NewtonGF[Radius] - compute the radius of convergence of the generating functions of a combinatorial system

Calling Sequence

Radius(**Sys**, **labelling**)

Parameters

`Sys` – set of equations; a grammar in the combstruct syntax
`labelling` – one of labelled, labeled, unlabelled, unlabeled, as in combstruct

Description

- The **Radius** command returns the radius of convergence of the generating series (only one radius, the gf being interpreted as a series of vectors).
- In the unlabelled case, it is computed by dichotomy using **NewtonGF**[**NumericalNewtonIteration**] for the evaluation at nonnegative real values. In the labelled case, it is computed by a recursive use of Newton iteration.
- This command is part of the **NewtonGF** package, so it can be used in the form **Radius(..)** only after executing the command **with(NewtonGF)**. However, it can always be accessed through the long form of the command by using **NewtonGF**[**Radius**](..).

Examples

```
> with(NewtonGF);
```

[*BoltzmannExpectedSize*, *BoltzmannParameter*, *GFSeries*, *NumericalNewtonIteration*, *Radius*, *SeriesNewtonIteration*] **(2.1)**

A grammar for binary sequences.

```
> sys:={B = Sequence(Union(Z, Z))};
```

$$sys := \{B = Sequence(Union(Z, Z))\} \qquad (2.2)$$

Here we have explicit generating functions:
combstruct[gfeqns](sys, labeled, z);

$$\left[B(z) = \frac{1}{1-2z}, Z(z) = z\right]$$

```
> Radius(sys, labeled);
```

$$0.500000000000000 \qquad (2.3)$$

```
> Digits := 100;
```

$$Digits := 100 \qquad (2.4)$$

```
> Radius(sys, labeled);
```

0.5000000000000000000000000000000000000000000000000000000000000000000000\
000000000000000000000000000000 **(2.5)**

```
> Digits := 10;
```

$$Digits := 10 \qquad (2.6)$$

A grammar for set partitions.

```
> sys:={F = Set(Set(Z, 1 <= card))};
```

$$sys := \{F = Set(Set(Z, 1 \leq card))\} \qquad (2.7)$$

combstruct[gfeqns](sys, labeled, z);

$\left[F(z) = \mathrm{e}^{\mathrm{e}^{z}-1}, Z(z) = z\right]$

```
> Radius(sys, labeled);
```

$$\infty \qquad (2.8)$$

A grammar for ternary trees (Sloane A001764).

```
> sys:={F=Union(Epsilon,Prod(Z,T)), T = Prod(F,F,F)};
```

$$sys := \{F = Union(\mathrm{E}, Prod(Z, T)), T = Prod(F, F, F)\} \qquad (2.9)$$

In this example, the generating function do not have a very nice closed form:
combstruct[gfeqns](sys, labeled, z); combstruct[gfsolve](sys, labeled, z);

$$\left[F(z) = 1 + z\,T(z), T(z) = F(z)^3, Z(z) = z\right]$$

$$\left\{F(z) = 1 + z\,RootOf\left(1 + z^3\,\_Z^3 + 3\,z^2\,\_Z^2 + (-1 + 3\,z)\,\_Z\right), T(z) = RootOf\left(1 + z^3\,\_Z^3 + 3\,z^2\,\_Z^2 + (-1 + 3\,z)\,\_Z\right), Z(z) = z\right\}$$

```
> Radius(sys, labeled);
```

$$0.148148148148148 \qquad (2.10)$$

A grammar for series-parallel circuits.

```
> circuit:={C=Union(P,S,R), P=Set(Union(S,R),card>=2), S=Set
  (Union(P,R),card>=2), R=Atom};
```

$$circuit := \{C = Union(P, S, R), P = Set(Union(S, R), 2 \leq card), R = Atom, S = Set(Union(P, R), 2 \leq card)\} \qquad (2.11)$$

Computing the radius for labelled circuits...

```
> Radius(circuit, labeled);
```

$$0.3862943611 \qquad (2.12)$$

... and also for unlabelled circuits.

```
> Radius(circuit, unlabeled);
```

$$0.2808326670 \qquad (2.13)$$

See Also

combstruct[gfseries], NewtonGF[SeriesNewtonIteration], NewtonGF[NumericalNewtonIteration], NewtonGF