Wissensbasierte Systeme

Arbeitsheft zur Lehrerfortbildung

Konzeption des Arbeitsheftes:

Das Arbeitsheft wird in den Lehrerfortbildungen „KI@Informatik13 – was, wozu, wie, womit unterrichten“ der Didaktik der Informatik der Universität Passau zum Thema „Künstliche Intelligenz“ in der 13. Jahrgangsstufe verwendet.

Dr. Wolfgang Pfeffer

Dominicus-von-Linprun-Gymnasium Viechtach
E-Mail: schule@pfeffer-wolfgang.de

Tobias Fuchs

Universität Passau
E-Mail: fuchs_unipa@outlook.de

Das Material zu den Arbeitsaufträgen findet sich im Mebis-Kurs „Material zu den KI-Fortbildungen der Universität Passau“ mit der `ID`-Nummer `1324361`. Der Einschreibeschlüssel ist `KI_FB_UP`.

Herzlichen Dank an Ute Heuer von der Didaktik der Informatik der Universität Passau für viele wertvolle Diskussionen und Anregungen für die Umsetzung des Arbeitsheftes.

Die enthaltenen Bilder sind (falls nicht anders angegeben) mit einer KI erstellt oder der Plattform `www.pixabay.com` entnommen und wurden teilweise weiter bearbeitet. Die Bilder von Pixabay können unter der dort aufgeführten Inhaltslizenz kostenlos genutzt werden, wobei kein Bildnachweis nötig ist.

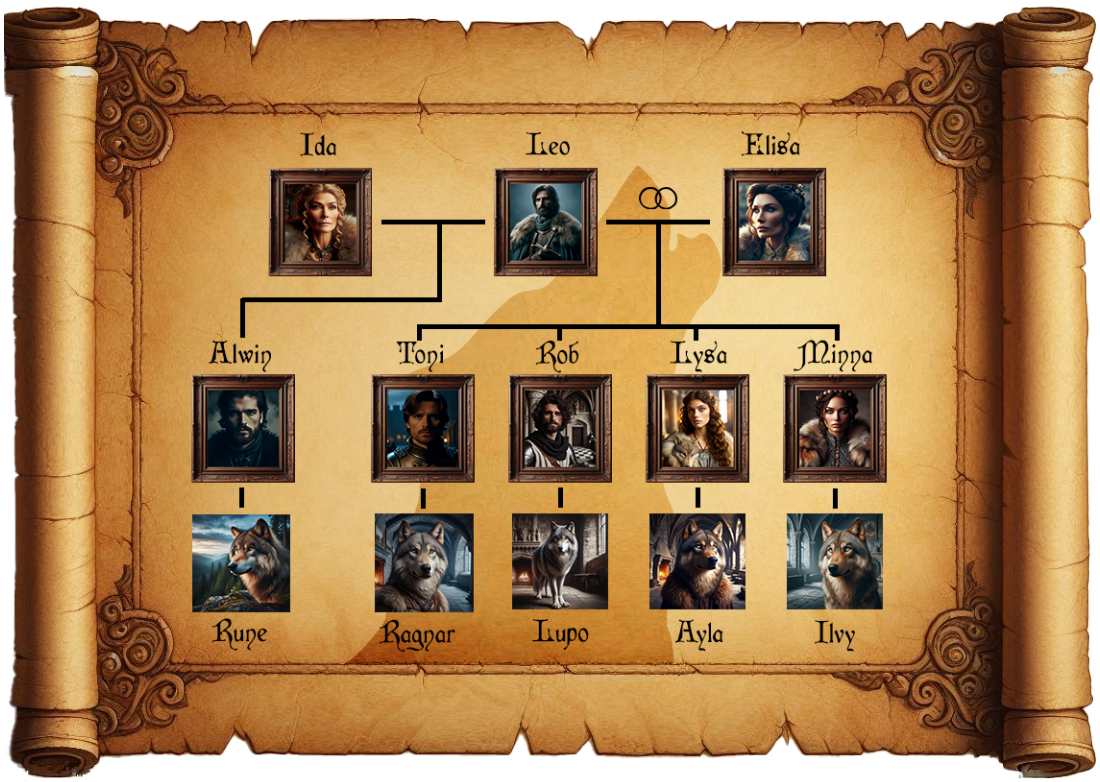
Das Arbeitsheft steht unter einer CC-BY-NC-SA-Lizenz.

1. Das Setting: Die Legenden um Burg Wolfsfels und Fuchsspitze

Die Einführung in PROLOG erfolgt schrittweise anhand eines fiktiven Szenarios rund um die Königshäuser von Burg Wolfsfels, Fuchsspitze und Reitersberg. Hierbei kommt der Stammbaum von Burg Wolfsfels, der Hofstaat von Burg Fuchsspitze, Landkarten über die Hoheitsgebiete der Könighäuser, die Türme von Burg Reitersberg sowie logische Rätsel wie etwa zu Ritterturnieren und Kräuterkunde zum Einsatz.

2. Fakten und einfache Anfragen

Für die Arbeitsaufträge in diesem Abschnitt betrachten wir nachfolgenden Ausschnitt aus dem Stammbaum von Burg Wolfsfels:



Beispiel (Fakten in PROLOG)

Beispielsweise lassen sich bestimmte Eigenschaften wie etwa Geschlecht oder Tierart des Stammbaumes wie folgt als **Fakten** in PROLGO darstellen:

```
maennlich(leo).
maennlich(alwin).
weiblich(ida).
wolf(rune).
```

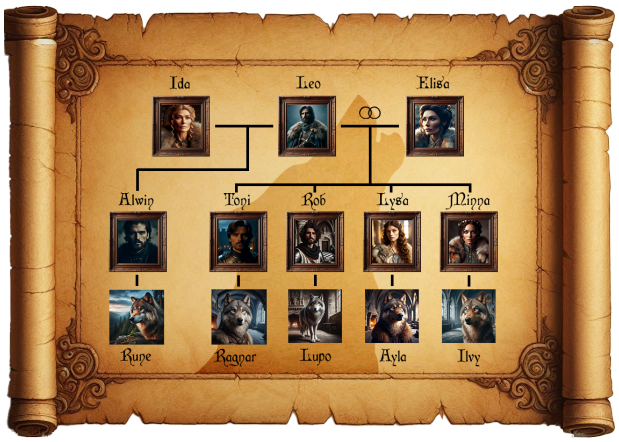
Fakten in PROLOG bestehen somit aus einem **Prädikatsnamen** wie z. B. `maennlich` gefolgt von endlich vielen Konstanten in Klammern wie z. B. `leo`.

Merke (Fakten (propädeutisch))

Ein **Fakt** ist eine wahre Aussage über Objekte bzw. über Beziehungen zwischen Objekten. Fakten sind somit die Grundbausteine der Wissensrepräsentation in PROLOG.

