

# symbolicsubstitution

November 7, 2023

This file is part of CasADi.

CasADi -- A symbolic framework for dynamic optimization.  
Copyright (C) 2010–2023 Joel Andersson, Joris Gillis, Moritz Diehl,  
KU Leuven. All rights reserved.  
Copyright (C) 2011–2014 Greg Horn

CasADi is free software; you can redistribute it and/or  
modify it under the terms of the GNU Lesser General Public  
License as published by the Free Software Foundation; either  
version 3 of the License, or (at your option) any later version.

CasADi is distributed in the hope that it will be useful,  
but WITHOUT ANY WARRANTY; without even the implied warranty of  
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU  
Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public  
License along with CasADi; if not, write to the Free Software  
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

## 1 Symbolic substitution

```
[1]: from casadi import *
```

Let's build a trivial symbolic SX graph

```
[2]: x = SX.sym("x")
     y = SX.sym("y")
     z_ = x*y
     z = z_+x
     print(type(z), z)
```

```
<class 'casadi.casadi.SX'> ((x*y)+x)
```

We need SXFunction to manipulate the SX graph

```
[3]: f = Function('f', [vertcat(x,y)], [z])
```

We can substitute a leaf in the graph

```
[4]: w = SX.sym("w")
     q = f(vertpat(w,y))
```

f.eval() returns a tuple with all outputs, we selected the first

```
[5]: print(type(q), q)
```

```
<class 'casadi.casadi.SX'> ((w*y)+w)
```

Note how q is now an SX

We can take a shortcut via substitute:

```
[6]: q = substitute(z,x,w)
     print(type(q), q)
```

```
<class 'casadi.casadi.SX'> ((w*y)+w)
```

Note that substitution of non-symbolic SX nodes is not permitted: substitute([z],[z\_],[w]) This would throw an error

This is actually a restriction of Function: Function([[z\_,y]], [z]) This would throw an error