Symbolic Computation with SymPy

1. Symbolic Computing in Python
   - SymPy in Python
   - SymPy in SageMath
2. Two Examples
   - series with a generator
   - solving recurrence relations

MCS 320 Lecture 37
Introduction to Symbolic Computation
Jan Verschelde, 29 July 2022

Symbolic Computation with SymPy

1. Symbolic Computing in Python
   - SymPy in Python
   - SymPy in SageMath
2. Two Examples
   - series with a generator
   - solving recurrence relations

SymPy in Python

Symbolic computing in *pure* Python.

Home: `https://www.sympy.org`.

Lightweight: depends only on `mpmath`.

Online shell at

`https://live.sympy.org`.

Free library, under the BSD license.

Symbolic Computation with SymPy

1. Symbolic Computing in Python
   - SymPy in Python
   - SymPy in SageMath
2. Two Examples
   - series with a generator
   - solving recurrence relations

SymPy in SageMath

Computing expressions starts with symbols.

- Both SymPy and SageMath use `var()` to declare symbols.
- To use SymPy symbols in SageMath:

```
import sympy
x = sympy.var('x')
```

- SymPy symbols are needed in `sympy.series`.
- SymPy results are cast into the Symbolic Ring with `SR()`.

Symbolic Computation with SymPy

1. Symbolic Computing in Python
   - SymPy in Python
   - SymPy in SageMath

2. Two Examples
   - series with a generator
   - solving recurrence relations

Series with a Generator

The *Taylor series* of $f(x)$ at $x = a$ is

$$f(a) + f'(a)(x-a) + f''(a)\frac{1}{2}(x-a)^2 + f'''(a)\frac{1}{3!}(x-a)^3 + \cdots$$

or written with the derivative operator D, up to order n

$$\sum_{k=0}^{n-1} \frac{D^k f(a)}{k!}(x-a)^k + O((x-a)^n),$$

where

- $D^k f(a)$ is the value of the k-th derivative of $f(x)$ at $x = a$,
- $O((x-a)^n)$ is the *order* of the series.

With a generator, we compute *the next term* in the series.

Symbolic Computation with SymPy

1. Symbolic Computing in Python
   - SymPy in Python
   - SymPy in SageMath

2. Two Examples
   - series with a generator
   - solving recurrence relations

solving recurrence relations

Examples of a one term and two term recurrent relations are below.

1. The cost $T(n)$ of merge sort on a list of n numbers:

$$T(n) = 2\,T\left(\frac{n}{2}\right) + n - 1, \quad T(1) = 0.$$

2. The Fibonacci numbers, for a nonnegative integer n,

$$f(n) = f(n-1) + f(n-2), \quad f(0) = 0, \quad f(1) = 1.$$

An explicit expression in function of n can be obtained for the solution.