Arbeitsauftrag 1: Fakten in Prolog – Teil 1

Öffnen Sie das PROLOG-Programm `wolfsfels_einstieg.pl` im Online-Editor SWISH (https://swish.swi-prolog.org/) und ergänzen Sie dieses so, dass alle Fakten über Geschlecht bzw. Tiere des Stammbaums berücksichtigt sind:



Merke (Prädikat (noch eingeschränkt auf Fakten))

Fakten mit demselben Prädikatsnamen und derselben Stelligkeit (Anzahl an Konstanten in den Klammern), werden unter dem Begriff **Prädikat** zusammengefasst.

Die Wissensbasis aus obigem Arbeitsauftrag enthält somit die Prädikate `maennlich/1`, `weiblich/1` und `wolf/1`.

```
/* Praedikat maennlich/1 */
maennlich(leo).
maennlich(alwin).
maennlich(toni).
maennlich(rob).
```

Beispiel (Einfache Anfragen in PROLOG (noch ohne Variablen))

Um Informationen aus der Wissensbasis abzurufen, müssen **Anfragen** formuliert werden. Diese beginnen immer mit den Zeichen `?-` und enden mit einem `.` Die Anfrage

```
?- maennlich(leo).
```

an unsere bisher erstellte Wissensbasis liefert die Antwort `true`, da der Fakt `maennlich(leo)` in der Wissensbasis vorhanden ist. Die Anfrage

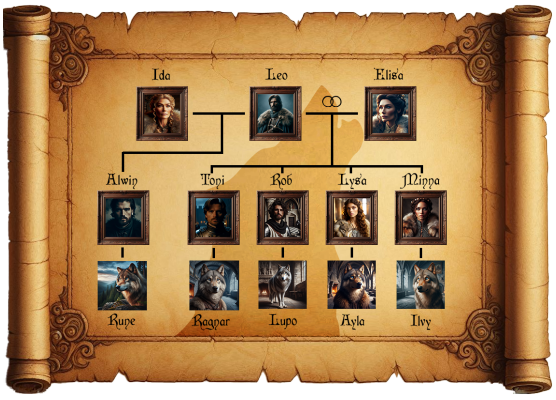
```
?- wolf(leo).
```

liefert die Antwort `false`. PROLOG überprüft hierbei, ob die Kombination aus Prädikatsbezeichner und Konstanten aus der Abfrage einem Fakt der Wissensbasis entspricht. Falls ja, wird `true` ausgegeben, sonst `false`.

Beispiel (Beziehungen als Fakten darstellen)

Neben den „einfachen“ Fakten zu Geschlecht und Tierart kann man auch Beziehungen zwischen Objekten als Fakten modellieren, wie etwa die **Eltern-Kind-Beziehung** oder die **Person-Wolf-Zuordnung**. Dies kann etwa wie folgt in PROLOG dargestellt werden:

```
elternteil(leo,toni).
elternteil(elisa,rob).
begleiter(toni,ragnar).
```



Arbeitsauftrag 2: Beziehungen als Fakten darstellen

Ergänzen Sie Ihr Programm um alle im Stammbaum angegebenen Beziehungen (Eltern + Begleiter) und überprüfen Sie die neue Wissensbasis mit geeigneten Anfragen:

3. Variablen und logische Verknüpfungen

Merke (Variablen)

In `PROLOG` werden **Variablen** nicht deklariert. Das einfache Benennen einer Variable ist ausreichend, wobei `PROLOG` eine Zeichenkette als Variable betrachtet, wenn diese mit einem Großbuchstaben oder einem Unterstrich beginnt. Weiter hat eine Variable zunächst keinen Wert; sie erhält diesen durch das Prologsystem während der Abarbeitung einer Anfrage. Variablen werden hierbei an Konstanten gebunden und es wird überprüft, ob eine Übereinstimmung mit der Wissensbasis gefunden werden kann. Sobald eine Variable an einen Wert gebunden ist, ist die Variable im Gegensatz zu vielen anderen Programmiersprachen nicht mehr veränderbar.

Beispiel (Variablen)

Die Anfrage

```
?- weiblich(X).
```

liefert die Antwort

```
X = elisa
X = ida
X = lysa
X = minna
```

PROLOG sucht hierbei in der Wissensbasis nach Variablenbelegungen für `X`, so dass der Prädikatsname `weiblich` mit dieser Belegung als Fakt in der Wissensbasis vorkommt.

Arbeitsauftrag 3: Variablen in Anfragen

Öffnen Sie die Wissensbasis zum Stammbaum von Burg Wolfsfels nach Arbeitsauftrag 2 in SWISH.

(a) Erläutern Sie, welche Bedeutung die Anfrage `?- elternteil(leo,X).` hat und geben Sie alle Belegungen von `X` an, die das PROLOG-System liefert. Überprüfen Sie Ihre Lösung, indem Sie die Anfrage in SWISH ausführen.

(b) Erläutern Sie, welche Bedeutung die Anfrage `?- elternteil(X,lysa).` hat und geben Sie alle Belegungen von `X` an, die das PROLOG-System liefert. Überprüfen Sie Ihre Lösung, indem Sie die Anfrage in SWISH ausführen.

Merke (Logische Verknüpfungen in PROLOG)

(a) **UND-Verknüpfung**: Wird verwendet, um mehrere Bindungen zu kombinieren, die erfüllt sein müssen. Die Anfrage

```
?- elternteil(leo,X),maennlich(X).
```

gibt alle Kinder von Leo aus, die männlich sind, d.h. es werden die Söhne von Leo ausgegeben.

(b) **ODER-Verknüpfung:** Wird verwendet, um alternative Bedingungen anzugeben, von denen mindestens eine erfüllt sein muss. Die Anfrage

```
?- elternteil(leo,X); elternteil(elisa,X).
```

gibt alle Kinder aus, die Leo oder Elisa (oder beide) als Elternteil haben.

(c) **NICHT-Verknüpfung:** Drückt Negation aus, also dass eine bestimmte Eigenschaft nicht erfüllt sein darf. Die Anfrage

```
?- elternteil(leo,X), \+maennlich(X).
```

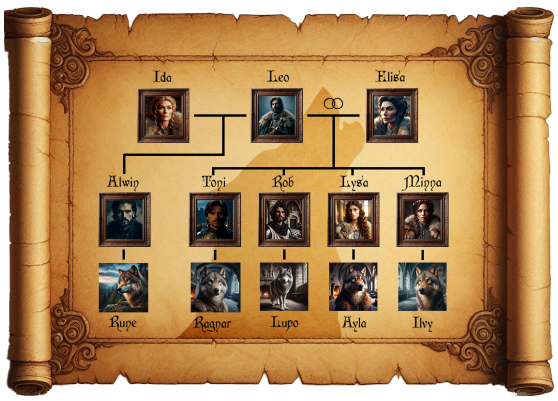
gibt alle Kinder von Leo aus, die nicht männlich sind. Für unsere Wissensbasis liefert diese Anfrage die Belegungen `X = lysa` und `X = minna` .

Arbeitsauftrag 4: Variablen und logische Verknüpfungen

(a) Erstellen Sie eine Anfrage in PROLOG, die die Eltern von Alwin ausgibt.

(b) Erstellen Sie eine Anfrage in PROLOG, die alle Kinder von Leo und Elisa anzeigt.

