

Übungen zur „Deskriptiven Programmierung“ Blatt 8

Aufgabe 15. Alle Prolog-Implementierungen bieten verschiedene Systemprädikate an, um die Struktur von Termen zu analysieren. Erweitere unseren Prolog-Interpreter um folgende primitive Relationen.

atom / 1
compound / 1
functor / 3
arg / 3

Das Ziel *atom* (X) ist wahr, wenn X ein Atom, ein 0-stelliges Funktionssymbol, ist; *compound* (X) ist wahr, wenn X ein zusammengesetzter Term ist. Das Prädikat *functor* ($Term, F, Arity$) ist wahr, wenn $Term$ ein Term ist, dessen oberstes Funktionssymbol den Namen F und die Stelligkeit $Arity$ hat. Zum Beispiel ist der Aufruf *functor* (*node* ($e, 2, e$), *node*, 3) erfolgreich. Während mit *functor* auf das Funktionssymbol eines Terms zugegriffen werden kann, erlaubt *arg* auf die Argumente eines Terms zuzugreifen. Das Ziel *arg* ($N, Term, Arg$) ist wahr, wenn Arg das N -te Argument von $Term$ ist. Zum Beispiel ist *arg* (3, *node* ($e, 2, e$), e) erfolgreich.

Definiere zunächst geeignete Haskell Funktionen, die auf dem *Term* Datentyp arbeiten und überlege dann, wie primitive oder externe Relationen in den Prolog-Interpreter integriert werden können. Beachte dabei, daß die Relationen unterschiedlich ‘aktiviert’ werden können.

Aufgabe 16. Verwende die primitiven Prädikate aus der vorherigen Aufgabe, um die folgende Relation zu implementieren.

subterm / 2

Das Ziel *subterm* ($T1, T2$) ist wahr, wenn $T1$ ein Teilterm von $T2$ ist. Für welche Aktivierungsformen funktioniert die Implementierung?