Die Online-Befragung zur gewählten alternativen Veranstaltungsform ist noch weiter online, es wird noch dringend gebeten den Bogen auszufüllen, wenn noch nicht geschehen. Bitte ausfüllen: https://forms.gle/UV19w4NxyLLe56fh9. Sie werden eventuell aufgefordert sich bei Google anzumelden, das ist nur notwendig, wenn Sie in der Bearbeitung eine Pause machen wollen und das Teilergebnis zwischenspeichern wollen. (Bitte auch ausfüllen, wenn Sie einen ähnlichen Bogen für OOAD ausfüllen.)

Frage: Ich habe das mit dem Cut-Operator nicht genau verstanden, wann er genutzt werden sollte.

Antwort: Verständlich, da die Regel nur besagt, dass er eingesetzt werden kann, wenn es sicher ist, dass für den Weg bis zur Ausführung des Cut-Operators keine Alternativen mehr betrachtet werden sollen. Das ist durchaus schwammig, bedeutet praktisch aber, dass zunächst eine Lösung ohne Cut-Operator formuliert werden sollte und dass danach z. B. aus Performance-Gründen oder da es nur eine Lösung geben soll, über die Nutzung des Cut-Operators nachgedacht werden soll.

„Es ist schwer möglich allgemeine Kriterien für die sorgfältige Verwendung des „Cut“ anzugeben“ (M. Hanus, Problemlösen mit PROLOG, 2. Auflage, S.100, Teubner, Stuttgart,1987)

Relativ einfach ist das bei den genutzten Testfällen, die einfach einmal erfolgreich durchlaufen sollen, da ist der Cut-Operator am Ende passend.

Um die Problematik zu verdeutlichen hier ein Beispiel für eine falsche Generalisierung.

Falsche Annahme: Wenn ich bei Fakten angekommen bin, ist ja eine Lösung gefunden, deshalb setze ich hinter jeden Fakt einen Cut-Operator.

```
fakt1(42) :- !.
fakt1(43) :- !.
suche1(X) :- fakt1(X), X>42.
```

Die folgenden Anfragen suggerieren zunächst, dass es funktioniert. Die letzte Anfrage zeigt, dass der Cut-Operator die Suche nach einer neuen Lösung abschneidet und so kein passender Wert gefunden wird.

```
?- suche1(43).
true.

?- suche1(42).
false.

?- suche1(X).
false.
```

Mit einer kleinen Änderung, es gibt nur einen passenden Fakt, funktioniert der Ansatz aber. Wichtig ist, dass der Cut-Operator nur besagt, dass für alle Literale vor dem Cut-Operator kein Backtracking, also eine Suche nach neuen Lösungen erfolgt. Da in suche2 selbst kein Cut-Operator steht, findet eine normale Auswertung statt.

```
vorher(0).
vorher(1).
fakt2(42) :- !.
suche2(V,X) :- vorher(V), fakt2(X), Tmp is X + V, Tmp > 42.
```

```
?- suche2(A,B).
A = 1,
B = 42.
```

Mit einem falsch gesetzten Cut-Operator gibt es ein Problem, dabei kann der Cut-Operator bei fakt2(42) für das folgende Beispiel sogar weggenommen werden. Es ist wichtig, in welcher Regel oder welchem Fakt der Cut-Operator steht.

```
suche3(V,X) :- vorher(V), fakt2(X), !, Tmp is X + V, Tmp > 42.
```

```
?- suche3(1,42).
true.

?- suche3(A,B).
false.
```

Ein Cut-Operator darf auch nicht ohne Nachdenken an das Ende einer Regel gesetzt werden:

```
fakt4(42).
regel(V,X) :- vorher(V), fakt4(X), !.
suche5(V,X) :- regel(V,X), Tmp is X + V, Tmp > 42.

?- suche5(1,42).
true.

?- suche5(A,B).
false.
```

Frage: Kann man mit Drools, ähnlich zu Prolog, auch alle Lösungen berechnen lassen?

Antwort: Theoretisch ja, aber. Drools ist dafür ausgelegt mit einer Tiefensuche eine Lösung zu finden. Da in den Berechnungen der Zugriff auf Minimums-, Maximums und vergleichbare Berechnungen möglich ist, kann so auch eine optimale Lösung gefunden werden. Generell können Regeln mehrfach gefeuert werden und es sind weitere Eingriffe in die Auswahl der von Drools genutzten Regeln möglich. Das sollte aber möglichst selten gemacht werden, da es schnell passieren kann, dass der Überblick verloren geht, was genau in der Drools-Engine passiert. Sollten mehrere Lösungen interessant sein, kann über Listen von Lösungen nachgedacht werden, da Listen sehr gut mit Drools bearbeitet werden können. Generell sollte aber das erste Ziel sein, die beste Lösung zu berechnen.