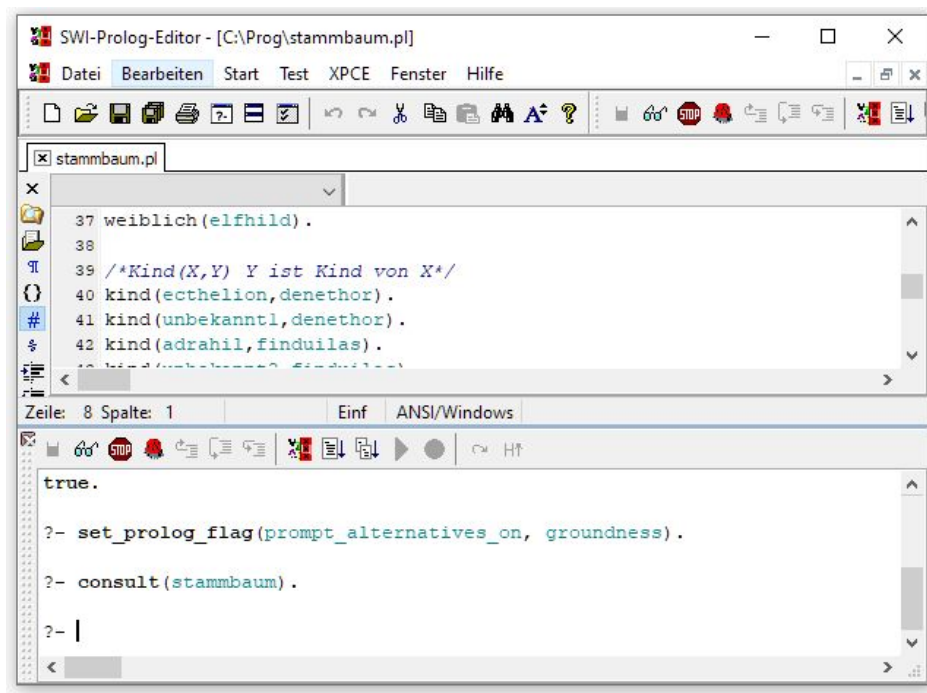
Sollte das aktuelle Installationsverzeichnis unbekannt sein, so kann dieses mit Hilfe der Prolog Konsole und dem Befehl "*current_prolog_flag(home, home).*" abgefragt werden. Hierbei den abschließenden Punkt nicht vergessen:



Um mehr Komfort zu haben, bietet sich die Installation einer besser ausgestatteten IDE an, von denen es jedoch leider eher wenige gibt. Eine Auswahl findet sich auf der Seite von

SWI-Prolog⁴. Für diese Arbeit kommt die externe und freie IDE *SWI-Prolog-Editor*⁵ zum Einsatz.

Hierbei muss der Begriff IDE jedoch etwas relativiert werden. In erster Linie leistet der Editor eine angenehme Handhabung des SWI Interpreters sowie der SWI Konsole. Beides ist quasi direkt in der Anwendung implementiert und ermöglicht so ein angenehmes arbeiten. Hilfreich ist auch die grundsätzliche Codeunterstützung. Bei der Installation wird für die Endung *“.pl“* die Anwendung registriert.



Erzeugung der Datenbasis und erste Abfragen

Auf Grundlage eines zur Verfügung gestellten Stammbaums soll eine Datenbasis definiert werden, welche die Eingangs genannten Fragestellungen beantworten kann. Die entstandene Datei *stammbaum.pl* ist auf GitHub⁶ verfügbar.

In einem ersten Schritte werden die atomaren Fakten definiert. Hierzu gehören als erstes *maenlich* und *weiblich*. Die hier getroffenen Aussagen sind immer wahr. Im zweiten Schritt können auf Basis dieser Fakten nun die weitere Regeln *mutter*, *vater*, *sohn* und *tochter* abgeleitet werden. Weitere ableitbare Fakten wären *eltern*, *großeltern* oder *enkel*. Da diese vom Prinzip her gleichartig wie die vorhergehenden Regeln sind, wird darauf verzichtet.

