ModelingToolkit.jl, An IR and Compiler for Scientific Models

Chris Rackauckas

August 7, 2021

A lot of people are building modeling languages for their specific domains. However, while the syntax my vary greatly between these domain-specific languages (DSLs), the internals of modeling frameworks are surprisingly similar: building differential equations, calculating Jacobians, etc.

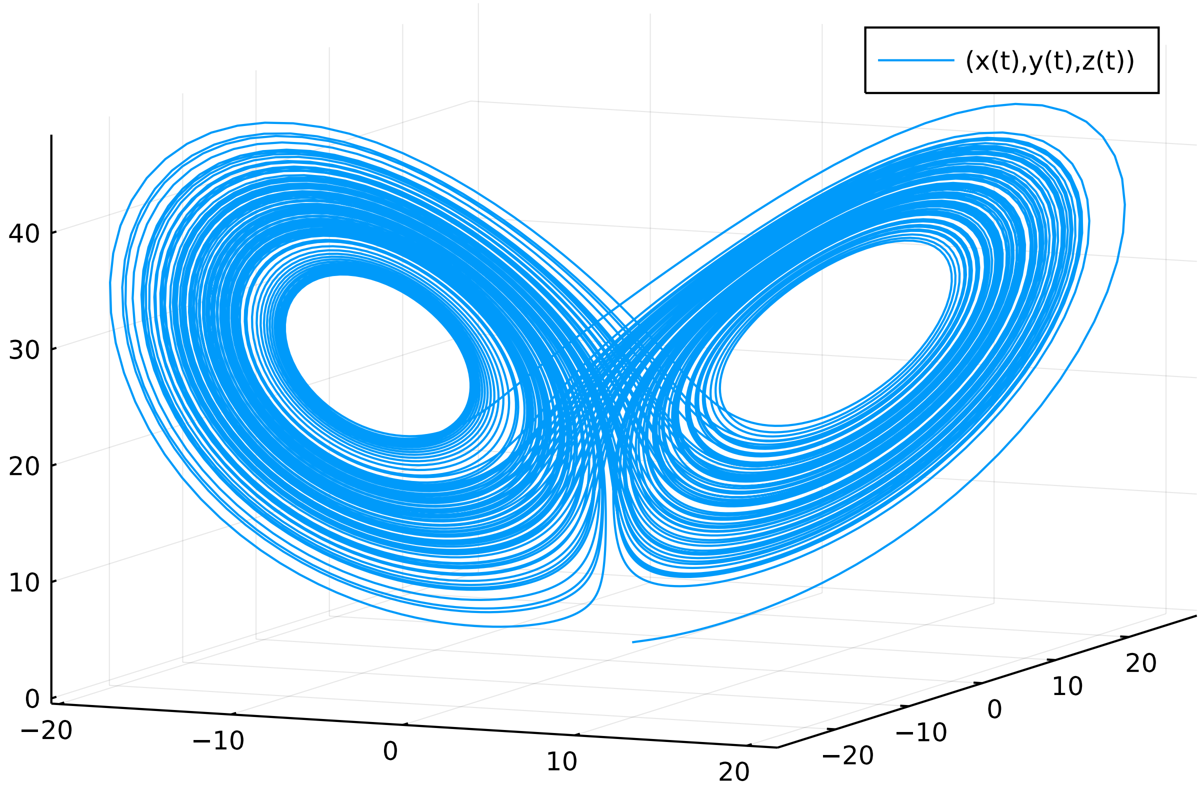
ModelingToolkit.jl is metamodeling systemitized After building our third modeling interface, we realized that this problem can be better approached by having a reusable internal structure which DSLs can target. This internal is ModelingToolkit.jl: an Intermediate Representation (IR) with a well-defined interface for defining system transformations and compiling to Julia functions for use in numerical libraries. Now a DSL can easily be written by simply defining the translation to ModelingToolkit.jl's primatives and querying for the mathematical quantities one needs.

