$SPAD/src/input pdecomp0.as

The Axiom Team

December 3, 2016

Abstract

Contents

1 License

—— * ——

```
#pile
#include "axiom.as"

--% Polynomial composition and decomposition functions
--  If f = g o h then  g = leftFactor(f, h)  &  h = rightFactor(f, g)
--  SMW Dec 86

--% PolynomialComposition
--)abbrev package PCOMP PolynomialComposition
--)abbrev package PDECOMP PolynomialDecomposition

PolynomialComposition(UP: UnivariatePolynomialCategory(R), R: Ring): with
        compose: (UP, UP) -> UP
```

```
    == add
        compose(g:UP, h:UP):UP ==
            r: UP := 0
            while g ~= 0 repeat
                r := leadingCoefficient(g)*h**degree(g) + r
                g := reductum g
            r


--  Ref: Kozen and Landau, Cornell University  TR 86-773

--% PolynomialDecomposition


PolynomialDecomposition(UP:UPC F, F:Field): PDcat == PDdef where
    UPC ==> UnivariatePolynomialCategory
    NNI ==> NonNegativeInteger
    LR  ==> Record(left: UP, right: UP)

    PDcat ==> with
        decompose: UP -> List UP
        decompose: (UP, NNI, NNI) -> Union(value1:LR, failed:'failed')
        leftFactor: (UP, UP) -> Union(value1:UP, failed:'failed')
        rightFactorCandidate:  (UP, NNI) -> UP
    PDdef ==> add

        import from F
        import from LR
        import from Union(value1:UP, failed:'failed')
        import from Float
        import from NNI
        import from UniversalSegment NNI
        import from Record(quotient:UP, remainder:UP);

        leftFactor(f:UP, h:UP):Union(value1:UP, failed:'failed')  ==
             g: UP := 0
             for i in 0.. while f ~= 0 repeat
                 fr := divide(f, h)
                 f := fr.quotient
                 r := fr.remainder
                 degree r > 0 => return [failed]
                 g := g + r * monomial(1, i)
             [g]

        decompose(f:UP, dg:NNI, dh:NNI):Union(value1:LR, failed:'failed') ==
            df := degree f
            dg*dh ~= df => [failed]
            h := rightFactorCandidate(f, dh)
            g:Union(value1:UP, failed:'failed') := leftFactor(f, h)
```

```
    g case failed => [failed]
    [[g.value1, h]]

decompose(f:UP):List UP ==
    df := degree f
    for dh in 2..df-1 | df rem dh = 0 repeat
        h := rightFactorCandidate(f, dh)
        g := leftFactor(f, h)
        g case value1 => return
            append(decompose(g.value1), decompose h)
    [f]


rightFactorCandidate(f:UP, dh:NNI):UP ==
    f  := f / leadingCoefficient f
    df := degree f
    dg := df quo dh
    h  := monomial(1, dh)
    for k in 1..dh repeat
        hdg:= h**dg
        c  := (coefficient(f,df-k)-coefficient(hdg,df-k))/
                                                (dg::Integer::F)
        h  := h + monomial(c, dh-k)
    h - monomial(coefficient(h, 0), 0) -- drop constant term


    ———————-
```

References

[1] nothing