1 Berechnen der *first* Relation (5 Punkte)

Sie sollen für alle Nichtterminale die Menge von Terminalen berechnen, die in der *first* Relation zu diesen stehen. Der Algorithmus funktioniert folgendermaßen:

Als Initialisierung werden für alle Regeln, deren rechte Seite mit einem Terminal beginnt, dieses Terminal zu den `first` Terminalen der linken Seite der Regel hinzugenommen und das Nichtterminal der linken Seite als aktiv markiert.

Dann werden alle aktiven Nichtterminale behandelt:

Wenn ein aktives Nichtterminal A als erstes Element auf der rechten Seite einer Regel auftaucht, werden alle `first` Terminale von A auch zu denen der linken Seite (B) hinzugefügt. Verändert sich die Menge `first` von B, wird B als aktiv markiert. Nach der Abarbeitung aller in Frage kommenden Regeln wird A als nicht aktiv markiert.

Dieser Prozess wird so lange fortgesetzt, bis die Menge der aktiven Nichtterminale leer ist.

Implementieren Sie die `void calculateFirstMap(Collection<Rule> rules)` Methode in `Grammar.java`, die die entsprechende Menge von Terminalen für die Nichtterminale berechnet.

2 Top Down Parsing (5 Punkte)

In `cfg.zip` ist auch die Implementierung eines Top-Down Parsers in `TopDownParser.java` verfügbar. Implementieren Sie die Methode `getRulesTopDown(String nonTerminal, String firstTerminal)` in `Grammar.java`. Die Funktion darf in jedem Fall nur die Regeln zurückliefern, auf deren linker Seite `nonTerminal` steht.

Falls Sie eine funktionierende Implementierung von Aufgabe 1 haben, sollte diese Funktion nur die Regeln zurückliefern, auf denen das erste (Nicht)terminal auf der rechten Seite in *first*-Relation mit dem `firstTerminal`-Parameter steht.

Fügen Sie zu `Grammar` eine Index-Datenstruktur hinzu, die entsprechend vorberechnete Regelmengen enthält, so dass während des Parsens ein effizienter Zugriff möglich ist.

3 Memoization (3+2 Punkte)

1. Implementieren Sie die Methoden in `TopDownParser` so, dass die rekursiven Aufrufe memoisiert werden. Da der Parameter, der den input transportiert, sich nicht ändert, kann er ignoriert werden. Aufrufe, in denen `toRecognize` ein Terminal ist, werden nicht memoisiert.
2. Denken Sie sich eine Testgrammatik und -eingabe aus, mit denen Sie den Laufzeitunterschied mit und ohne Memoization zeigen.

Abgabetermin ist der 2. Februar

Bitte nur in gewohnter Form an steffen@dfki.de