Im folgenden wird das Ergebnis und Verhalten der Abfrage nach Tochter angezeigt. Tochter ist definiert als:

⁴ <https://www.swi-prolog.org/IDE.html>

⁵ <https://arbeitsplattform.bildung.hessen.de/fach/informatik/swiprolog/swiprolog.html>

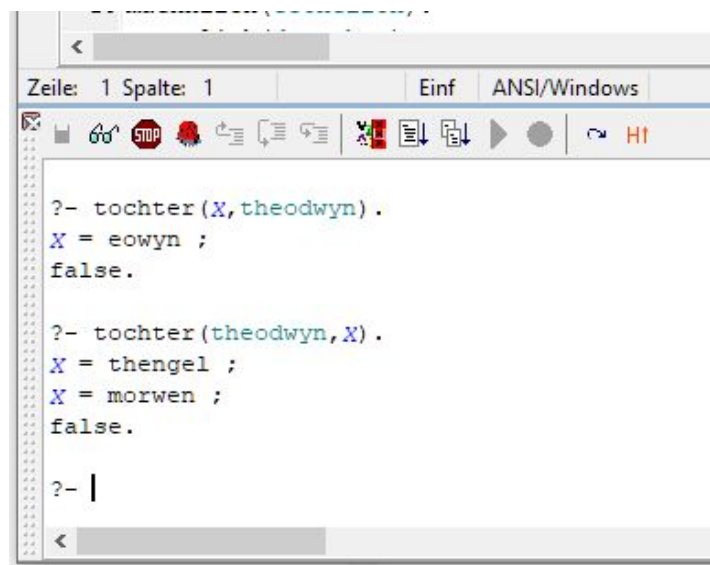
⁶ <https://github.com/ChristianKitte/PrologStammbaum>

```
10 /*X ist Tochter von Y*/  
11 tochter(X,Y) :- weiblich(X), kind(Y,X).
```

zudem gilt:

```
34 weiblich(theodwyn).
```

In der ersten Abfrage⁷, wird gefragt: Für welche X ist Theodwyn Mutter (man beachte die oben stehenden Definitionen !). Dies führt korrekterweise zu der Tochter Eowyn. In der zweiten Frage wird gefragt: Für wen ist Theodwyn Tochter, was logischerweise zu Ihren Eltern führt:



```
Zeile: 1 Spalte: 1 Einf ANSI/Windows  
?- tochter(X,theodwyn).  
X = eowyn ;  
false.  
  
?- tochter(theodwyn,X).  
X = thengel ;  
X = morwen ;  
false.  
  
?- |
```

Anhand dieser Beispiele ist gut zu sehen, wie flexible Prolog bei den Abfragen ist und welche Rolle hierbei die Angabe von Variablen hat. Aufgabe ist nicht die Übergabe eines Wertes, sondern vielmehr das Aufnehmen von Werten. Sie stehen somit in der Bedeutung von einem unbekannten und zurück zu liefernden Wert.

Ausführung innerhalb von C#

Um die zuvor erzeugte Wissensbasis in C# verfügbar zu machen, bediene ich mich der Bibliothek *CSProlog*⁸, welche als NuGet Package⁹ einfach zu installieren ist. Deren gesamter Sourcecode ist auf GitHub¹⁰ verfügbar.

⁷ Abfragen wurden innerhalb des SWI-Prolog-Editor ausgeführt

⁸ <https://github.com/jsakamoto/CSharpProlog/>

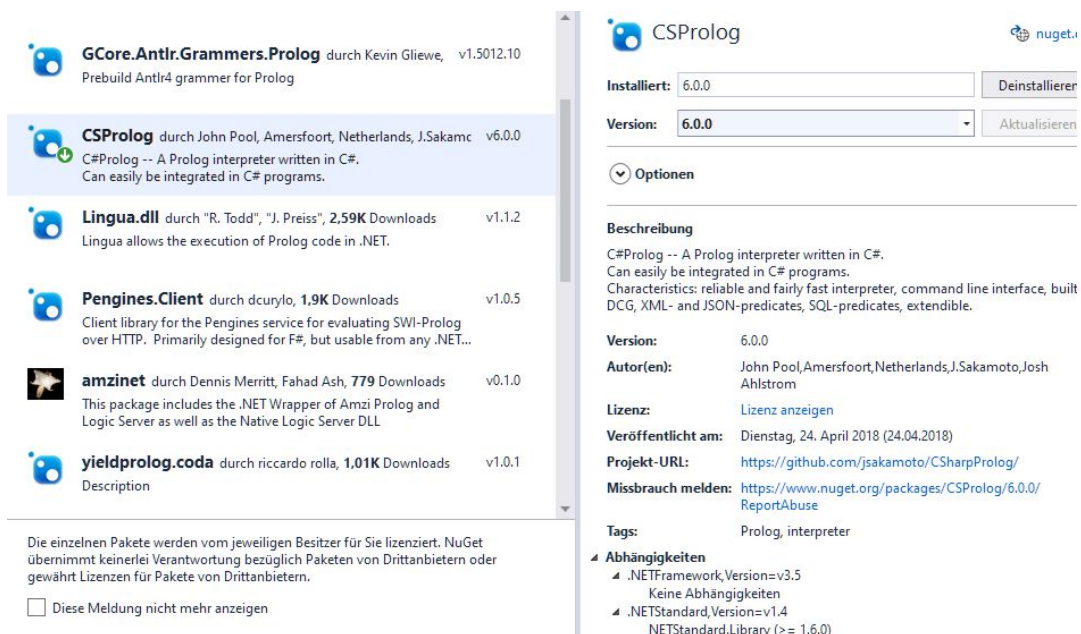
⁹ <https://www.nuget.org/packages/CSProlog/#>