0.0.1 Basic usage: defining differential equation systems, with performance!

Let's explore the IR itself. ModelingToolkit.jl is friendly to use, and can used as a symbolic DSL in its own right. Let's define and solve the Lorenz differential equation system using ModelingToolkit to generate the functions:

```
using ModelingToolkit

### Define a differential equation system

@parameters t σ ρ β
@variables x(t) y(t) z(t)
@derivatives D'~t

eqs = [D(x) ~ σ*(y-x),
       D(y) ~ x*(ρ-z)-y,
       D(z) ~ x*y - β*z]
de = ODESystem(eqs, t, [x,y,z], [σ,ρ,β])
ode_f = ODEFunction(de)

### Use in DifferentialEquations.jl

using OrdinaryDiffEq
u_0 = ones(3)
tspan = (0.0,100.0)
```

```
p = [10.0,28.0,10/3]
prob = ODEProblem(ode_f,u_0,tspan,p)
sol = solve(prob,Tsit5())

using Plots
plot(sol,vars=(1,2,3))
```



0.0.2 ModelingToolkit is a compiler for mathematical systems

At its core, ModelingToolkit is a compiler. It's IR is its type system, and its output are Julia functions (it's a compiler for Julia code to Julia code, written in Julia).

DifferentialEquations.jl wants a function `f(u,p,t)` or `f(du,u,p,t)` for defining an ODE system, so ModelingToolkit.jl builds both. First the out of place version:

```
generate_function(de)[1]
```

```
:(function (var"##arg#1059", var"##arg#1060", t)
      #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28aa57c6c6ea/
packages/SymbolicUtils/9iQGH/src/code.jl:
282 =#
      #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28aa57c6c6ea/
packages/SymbolicUtils/9iQGH/src/code.jl:
283 =#
      let var"x(t)" = #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-
bc97-28aa57c6c6ea/packages/SymbolicUtils/9i
QGH/src/code.jl:169 =# @inbounds(var"##arg#1059"[1]), var"y(t)" = #= /root/.cache/julia-
buildkite-plugin/depots/a6029d3a-f78b-41ea
-bc97-28aa57c6c6ea/packages/SymbolicUtils/9iQGH/src/code.jl:169 =# @inbounds(var"##arg
#1059"[2]), var"z(t)" = #= /root/.cache/juli
```

```
a-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28aa57c6c6ea/packages/SymbolicUtils/9
iQGH/src/code.jl:169 =# @inbounds(var"##arg
#1059"[3]), σ = #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28
aa57c6c6ea/packages/SymbolicUtils/9iQGH/sr
c/code.jl:169 =# @inbounds(var"##arg#1060"[1]), ρ = #= /root/.cache/julia-buildkite-
plugin/depots/a6029d3a-f78b-41ea-bc97-28aa57c6
c6ea/packages/SymbolicUtils/9iQGH/src/code.jl:169 =# @inbounds(var"##arg#1060"[2]), β =
#= /root/.cache/julia-buildkite-plugin/dep
ots/a6029d3a-f78b-41ea-bc97-28aa57c6c6ea/packages/SymbolicUtils/9iQGH/src/code.jl:169 =#
@inbounds(var"##arg#1060"[3])
          #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28
aa57c6c6ea/packages/SymbolicUtils/9iQGH/src/code
.jl:375 =#
          (SymbolicUtils.Code.create_array)(typeof(var"##arg#1059"), nothing, Val{1}(),
Val{(3,)}(), (*)(σ, (+)(var"y(t)", (*)(-1,
 var"x(t)"))), (+)((*)(var"x(t)", (+)(ρ, (*)(-1, var"z(t)"))), (*)(-1, var"y(t)")), (+)
((*)(var"x(t)", var"y(t)"), (*)(-1, β, var"
z(t)")))
      end
  end)
```