(c) Erstellen Sie eine Anfrage in PROLOG, die den Vater von Rob ausgibt.



Testen Sie die Anfragen anhand der bereits erstellten Wissensbasis in SWISH.

Merke (Bindung von Variablen)

Bei der Bindung von Variablen unterscheidet PROLOG die beiden Operatoren `=` und `is`, die unterschiedliche Zwecke erfüllen:

(a) `=` (Unifikationsoperator): PROLOG prüft, ob zwei Terme gleich gemacht werden können, indem es Variablen geeignete Konstanten zuweist. Dabei werden keine Berechnungen durchgeführt.

Beispiel:

Die Anfrage `?- X = 2, Y = X+3.` liefert als Antwort:

```
X = 2
Y = 2+3
```

(b) `is` (Berechnungsoperator): Der `is`-Operator führt eine berechnete Bindung durch. Der Ausdruck auf der rechten Seite wird vollständig ausgewertet, bevor der resultierende Wert an die Variable gebunden wird. Er wird vor allem für arithmetische Berechnungen verwendet.

Beispiel:

Die Anfrage `?- X = 2, Y is X+3` liefert als Antwort:

```
X = 2
Y = 5
```

Arbeitsauftrag 5: Bindung von Variablen durch die Operatoren = und is

(a) Betrachten Sie beiden Anfragen `?- X = 2, Y = 2*X-4` und `?- X = 2, Y is 2*X-4`. Stellen Sie Vermutungen an, was die jeweilige Antwort auf diese Anfragen ist. Führen Sie die Anfrage anschließend in SWISH aus:

(b) Betrachten Sie beiden Anfragen `?- X = 2, X = X+3` und `?- X = 2, X is X+3`. Stellen Sie Vermutungen an, was die jeweilige Antwort auf diese Anfragen ist. Führen Sie die Anfrage anschließend in SWISH aus. Deuten Sie die Ausgabe im Kontext des Variablenparadigmas in PROLOG.

Hinweis: Sie können die detaillierte Abarbeitung der obigen Anfragen schrittweise nachvollziehen, indem Sie der Anfrage ein `trace,` voranstellen.

4. Regeln

Merke (Regeln in PROLOG)

Regeln bestehen aus einem **Regelkopf** und einem **Regelkörper**, getrennt durch `:-` . Der Kopf ist dabei eine Schlussfolgerung und der Körper enthält Bedingungen, die erfüllt sein müssen, damit der Kopf wahr wird. Die Bedingungen im Körper können durch logische Verknüpfungen, wie `,` (und) und `;` (oder) kombiniert werden. Der Körper kann eine beliebige Anzahl von Bedingungen (Prädikaten) enthalten.

Die Regel

```
mutter(X,Y) :- elternteil(X,Y), weiblich(X).
```

bedeutet somit, dass `X` die Mutter von `Y` ist, wenn `X` ein Elternteil von `Y` ist und `X` weiblich ist.

Regeln helfen somit, Redundanzen zu reduzieren, indem ähnliche Bedingungen nur einmal definiert und an mehreren Stellen im Programm verwendet werden.

Hinweis: Fakten sind genau genommen Regeln ohne Körper. So bedeutet `maennlich(leo).` , dass Leo männlich ist, unabhängig von weiteren Bedingungen.

Arbeitsauftrag 6: Analyse der Regel `mutter(X,Y)`

Öffnen Sie die Wissensbasis zum Stammbaum von Burg Wolfsfels in SWISH.

(a) Erweitern Sie Wissensbasis um die obige Regel zur Mutter-Beziehung.

(b) Geben Sie die Antwort auf folgende Anfrage an:

```
?- mutter(elisa,X).
```

(c) Geben Sie die Antwort auf folgende Anfrage an:

```
?- mutter(X,minna).
```

(d) Geben Sie die Antwort auf folgende Anfrage an:

```
?- mutter(elisa,alwin).
```

(e) Die schrittweise Abarbeitung einer Anfrage durch das PROLOG-System lässt sich mithilfe des trace-Befehls nachverfolgen. Stellen Sie die Anfrage

```
?- trace, mutter(X,minna).
```

und führen Sie diese durch wiederholtes Klicken auf die -Schaltfläche aus. Erläutern Sie, wie die Anfrage durch PROLOG abgearbeitet wird.

Arbeitsauftrag 7: Weitere Eltern-Kind-Beziehungen

(a) Erstellen Sie eine Regel `vater(X,Y)`.

(b) Erstellen Sie die Regeln `sohn(X,Y)`, `tochter(X,Y)` und `kind(X,Y)` und überlegen Sie sich eine geeignete Reihenfolge der Regeldefinitionen.

(c) Testen Sie die erstellten Regeln durch geeignete Anfragen an das PROLOG-System.

Arbeitsauftrag 8: Bruder und Schwester

Die Wissensbasis soll um die Regeln `bruder(X,Y)`, `schwester(X,Y)` und `geschwister(X,Y)` erweitert werden.

(a) Definieren Sie zunächst eine Regel für Bruder und Schwester und leiten Sie daraus eine Regel für Geschwister ab.

(b) Definieren Sie zunächst eine Regel für Geschwister und leiten Sie daraus eine Regel für Bruder und Schwester ab.

(c) Überprüfen Sie Ihre Regeln anhand geeigneter Anfragen an das `PROLOG-System` und erläutern Sie, welches Vorgehen geschickter ist.

Arbeitsauftrag 9: Halbbruder und Halbschwester

(a) Erstellen Sie die Regeln `halbbruder(X,Y)`, `halbschwester(X,Y)` und `halbgeschwister(X,Y)`.

(b) Überprüfen Sie Ihre Regeln anhand geeigneter Anfragen an das `PROLOG-System`.

Merke (Prädikat (allgemein))

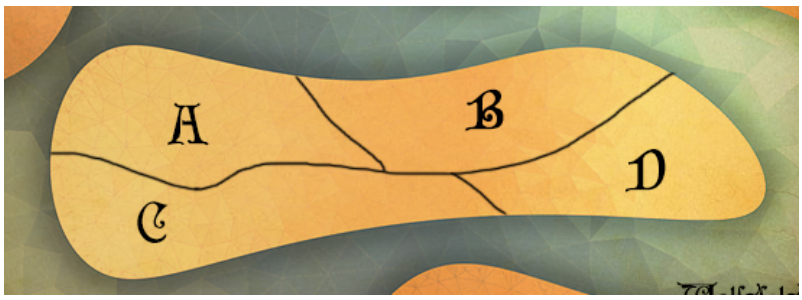
Alle Fakten und Regeln einer Wissensbasis, die den gleichen Bezeichner und die gleiche Stelligkeit haben, bilden ein **Prädikat**.

5. Strategie der automatisierten Ableitung von Aussagen durch das PROLOG-System

Arbeitsauftrag 10: Färben von Landkarten

Die Insel, bestehend aus den vier Gebieten A, B, C und D, soll mit den drei Farben gelb, rot und blau so eingefärbt werden, dass keine gleichfarbigen Gebiete aneinandergrenzen.