¹⁰ <https://github.com/ChristianKitte/PrologStammbaum>

Neben einer einfachen Syntax überzeugt die Anzahl der Überarbeitungen (die letzte im April 2018) und die Verfügbarkeit auf GitHub. Als .net Komponente sollte überdies der Einsatz auch in F# unproblematisch sein.

Darüber hinaus gibt es weitere Bibliotheken, welche man verwenden kann. Zudem steht die Möglichkeit offen, SWI-Prolog selbst aus C# heraus anzusprechen.

In der folgenden Abbildung sieht man die wichtigsten Daten des Paketes im Fenster des Paketmanagers von Visual Studio.



Als Basis meiner Anwendung dient eine einfache Konsolenanwendung, basierend auf dem .NET Framework 4.7.2.

Nach dem Hinzufügen des Paketes kann über den auf GitHub verfügbaren Beispielcode sofort der Erfolg der Installation getestet werden. Die folgende Abbildung zeigt einen Ausschnitt des hierzu verwendeten Beispielcodes innerhalb der Main Methode:

```
var prolog = new PrologEngine(persistentCommandHistory: false);

// 'socrates' is human.
prolog.ConsultFromString(prologCode: "human(socrates).");
// human is bound to die.
prolog.ConsultFromString(prologCode: "mortal(X) :- human(X).");

// Question: Shall 'socrates' die?
var solution = prolog.GetFirstSolution(query: "mortal(socrates).");
Console.WriteLine(solution.Solved); // = "True" (Yes!)
```

Sehr schön ist zu erkennen, dass in dem Beispiel keine Datei als Wissenspeicher Verwendung findet, sondern einzelne Prädikate direkt im Programm angelegt werden.

Als ein Defizit der Bibliothek ist die leider nur sehr spärliche Dokumentation zu nennen. Zwar existieren auf GitHub eine Reihe von Beispielen und auch der Sourcecode, jedoch keine umfassende Referenz oder gar ein Tutorial. So bleibt nichts weiter übrig, als sich die notwendige Syntax selbst zu erschließen.

Für meine Umsetzung bin ich auf diesem Wege zu dem folgenden Code gekommen, der für die hier vorgesehenen Zwecke ausreichend erscheint.

```
// Prolog initialisieren
var prolog = new PrologEngine(persistentCommandHistory: false);
var wissensdatenbank :string = Environment.CurrentDirectory + @"\stammbaum.pl";

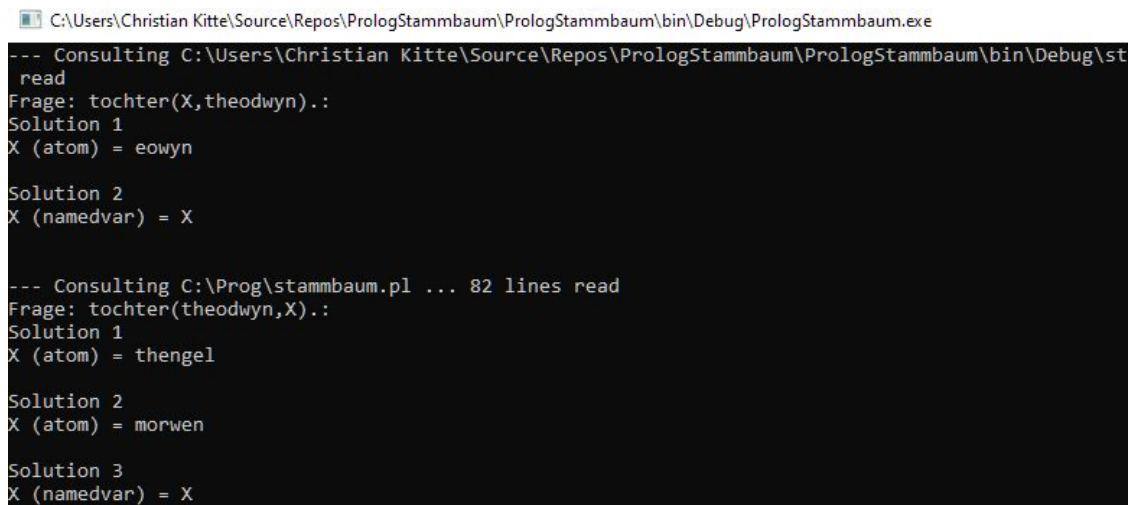
// Frage stellen und Antwort ausgeben
SolutionSet result = prolog.GetAllSolutions( sourceFileName:wissensdatenbank, query: "tochter(X,theodwyn).");
Console.WriteLine("Frage: tochter(X,theodwyn).: ");
Console.WriteLine(result.ToString());

result = prolog.GetAllSolutions( sourceFileName:@"C:\Prog\stammbaum.pl", query: "tochter(theodwyn,X).");
Console.WriteLine("Frage: tochter(theodwyn,X).: ");
Console.WriteLine(result.ToString());

// Konsole offen halten
Console.ReadLine();
```