and the in-place:

```julia
generate_function(de)[2]
```

```
:(function (var"##out#1064", var"##arg#1062", var"##arg#1063", t)
      #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28aa57c6c6ea/
packages/SymbolicUtils/9iQGH/src/code.jl:
282 =#
      #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28aa57c6c6ea/
packages/SymbolicUtils/9iQGH/src/code.jl:
283 =#
      let var"x(t)" = #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-
bc97-28aa57c6c6ea/packages/SymbolicUtils/9i
QGH/src/code.jl:169 =# @inbounds(var"##arg#1062"[1]), var"y(t)" = #= /root/.cache/julia-
buildkite-plugin/depots/a6029d3a-f78b-41ea
-bc97-28aa57c6c6ea/packages/SymbolicUtils/9iQGH/src/code.jl:169 =# @inbounds(var"##arg
#1062"[2]), var"z(t)" = #= /root/.cache/juli
a-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28aa57c6c6ea/packages/SymbolicUtils/9
iQGH/src/code.jl:169 =# @inbounds(var"##arg
#1062"[3]), σ = #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28
aa57c6c6ea/packages/SymbolicUtils/9iQGH/sr
c/code.jl:169 =# @inbounds(var"##arg#1063"[1]), ρ = #= /root/.cache/julia-buildkite-
plugin/depots/a6029d3a-f78b-41ea-bc97-28aa57c6
c6ea/packages/SymbolicUtils/9iQGH/src/code.jl:169 =# @inbounds(var"##arg#1063"[2]), β =
#= /root/.cache/julia-buildkite-plugin/dep
ots/a6029d3a-f78b-41ea-bc97-28aa57c6c6ea/packages/SymbolicUtils/9iQGH/src/code.jl:169 =#
@inbounds(var"##arg#1063"[3])
          #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28
aa57c6c6ea/packages/Symbolics/h8kPL/src/build_fu
nction.jl:331 =#
          #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28
aa57c6c6ea/packages/SymbolicUtils/9iQGH/src/code
.jl:329 =# @inbounds begin
                  #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97
-28aa57c6c6ea/packages/SymbolicUtils/9iQGH/
src/code.jl:325 =#
                  var"##out#1064"[1] = (*)(σ, (+)(var"y(t)", (*)(-1, var"x(t)")))
                  var"##out#1064"[2] = (+)((*)(var"x(t)", (+)(ρ, (*)(-1, var"z(t)"))),
(*)(-1, var"y(t)"))
```

```
                  var"##out#1064"[3] = (+)((*)(var"x(t)", var"y(t)"), (*)(-1, β, var"z(t)
"))
                  #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97
-28aa57c6c6ea/packages/SymbolicUtils/9iQGH/
src/code.jl:327 =#
                  nothing
              end
      end
  end)
```

ModelingToolkit.jl can be used to calculate the Jacobian of the differential equation system:

```
jac = calculate_jacobian(de)
```

$$\left[\begin{array}{ccc} -\sigma & \sigma & 0 \\ \rho - z\left(t\right) & -1 & -x\left(t\right) \\ y\left(t\right) & x\left(t\right) & -\beta \end{array}\right] \tag{1}$$