(a) Lösen Sie Aufgabe zunächst ohne PROLOG. Geben Sie insbesondere an, wie viele verschiedenen Lösungen es gibt:



(b) Betrachten Sie die unten angegebenen Fakten und die Regel `einfaerbung(A,B,C,D)`. Übertragen Sie die Wissensbasis in `SWISH` und vergleichen Sie Ihre Lösungen aus (a) mit den Antworten der Anfrage `?- einfaerbung(A,B,C,D).`

```
farbe(rot).
farbe(blau).
farbe(gelb).
einfaerbung(A,B,C,D) :- farbe(A), farbe(B), farbe(C), farbe(D), A\=B, A\=C, B\=C,
    B\=D, C\=D.
```

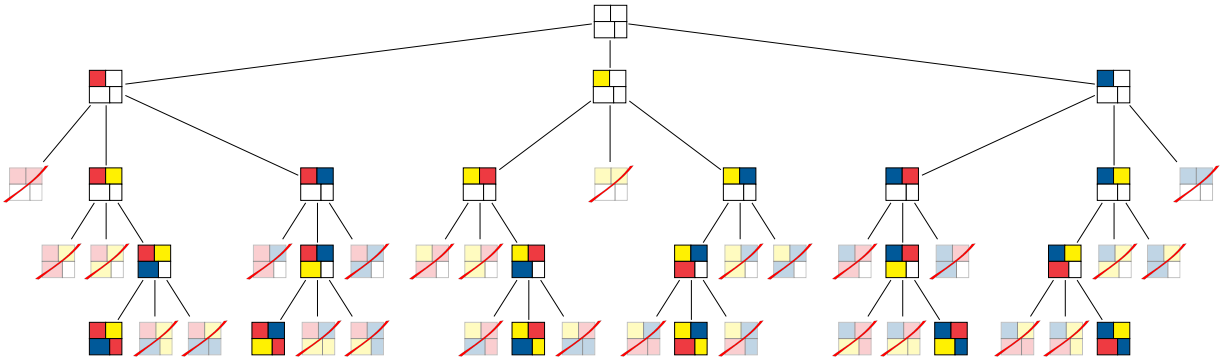
(c) Analysieren Sie mithilfe des `trace`-Befehls die Abarbeitung der Anfrage `?- einfaerbung(A,B,C,D).` durch das `PROLOG`-System. Erläutern Sie die Vorgehensweise und geben Sie an, ob diese effektiv ist.

(d) Überlegen Sie, ob sich das Programm optimieren lässt. Nehmen Sie hierzu ggf. Änderungen an der Regel `einfaerbung(A,B,C,D)` vor und testen Sie mithilfe des `trace`-Befehls, ob bzw. was sich nun ändert.

Merke (Backtracking)

Die Strategie, die `PROLOG` verwendet, um die Lösungen für eine Anfrage zu finden, heißt **Backtracking**. `PROLOG` prüft zuerst die Bedingungen einer Regel oder Anfrage von links nach rechts. Wenn eine Bedingung fehlschlägt, geht `PROLOG` automatisch zum letzten Entscheidungspunkt zurück und versucht eine andere Möglichkeit der Variablenbelegung.

Die Abarbeitung der Anfrage `einfaerbung(A,B,C,D)` an die optimierte Wissensbasis aus obigem Arbeitsauftrag lässt sich schön mithilfe eines Baums darstellen:



Die Abarbeitung der Anfrage entspricht einer Tiefensuche durch den Baum mit allen möglichen Variablenbelegungen. Wird ein Szenario entdeckt, bei dem keine Lösung des Problems mehr möglich ist, wird im Baum „zurückgesprungen“. Wird eine Lösung entdeckt, wird diese gespeichert bzw. ausgegeben und anschließend ebenfalls im Baum „zurückgesprungen“, um nach weiteren Lösungen zu suchen.

Arbeitsauftrag 11: Färben von Landkarten

(a) Erstellen Sie zu den obigen beiden Landkarten jeweils eine geeignete Wissensbasis sowie eine Anfrage, mit der die Lösungen aller Einfärbemöglichkeiten gefunden werden können.

(b) Für eines der beiden Szenarien reichen drei Farben nicht aus, um alle Bedingungen zu erfüllen. Erweitern Sie das Programm um eine vierte Farbe und prüfen Sie, ob es nun möglich ist, die Karte entsprechend einzufärben.

Exkurs (Der Vierfarbensatz)

Der **Vierfarbensatz** besagt, dass jede flache, zusammenhängende Landkarte auf einer Ebene so eingefärbt werden kann, dass keine benachbarten Regionen die gleiche Farben haben, und dabei höchstens vier Farben benötigt werden.

Auch wenn die Aussage zunächst einfach wirken sollte, handelt es sich hierbei um den vermutlich kontroversesten Beweis der Mathematikgeschichte. Das Problem wurde 1852 erstmals vermutet, aber erst über 120 Jahr später im Jahr 1976 von Kenneth Appel und Wolfgang Haken bewiesen. Es war dabei der erste bedeutende mathematische Satz, der mithilfe eines Computers bewiesen wurde, was heftige Diskussionen auslöste.[a]

[a] https://www.spektrum.de/kolumne/vier-farben-satz-der-kontroverseste-beweis-der-mathematikgeschichte/2163954.

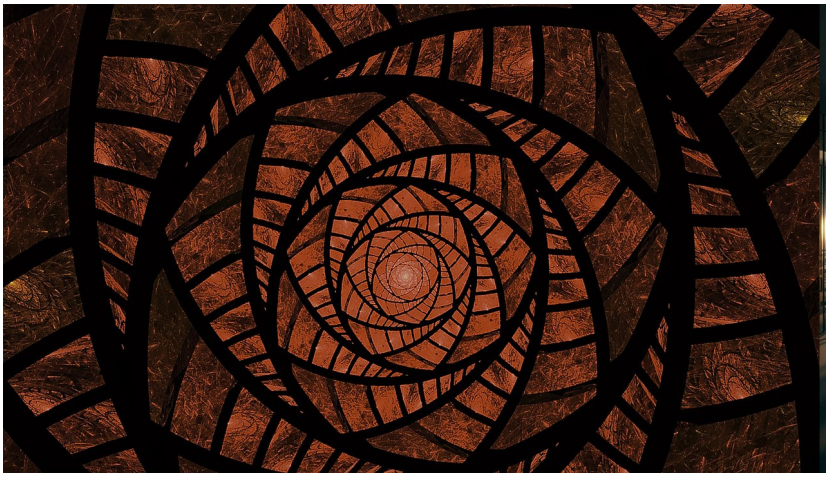
Arbeitsauftrag 12: Weiteres Färbeproblem

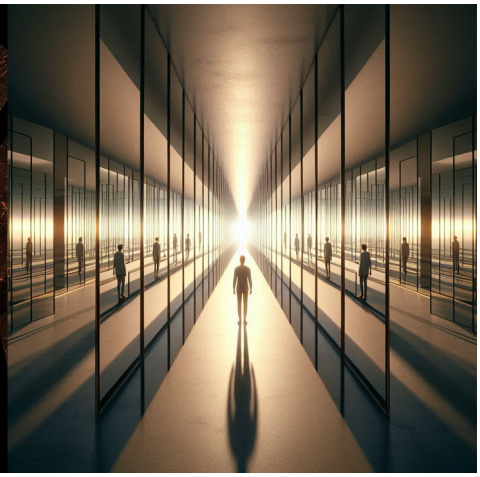
Erstellen Sie zu nebenstehender Landkarte eine geeignete Wissensbasis sowie eine Anfrage an diese, mit der alle möglichen Färbungen bestimmten werden können, wenn erneut keine zwei benachbarten Regionen die gleiche Farbe besitzen dürfen.

