

Beispielmakro: Idempotent_unary_polynoms_ax^n_in_groups.omf

Makro zur Bestimmung idempotenter Polynome der Form ax^n in Gruppen

Allgemeine Bemerkungen:

Die Berechnung der idempotenten Polynome der Form ax^n ist eigentlich nicht auf Gruppen beschränkt. Der Anwender kann die Struktur auswählen, die untersucht werden soll, wie auch die betrachtete binäre (Gruppen-)Operation. Der Algorithmus ist naiv und ohne zusätzliche Heuristiken oder Kenntnis über die Struktur konstruiert: Für alle a und für alle $n < \text{card}(G)$ wird geprüft, ob $a(ax^n)^n = ax^n$ für alle Elemente x . Der unformatierte Quelltext einer Variante für zwei Operationen und Polynome der Form $ax^n + b$ findet sich in [Idempotent_unary_polynoms_ax^n + b_in_rings.omf](#).

Die Implementation (Download: unformatierter Quelltext in [Idempotent_unary_polynoms_ax^n_in_groups.omf](#)):

```
% Das Makro "idpot_tools" (siehe unten) enthält Hilfsfunktionen (lambda-Makros), die hier
verwendet werden. Damit usemakro korrekt funktioniert, muss der Quelltext aus
idpot_tools.omf in ein Makro-Fenster mit der Bezeichnung "idpot_tools" geladen werden. %
usemakro("idpot_tools");
% Der Benutzer wählt eine Struktur aus: %
Group:=listprompt("Select a structure", structures);
% Die binären Operationen der ausgewählten Struktur werden für die Auswahl vorbereitet: %
binary_operations:=emptyset;
for op in operations(Group) do
  if equal(arity(op),2) then additem(binary_operations, op) else noop endif
endfor;
% Wenn keine binäre Operation definiert ist, wird auch nichts gemacht bzw. nur eine Meldung
ausgegeben: %
if >(card(binary_operations),0) then
  % Der Benutzer wählt eine der binären Operationen: %
  op:=listprompt("Select a binary operation", binary_operations);
  % Ausgabe Überschrift: %
  write("Idempotent unary polynoms a*x^n in <"); write(Group); write(",");write(op);writeln(">");
  % Für alle n: %
  for n in card(Group) do
    % n=0 nicht interessant: %
    if >(n,0) then
      % Für alle g (unser a): %
      for g in Group do
        % Das Polynom wird als lambda-Makro dargestellt; op ist die ausgewählte Operation,
power_op stammt aus idpot_tools (siehe unten): %
        poly:=lambda(x) op(g,power_op(op,n,x)) endlambda;
        % Falls poly ein idempotentes Polynom implementiert, dann Ausgabe: %
        if idempotent(poly,Group) then write("a:=");write(g);write(" n:=");writeln(n)
        else noop endif
      endfor
    endif
  endfor
endif
```

```
    else noop endif
  endfor
else writeln("No binary operations!") endif;
```

Die Implementation von `idpot_tools.omf` (Download unformatierter Quelltext in [idpot_tools.omf](#)):

```
% power_op(f, n, x) := f^n(x), wobei f^I(x) := x und f^{i+I}(x) := f(f^i(x)); insbesondere für die
```

```
Gruppenoperation * folgt power_op(*, n, x) = x^n: %
```

```
power_op:=lambda(o,m,h)
  if leq(m,1) then h else o(h,power_op(o,-(m,1),h)) endif;
endlambda;
```

```
% idempotent(f, U) = true, falls f(x) = f(f(x)) für alle x aus U: %
```

```
idempotent:=lambda(f,U)
  forall(x in U, equal(f(x),f(f(x))))
endlambda;
```