Im Gegensatz zur obigen Lösung verwende ich eine andere Methode, welche den Name der Datei, welche das Wissen beinhaltet, als Argument verlangt. Letztlich die zu Anfang erstellte Prolog Datei *Stammbaum.pl*.

Bei der Ausführung ergibt sich mit diesem Code für die bereits zuvor gestellten Fragen die folgende Ausgabe:



```
C:\Users\Christian Kitte\Source\Repos\PrologStammbaum\PrologStammbaum\bin\Debug\PrologStammbaum.exe
--- Consulting C:\Users\Christian Kitte\Source\Repos\PrologStammbaum\PrologStammbaum\bin\Debug\st
read
Frage: tochter(X,theodwyn).:
Solution 1
X (atom) = eowyn

Solution 2
X (namedvar) = X

--- Consulting C:\Prog\stammbaum.pl ... 82 lines read
Frage: tochter(theodwyn,X).:
Solution 1
X (atom) = thengel

Solution 2
X (atom) = morwen

Solution 3
X (namedvar) = X
```

Wie zu sehen ist, wird die Gleiche Antwort wie zuvor zurück geliefert. Der nicht zu unterschätzende Vorteil ist jedoch, dass die Antwort diesmal durch eine C# Programm erstellt wurden und Frage sowie Antwort innerhalb des Programms verfügbar sind.

Mit der Möglichkeit, Fakten auch komplett innerhalb eines Programms zu erstellen, statt wie hier aus einer Datei einlesen zu lassen, ergeben sich interessante Möglichkeiten, die Mächtigkeit von Prolog in eigenen Anwendungen zu verwenden.

Fazit

Ich hatte mich bereits früher schon einmal mit der Sprache beschäftigt, bin aber nicht sehr tief eingestiegen, woran auch mein damaliger Wissensstand schuld war. Ein Bachelorstudium und Teile vom Master später fällt der (Wieder-) Einstieg deutlich leichter.

Für mich scheint der anfänglich zunächst eher abstrakte Einstieg, abgesehen vom Hype um die Objektorientierung, mit ein Grund zu sein, dass logische Sprachen nicht so verbreitet und/oder bekannt sind.

Zudem habe ich es bei der Beschäftigung im Rahmen dieser Arbeit geschafft, Prolog tatsächlich innerhalb von C# zu nutzen. Hierdurch eröffnen sich wie zuvor schon bei der Beschäftigung mit F# viele interessante Perspektiven.

Weiterführende Lektüre

Im Rahmen dieser Arbeit sind mir besonder drei Seiten aufgefallen, welche einen guten und fundierten Einstieg ermöglichen. Dies ist zunächst eine 141 Seiten starke Einführung in Prolog¹¹ von Sven Naumann aus dem Jahr 2007.

Von der TU Darmstadt sind die Vorlesungsfolien der Veranstaltung Einführung in das Programmieren : PROLOG - SS06¹² verfügbar, welche ebenfalls sehr gut aufgebaut sind. Hierzu gehören auch Übungsblätter.

Eine für meinen Geschmack sehr schöne und leicht verständliche Einführung nicht nur in Prolog bietet die Website¹³ von Susanne Hackmann von der Uni Bremen. Die Einführung in Prolog ist ebenfalls als PDF¹⁴ verfügbar.

¹¹ <https://www.uni-trier.de/fileadmin/fb2/LDV/Naumann/prolog.pdf>

¹² <https://www.ke.tu-darmstadt.de/lehre/archiv/ss06/prolog>

¹³ <http://www.fb10.uni-bremen.de/homepages/hackmack/prolog/sitemap.vbhtml>

¹⁴ <http://www.fb10.uni-bremen.de/homepages/hackmack/prolog/skript>