6. Rekursion in PROLOG

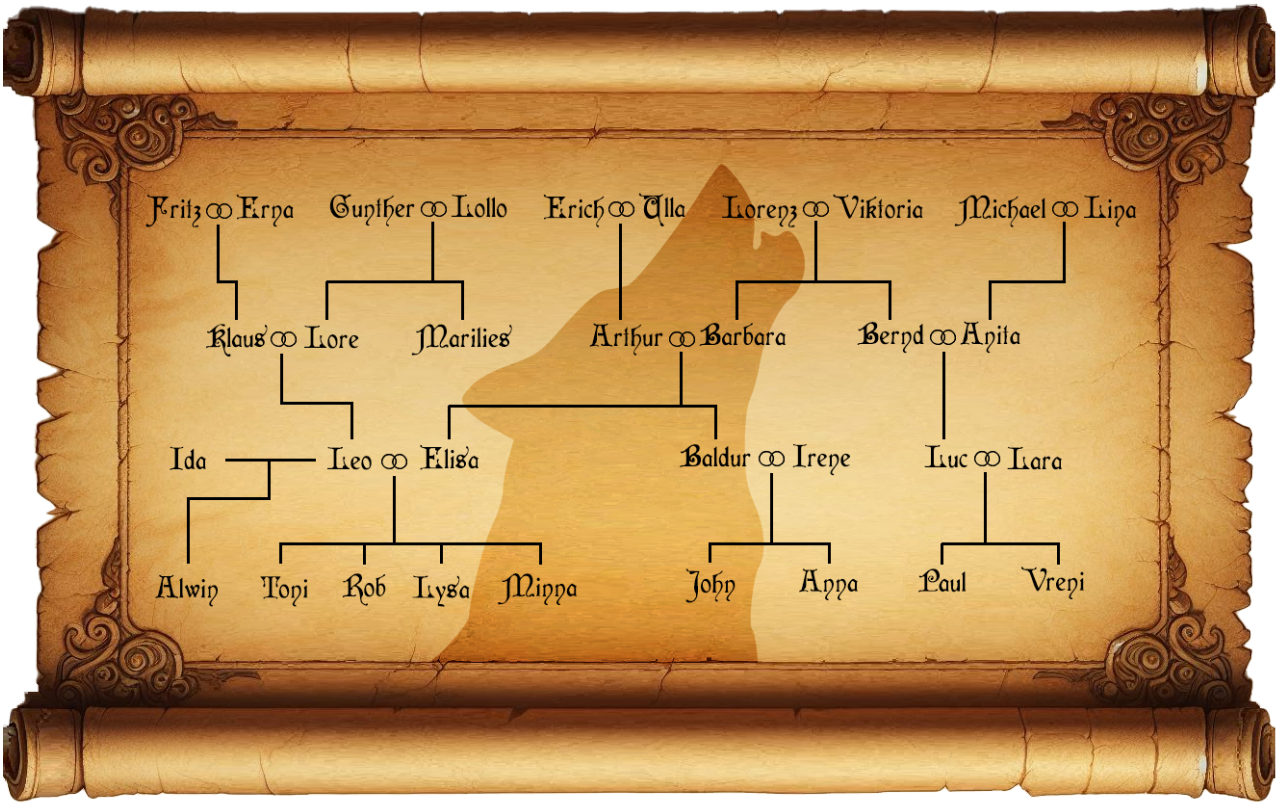
Um Rekursion zu verstehen, muss man entweder einen kennen, der sie versteht, oder sie schon verstanden haben.

M. FREERICKS





Für nachfolgende Aufgaben betrachten wir nun folgende erweiterte Version des Stammbaums von Wolfsfels:



Arbeitsauftrag 13: Die Vorfahren von Burg Wolfsfels

Öffnen Sie die Datei `stammbaum_erweitert_vorlage.pl` in SWISH.

(a) Definieren Sie eine Regel `grosselternteil(X,Y)` bzw. `urgrosselternteil(X,Y)` sowie die zugehörigen weiblichen und männlichen Pendants.

(b) Überprüfen Sie Ihre Regeln durch geeignete Anfragen an die Wissensbasis.

(c) Geben Sie allgemein ein Vorgehen an, wie man auch Regeln für `ururgrosselternteil(X,Y)`, `urururgrosselternteil(X,Y)` etc. definieren kann (Regeln müssen nicht in SWISH angelegt werden).

Arbeitsauftrag 14: Die Regel `vorfahr(X,Y)`

Als Verallgemeinerung der Regeln `grosselternteil`, `urgrosselternteil`, etc. soll eine Regel `vorfahr(X,Y)` definiert werden. So soll die Anfrage

```
?- vorfahr(X,paul).
```

alle Vorfahren von Paul ausgeben.

(a) Machen Sie sich Gedanken, wie Sie diese Regel definieren würden. Beschreiben Sie, welche Probleme dabei auftreten.

(b) In PROLOG lässt sich die Regel `vorfahr(X,Y)` wie folgt definieren:

```
vorfahr(X,Y) :- elternteil(X,Y).
vorfahr(X,Y) :- elternteil(Z,Y), vorfahr(X,Z).
```

(c) Ergänzen Sie die Regel `vorfahr(X,Y)` in der Wissensbasis und analysieren Sie mithilfe des `trace`-Befehls die Abarbeitung der Anfrage

```
?- vorfahr(X,paul).
```

durch das PROLOG-System.

