Aufgabe: Wem gehört das Zebra?

Schreiben Sie ein SetlX-Programm, welches das folgende Rätsel löst.

1. Es gibt 5 Häuser. Jedes der Häuser hat eine andere Farbe.
2. In jedem Haus wohnt ein Bewohner einer anderen Nationalität.
3. Jeder Hausbewohner bevorzugt ein anderes Getränk, raucht eine andere Marke Zigaretten und hält genau ein Haustier.
4. Keine der Personen trinkt das Gleiche, raucht das Gleiche oder hält die gleiche Art Haustier.

Außerdem wissen wir folgendes:

1. Der Brite wohnt im roten Haus.
2. Der Schwede hält einen Hund.
3. Der Amerikaner trinkt Whisky.
4. Das grüne Haus steht links vom weißen Haus.
5. Der Besitzer vom grünen Haus trinkt Kaffee.
6. Die Person, die Camel raucht, hält einen Papagei.
7. Der Mann, der im mittleren Haus wohnt, trinkt Milch.
8. Der Besitzer vom gelben Haus raucht Dunhill.
9. Der Norweger wohnt im ersten Haus.
10. Der Marlboro-Raucher wohnt neben dem, der eine Katze hält.
11. Der Mann, der ein Schwein hält, wohnt neben dem, der Dunhill raucht.
12. Der Winfieldraucher trinkt gerne Bier.
13. Der Norweger wohnt neben dem blauen Haus.
14. Der Deutsche raucht Rothmanns.
15. Der Marlboro-Raucher hat einen Nachbarn, der Wasser trinkt.

Die Frage lautet nun: Wem gehört das Zebra?

Verwenden Sie zur Lösung des Problems die Vorlage, die Sie im Netz unter der Adresse

`github.com/karlstroetmann/Logik/blob/master/Aufgaben/Blatt-10/zebra-frame.stlx`

finden. Dieser Rahmen enthält bereits eine Implementierung der Funktion

`parseKNF`(s),

die eine Formel, die als String s gegeben ist, in eine Menge von Klauseln umwandelt. Weiter enthält der Rahmen die Implementierung der Funktion

`davisputnam`(*clauses*, *literals*),

die für eine gegebene Menge von Klauseln eine Lösung berechnet. Wenn Sie diese Funktion verwenden, sollten Sie für den Parameter *literals* die leere Menge einsetzen.

Bei der Lösung dieser Aufgabe sollten Sie die folgenden Terme als aussagenlogische Variablen verwenden:

1. `Briton`(i) mit $i \in \{1, \cdots, 5\}$. Diese Variable drückt aus, dass der Brite im Haus mit der Nummer i wohnt. Die anderen Nationalitäten werden mit Hilfe der Variablen `German`(i), `Swede`(i), `American`(i) und `Norwegian`(i) kodiert.

2. `Red`(i) drückt aus, dass das Haus mit der Nummer i rot ist. Für die anderen Farben benutzen Sie die Variablen `Green`(i), `White`(i), `Blue`(i) und `Yellow`(i).

3. `Dunhill`(i) drückt aus, dasss der Bewohner des i-ten Hauses Dunhill raucht. Für die anderen Zigaretten-Marken benutzen Sie die Variablen `Camel`(i), `Marlboro`(i), `Winfield`(i) und `Rothmanns`(i).

4. `Dog`(i) drückt aus, dass der Bewohner des i-ten Hauses einen Hund hat. Für die anderen Haustiere benutzen Sie die Variablen `Parrot`(i), `Cat`(i), `Pig`(i) und `Zebra`(i).

5. `Whiskey`(i) sagt aus, dass der Bewohner des i-ten Hauses Whiskey trinkt. Für die anderen Getränke benutzen Sie die Variablen `Coffee`(i), `Beer`(i), `Milk`(i) und `Water`(i).

Zur Lösung des Problems sollten Sie die folgenden Teilaufgaben bearbeiten:

1. Implementieren Sie eine Funktion `onePerHouse` die beispielsweise in der Form

   ```
   onePerHouse("German", "Briton", "Swede", "American", "Norwegian")
   ```

   aufgerufen wird und die ausdrückt, dass es ein Haus gibt, in dem der Deutsche wohnt, ein Haus, in dem der Britte wohnt, etc. Außerdem soll diese Formel noch ausdrücken, dass in jedem Haus nur eine Person wohnt. Wenn also beispielsweise der Deutsche im Haus Nummer 4 wohnt, dann kann dort sonst niemand mehr wohnen.

   Es ist sinnvoll, wenn Sie sich geeignete Hilfsprozeduren definieren, mit denen Sie diese Prozedur implementieren können. In dem vorgegebenen Rahmen ist bereits die Prozedur `somewhere` vordefiniert, die ausdrückt, dass jeder irgendwo wohnen muss. Sie sollten noch die Prozeduren `someone` und `atMostOneAt` definieren. Dir Prozedur `someone` drückt aus, dass in jedem Haus jemand wohnt, während `atMostOneAt` ausdrückt, dass in jedem Haus höchstens einer wohnt.

   Beachten Sie hier, dass die Prozeduren `atMostOne` und `propVar` in dem vorgegebenen Rahmen bereits definiert sind.

2. Die Prozedur `sameHouse`, mit der Sie ausdrücken können, dass beispielsweise der Brite in dem roten Haus wohnt, ist bereits vorgegeben und zeigt, wie Sie mit Hilfe von String-Interpolation Formeln berechnen können.

3. `nextTo` kann beispielsweise in der Form `nextTo("Marlboro", "Cat")` aufgerufen werden um auszudrücken, dass derjenige, der Marlboro raucht, neben dem Haus mit der Katze wohnt. Die Prozedur `nextTo` muss von Ihnen implementiert werden.

4. `allClauses` berechnet eine Menge von Klauseln, die zusammen genau der Problem-Beschreibung entsprechen. Wenn Sie die Prozedur korrekt implementieren, wird das Problem durch den Aufruf der Funktion `solve` gelöst. Die Berechnung der Lösung dauert auf meinem Rechner weniger als drei Sekunden.