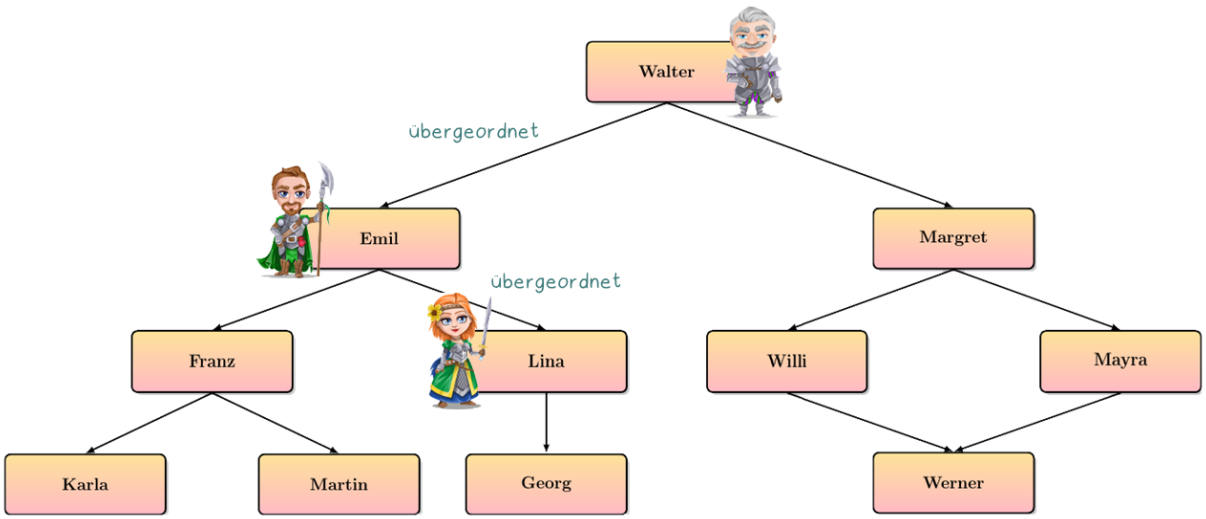
Arbeitsauftrag 15: Vorfahrgrad

Erweitern Sie die Regel aus dem vorherigen Arbeitsauftrag so, dass neben den Vorfahren gleichzeitig auch der sog. Vorfahrgrad ausgegeben wird. Eltern besitzen dabei den Grad 1, Großeltern den Grad 2 etc.

Hinweis:

- Sie benötigen hierfür eine weitere Variable, die den Grad entsprechend „mitzählt“
- Beachten Sie den Merke-Kasten zur Bindung von Variablen auf Seite 9 in PROLOG und dass eine Variable immer nur einmal an einen Wert gebunden werden kann.

Arbeitsauftrag 16: Hofstaat auf Burg Fuchsspitze



(a) Öffnen Sie das Programm `hofstaat_fuchsspitze.pl` in SWISH und untersuchen Sie, welche Fakten zur Hierarchie im Hofstaat schon umgesetzt sind.

(b) Überprüfen Sie mit geeigneten Anfragen, ob die drei abgebildeten „Übergeordnet“-Beziehungen

anhand der gegebenen Wissensbasis abgeleitet werden können:

(c) Definieren Sie eine Regel, mit der auch „Überübergeordnete“, „Überüberübergeordnete“, etc. als übergeordnet erkannt werden. Überprüfen Sie Ihre Regel anhand mehreren Anfragen.

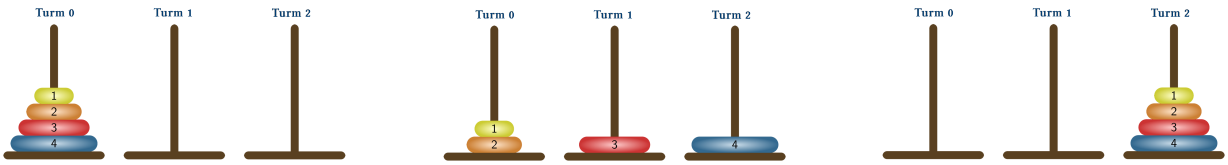
(d) Für den Fall, dass Anfragen aus (c) das `Stack limit` überschreiten und somit abstürzen: Analysieren Sie mithilfe des `trace`-Befehls eine Anfrage, die zu keinem Ende kommt und erklären Sie, warum dies so ist. Ändern Sie die Wissensbasis bzw. die Regel so ab, dass sie das Gewünschte leistet.

Arbeitsauftrag 17: Die Türme von Reitersberg

Die Burg Reitersberg war bekannt für Ihre markanten Türme. Hieraus entwickelte sich ein Spiel, das die Strategie und den Verstand seiner Spieler herausforderte.

Das Spiel bestand aus drei Türmen und einer bestimmten Anzahl von unterschiedlich großen Scheiben, die zunächst auf dem linken der Türme gestapelt werden, wobei keine größere Scheibe auf einer kleineren Scheibe liegen darf. Das Ziel des Spiels ist es, alle Scheiben von dem linken Stab auf den rechten Stab zu bewegen. Dabei darf in einem Zug nur eine Scheibe bewegt werden. Eine Scheibe kann nur auf einen leeren

Turm oder auf einen Turm gelegt werden, wenn die oberste Scheibe dort größer ist als die bewegte Scheibe.



Ausgangssituation | Zwischenstand | Spielende

(a) Notieren Sie eine Lösung des Spiels „Die Türme von Reitersberg“ für zwei Scheiben, drei und vier Scheiben, indem du die Zugfolgen wie in nachfolgendem Hinweis dokumentierst.

Hinweis: **0 → 1** bedeutet, dass eine Scheibe vom linken Turm auf den mittleren Turm verschoben wurde.

(b) Erläutern Sie, inwiefern Rekursion beim Spiel „Die Türme von Reitersberg“ eine Rolle spielt. Formulieren Sie einen rekursiven Algorithmus in Pseudocode, mit dem n Scheiben von Turm 0 auf Turm 2 verschoben werden können.

(c) Begründen Sie weiter, wie viele Züge bei einem Spiel mit n Scheiben mindestens notwendig sind.

(d) Betrachte folgende beiden Regeln zum Prädikat `bewege/3`:

```
bewege(1,Von,Nach) :- write('Zug: '), write(Von), write(' nach '), write(Nach),
    writeln('').
bewege(N,Von,Nach) :- N > 1, M is N-1, Ablage is 3-Von-Nach,
                      bewege(M,Von,Ablage),
                      bewege(1,Von,Nach),
                      bewege(M,Ablage,Nach).
```

Geben Sie die Ausgabe der Anfragen `?- bewege(2,0,2)` und `?- bewege(3,0,2)` an und überprüfen Sie Ihr Ergebnis, indem Sie das Programm in SWISH übertragen und die entsprechenden Anfragen stellen.

(e) Erläutern Sie die Bedeutung der einzelnen Bestandteile der Regel `?- bewege(N,Von,Nach)` :

(f) Erläutern Sie, warum es nicht möglich ist, die beiden Regeln des Prädikats `bewege/3` wie folgt zu definieren:

```
bewege(1,Von,Nach) :- write('Zug: '), write(Von), write(' nach '), write(Nach),
    writeln('').
bewege(N,Von,Nach) :- N > 1, Ablage is 3-Von-Nach,
                      bewege(N-1,Von,Ablage),
                      bewege(1,Von,Nach),
                      bewege(N-1,Ablage,Nach).
```

Überprüfen Sie Ihre Überlegungen, indem Sie mithilfe des `trace`-Befehls die Abarbeitung der Anfrage `?- bewege(3,0,2)` analysieren.

Arbeitsauftrag 18: Vorfahrgrad (Fortsetzung)

Betrachten Sie folgende beiden Regeln zum Prädikat `vorfahr/3` im Kontext des erweiterten Stammbaums von Seite 18:

```
vorfahr(X,Y,1) :- elternteil(X,Y).
vorfahr(X,Y,N) :- M is N-1, elternteil(X,Z), vorfahr(Z,Y,M).
```

Die Anfrage `?- vorfahr(X,Y,2).` liefert passende Ergebnisse, aber `?- vorfahr(fritz,alwin,N).` liefert statt dem gewünschten Wert für `N` die Fehlermeldung:

```
Arguments are not sufficiently instantiated
In:
   [1] vorfahr(fritz,alwin,_1972)
```

Beheben Sie den Fehler! Erläutern Sie, warum man bei den Türmen von Reitersberg mit `M is N-1` arbeiten kann, aber bei der Bestimmung des Vorfahrgrades nicht.

7. Die Rätsel von Burg Wolfsfels[1]

Arbeitsauftrag 19: Ross und Reiter

Die drei Geschwister Toni, Rob und Lysa ritten mit ihren Pferden durch die Wälder rund um Burg Wolfsfels. Eines der Pferde war schwarz, eines braun und eines weiß. Es ist bekannt, dass Toni nicht auf dem schwarzen und Rob nicht auf dem braunen Pferd sitzt. Die Person auf dem braunen Pferd ist jünger als Toni.

(a) Ermitteln Sie zunächst ohne PROLOG, welche Person auf welchem Pferd reitet.

(a) Betrachten Sie folgende Fakten- und Regeldefinitionen in PROLOG:

```
pferd(schwarz).
pferd(braun).
pferd(weiss).

loesung(Pferd_Toni,Pferd_Rob,Pferd_Lysa) :- pferd(Pferd_Toni), pferd(Pferd_Rob),
    pferd(Pferd_Lysa), Pferd_Toni\=Pferd_Rob, Pferd_Toni\=Pferd_Lysa,
    Pferd_Rob\=Pferd_Lysa, Pferd_Toni\=schwarz, Pferd_Rob\=braun,
    Pferd_Toni\=braun.
```

Öffnen Sie die Datei `ross_reiter.pl` in SWISH und stellen Sie die Anfrage

```
?- loesung(Pferd_Toni,Pferd_Rob,Pferd_Lysa).
```

Vergleichen Sie Ihre Lösung mit der Lösung von PROLOG und geben Sie an, ob die Lösung eindeutig ist.

[1] Grundlegende Ideen der Rätsel basieren auf dem Arbeitsheft PROLOG von H. Göhner und B. Hafenbrak.

(b) In obiger Wissensbasis wurden die Pferde als Fakten angegeben. Es ist aber genauso möglich, die Personen als Fakten zu definieren. Ändern Sie das Programm entsprechend ab, so dass sich damit ebenfalls die Lösung des Rätsels ermitteln lässt.

Arbeitsauftrag 20: Üben für das Ritterturnier

Lysa, Minna, Anna und Vreni trainieren zusammen mit Toni, Rob, John und Paul für das alljährliche Ritterturnier. Hierbei gibt es die vier Disziplinen Schwertkampf, Bogenschießen, Lanzenstechen und Zweikampf. Am heutigen Tag üben immer jeweils ein Junge und ein Mädchen zusammen, wobei darüber hinaus Folgendes bekannt ist: Lysa will heute unbedingt Schwertkampf üben, Minna übt zusammen mit Rob, Anna möchte auf keinen Fall mit ihrem Bruder John üben, Paul schießt mit den Bogen, John wurde mit einer Lanze gesichtet.

(a) Überlegen Sie sich, welche Informationen zum Fakt, welche zur Variable werden. Erstellen Sie anschließend ein Programm in `PROLOG`, mit dem sich bestimmen lässt, wer mit wem welche Disziplin geübt hat. Geben Sie an, ob die Lösung eindeutig ist.

(b) Probieren Sie auch die anderen Möglichkeiten der Modellierung aus, d.h. definieren Sie andere Informationen als Fakten. Führen alle Modellierungen zum Erfolg?

Arbeitsauftrag 21: Tafel im Rittersaal

Nach einem weiteren Trainingstag für das große Ritterturnier nehmen Toni, Rob, John und Paul in genau dieser Reihenfolge an der großen Tafel im Rittersaal nebeneinander Platz. Jeder der vier hat heute eine unterschiedliche Disziplin. Der Junge im roten Wams hat heute mit dem Schwert gekämpft, Paul war Bogenschießen, Rob sitzt neben demjenigen, der mit der Lanze geübt hat, der Junge im blauen Wams sitzt neben dem Bogenschützen und derjenige, der Zweikampf geübt hat, sitzt nicht neben dem Bogenschützen.

(a) Geben Sie die Fakten der Wissensbasis an. Modellieren Sie dabei auch die Informationen, wer neben wem sitzt als Fakten. Beachten Sie hierbei die Kommutativität.

(b) Erstellen Sie eine Regel `loesung(LS)` , mit der sich ermitteln lässt, wer mit der Lanze gekämpft hat.

Hinweis: „NICHT“ in PROLOG: `\+` (vorangestellt).

(c) Deuten Sie das Ergebnis der Anfrage `?- loesung(LS).`

Arbeitsauftrag 22: Kräuterkunde

Lysa, Minna und Anna müssen im Rahmen von Kräuterkunde einmal wöchentlich die drei Kräuterhexen Aurelia, Elowen und Sylvia besuchen. Der Plan sieht vor, dass jeweils eine von ihnen am Mittwoch, Donnerstag bzw. Freitag eine der drei Hexen besucht, um von diesen zu lernen. Weder Lysa noch Minna und Anna wollen mehrmals in einer Woche gehen, so dass jede genau einmal zu einer Kräuterhexe muss. Weiter haben die drei Mädchen noch einige Sonderwünsche:

- Niemand will am Mittwoch zu Aurelia.
- Lysa will nicht donnerstags fahren und will nicht zu Sylvia.
- Minna will nicht am Mittwoch fahren und will nicht zu Elowen; zudem will sie freitags nicht zu Aurelia.
- Anna will nicht am Freitag fahren und möchte nicht zu Aurelia.

Ermitteln Sie mit einer geeigneten Regel in PROLOG, ob alle Sonderwünsche erfüllbar sind und geben Sie die möglichen „Stundenpläne“ für den Kräuterkunde-Unterricht an.

Arbeitsauftrag 23: Das vierjährige Turnier

Alle vier Jahre ermitteln die fünf großen Königshäuser Wolfsfels, Fuchsspitze, Reitersberg, Heuerstein und Dachsriegel untereinander den größten Krieger bzw. die größte Kriegerin. Dafür entsendet jedes der Königshäuser den besten Kandidaten zu einem spektakulären Turnier, in dem die Krieger in unterschiedlichen Disziplinen gegeneinander antreten müssen.

Während des Turniers sind die Krieger jeweils in einem von fünf verschiedenfarbigen Zelten untergebracht, die direkt nebeneinander in einer Reihe angeordnet.

Die Krieger haben das Recht, eine Waffe, eine Rüstung sowie einen Glücksbringer mit zum Turnier zu bringen. Bekanntermaßen haben alle fünf Krieger unterschiedliche Vorlieben in diesen drei Bereichen.

Vom vergangengen Turnier ist leider nicht mehr genau bekannt, welcher Krieger welche Waffe, Rüstung bzw. Glücksbringer hatte bzw. in welchem der Häuser er untergebracht war. Allerdings wurden noch Aufzeichnungen gefunden, die zumindest einzelne Zusammenhänge abbilden:

(a) Der Krieger aus Wolfsfels war im roten Zelt.

(b) Der Krieger aus Fuchsspitze hatte eine Spieluhr dabei.

(c) Der Krieger im grünen Zelt hatte einen Plattenpanzer an.

(d) Der Krieger aus Heuerstein hatte ein Kettenhemd an.

(e) Das grüne Zelt stand direkt neben dem schwarzen Zelt.

(f) Im gelben Zelt war der Krieger mit der Lanze untergebracht.

(g) Der Krieger mit dem Morgenstern hatte ein Hufeisen als Glücksbringer dabei.

(h) Der Krieger aus Reitersberg war in dem Zelt untergebracht, das neben dem blauen Zelt stand.

(i) Der Krieger, der im mittleren Zelt wohnte, hatte einen Schuppenpanzer an.

(j) Der Krieger aus Reitersberg war im ersten Zelt ganz links untergebracht.

(k) Der Krieger mit der Armbrust war Nachbar des Kriegers, der einen Runenstein als Talisman dabei hatte.

(l) Der Krieger mit der Lanze war Nachbar des Kriegers, der einen Edelstein dabei hatte.

(m) Der Krieger mit der Streitaxt hatte einen Harnisch an.

(n) Der Krieger aus Dachsriegel hatte ein Schwert dabei.

Modellieren Sie mithilfe der Angaben die Situation beim letzten großen Turnier mit geeigneten Fakten und einer Regel `loesung(...)` in `PROLOG`. Geben Sie insbesondere an, welcher Krieger den Lamellenpanzer anhatte bzw. welcher Krieger eine Goldmünze als Glücksbringer dabei hatte.

Hinweis: Der „Anfrage-Baum“ ist bei so vielen Variablen sehr groß. Dies kann bei einer „nicht optimierten“ Modellierung der Regel `loesung(...)` dazu führen, dass die Anfrage wegen Überschreitung des `Stack-Limit` abstürzt. Passen Sie deswegen den Regelkörper derart an, dass Sie Beziehungen zwischen Variablen modellieren, sobald dies möglich ist. Durch effiziente Modellierung kann die Lösung sehr schnell gefunden werden.

8. Logikprogrammierung und PROLOG[2]

Ein logisches Programm ist eine Menge von Axiomen oder Regeln, die Beziehungen zwischen Objekten definieren. Eine Berechnung eines logischen Programms ist eine Ableitung von Eigenschaften und Zusammenhängen aus dieser Menge.

(Sterling & Shapiro, 1994, S. 9)

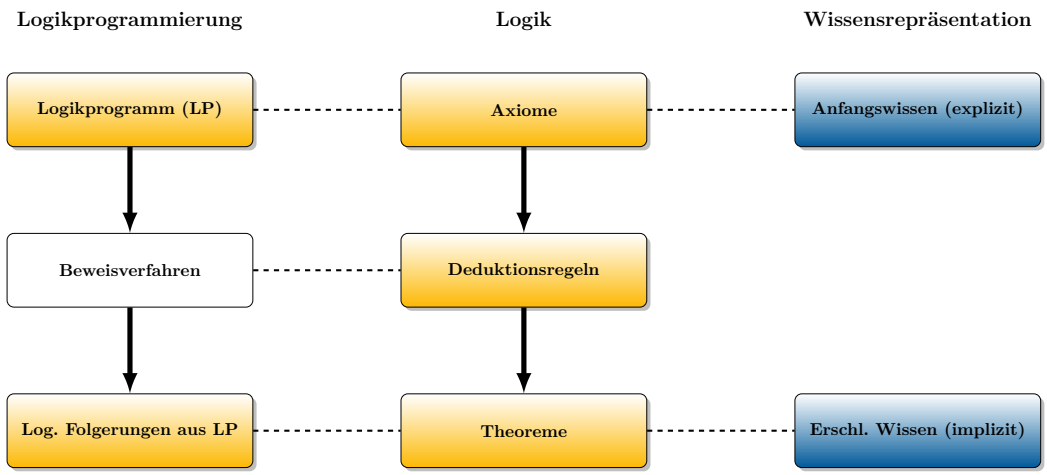
Merke (Logikprogramm)

Ein Logikprogramm lässt sich in zwei zentrale „Phasen“ unterteilen, die **Entwicklung** und die **Ausführung** des Logikprogramms:

(a) Bei der Entwicklung eines Logikprogramms beschreibt man das für die Lösung einer Aufgabenstellung relevante Wissen, indem man alle Objekte und die zwischen ihnen bestehenden Beziehungen in Form von logischen Axiomen notiert.

(b) Bei der Ausführung eines Logikprogramms wird versucht, die durch eine Frage formulierte Zielaussage aus dem Programm abzuleiten.

Ein Logikprogramm kann somit als eine Art der **Wissensrepräsentation** betrachtet werden. Das Logikprogramm kodiert zunächst das verfügbare Anfangswissen bzw. **explizite Wissen**. Die aus dem expliziten Wissen ableitbaren Folgerungen stellen das erschließbare Wissen bzw. **implizite Wissen** dar.



Merke (PROLOG)

PROLOG ist eine deklarative Programmiersprache, die sich auf die Verarbeitung von symbolischen Informationen (logische Axiome) spezialisiert hat. Die Wissensbasis wird hierzu als eine Sammlung von **Fakten** und **Regeln** ausgedrückt. An diese Wissensbasis können dann Anfragen gestellt werden. Grundlegende Eigenschaften von PROLOG sind unter anderem:

[2] Dieser Abschnitt orientiert sich in Grundzügen an einer Vorlesung „Prolog – Eine Einführung“ von Sven Naumann aus dem Sommersemester 2007 der Universität Trier.

(a) **Deklarative Programmierung:** In Prolog beschreibt man, was man erreichen möchte, anstatt explizit den Weg dorthin zu programmieren. Dies führt zu einem deklarativen Programmierstil, bei dem der Fokus auf der Definition von Beziehungen und Bedingungen liegt.

(b) **Rekursion:** Prolog unterstützt natürliche Rekursion und ist gut geeignet für Probleme, die auf rekursiven Algorithmen basieren.

(c) **Backtracking:** Prolog verwendet Backtracking, um alternative Lösungen zu finden. Wenn eine Lösung nicht erfolgreich ist, kehrt das System zu vorherigen Entscheidungspunkten zurück und versucht alternative Wege.

(d) **Anfragen:** In Prolog werden Anfragen gestellt, um Beziehungen und Bedingungen zu überprüfen oder um mögliche Werte für Variablen zu finden.

Der Name `PROLOG` steht für **Programmieren in Logik**. Die Sprache wurde in den 1970er Jahren von Alain Colmerauer und Robert Kowalski entwickelt.