It will automatically generate functions for using this Jacobian within the stiff ODE solvers for faster solving:

```
jac_expr = generate_jacobian(de)
```

```
(:(function (var"##arg#1066", var"##arg#1067", t)
      #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28aa57c6c6ea/
packages/SymbolicUtils/9iQGH/src/code.jl:
282 =#
      #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28aa57c6c6ea/
packages/SymbolicUtils/9iQGH/src/code.jl:
283 =#
      let var"x(t)" = #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-
bc97-28aa57c6c6ea/packages/SymbolicUtils/9i
QGH/src/code.jl:169 =# @inbounds(var"##arg#1066"[1]), var"y(t)" = #= /root/.cache/julia-
buildkite-plugin/depots/a6029d3a-f78b-41ea
-bc97-28aa57c6c6ea/packages/SymbolicUtils/9iQGH/src/code.jl:169 =# @inbounds(var"##arg
#1066"[2]), var"z(t)" = #= /root/.cache/juli
a-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28aa57c6c6ea/packages/SymbolicUtils/9
iQGH/src/code.jl:169 =# @inbounds(var"##arg
#1066"[3]), σ = #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28
aa57c6c6ea/packages/SymbolicUtils/9iQGH/sr
c/code.jl:169 =# @inbounds(var"##arg#1067"[1]), ρ = #= /root/.cache/julia-buildkite-
plugin/depots/a6029d3a-f78b-41ea-bc97-28aa57c6
c6ea/packages/SymbolicUtils/9iQGH/src/code.jl:169 =# @inbounds(var"##arg#1067"[2]), β =
#= /root/.cache/julia-buildkite-plugin/dep
ots/a6029d3a-f78b-41ea-bc97-28aa57c6c6ea/packages/SymbolicUtils/9iQGH/src/code.jl:169 =#
@inbounds(var"##arg#1067"[3])
          #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28
aa57c6c6ea/packages/SymbolicUtils/9iQGH/src/code
.jl:375 =#
          (SymbolicUtils.Code.create_array)(typeof(var"##arg#1066"), nothing, Val{2}(),
Val{(3, 3)}(), (*)(-1, σ), (+)(ρ, (*)(-1,
var"z(t)")), var"y(t)", σ, -1, var"x(t)", 0, (*)(-1, var"x(t)"), (*)(-1, β))
      end
  end), :(function (var"##out#1068", var"##arg#1066", var"##arg#1067", t)
      #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28aa57c6c6ea/
packages/SymbolicUtils/9iQGH/src/code.jl:
282 =#
      #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28aa57c6c6ea/
packages/SymbolicUtils/9iQGH/src/code.jl:
```

```
283 =#
      let var"x(t)" = #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-
bc97-28aa57c6c6ea/packages/SymbolicUtils/9i
QGH/src/code.jl:169 =# @inbounds(var"##arg#1066"[1]), var"y(t)" = #= /root/.cache/julia-
buildkite-plugin/depots/a6029d3a-f78b-41ea
-bc97-28aa57c6c6ea/packages/SymbolicUtils/9iQGH/src/code.jl:169 =# @inbounds(var"##arg
#1066"[2]), var"z(t)" = #= /root/.cache/juli
a-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28aa57c6c6ea/packages/SymbolicUtils/9
iQGH/src/code.jl:169 =# @inbounds(var"##arg
#1066"[3]), σ = #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28
aa57c6c6ea/packages/SymbolicUtils/9iQGH/sr
c/code.jl:169 =# @inbounds(var"##arg#1067"[1]), ρ = #= /root/.cache/julia-buildkite-
plugin/depots/a6029d3a-f78b-41ea-bc97-28aa57c6
c6ea/packages/SymbolicUtils/9iQGH/src/code.jl:169 =# @inbounds(var"##arg#1067"[2]), β =
#= /root/.cache/julia-buildkite-plugin/dep
ots/a6029d3a-f78b-41ea-bc97-28aa57c6c6ea/packages/SymbolicUtils/9iQGH/src/code.jl:169 =#
@inbounds(var"##arg#1067"[3])
          #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28
aa57c6c6ea/packages/Symbolics/h8kPL/src/build_fu
nction.jl:331 =#
          #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28
aa57c6c6ea/packages/SymbolicUtils/9iQGH/src/code
.jl:329 =# @inbounds begin
                  #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97
-28aa57c6c6ea/packages/SymbolicUtils/9iQGH/
src/code.jl:325 =#
                  var"##out#1068"[1] = (*)(-1, σ)
                  var"##out#1068"[2] = (+)(ρ, (*)(-1, var"z(t)"))
                  var"##out#1068"[3] = var"y(t)"
                  var"##out#1068"[4] = σ
                  var"##out#1068"[5] = -1
                  var"##out#1068"[6] = var"x(t)"
                  var"##out#1068"[7] = 0
                  var"##out#1068"[8] = (*)(-1, var"x(t)")
                  var"##out#1068"[9] = (*)(-1, β)
                  #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97
-28aa57c6c6ea/packages/SymbolicUtils/9iQGH/
src/code.jl:327 =#
                  nothing
              end
      end
  end))
```

It can even do fancy linear algebra. Stiff ODE solvers need to perform an LU-factorization which is their most expensive part. But ModelingToolkit.jl can skip this operation and instead generate the analytical solution to a matrix factorization, and build a Julia function for directly computing the factorization, which is then optimized in LLVM compiler passes.

```
ModelingToolkit.generate_factorized_W(de)[1]
```

```
Error: MethodError: no method matching generate_factorized_W(::ModelingToolkit.ODESystem)
```

0.0.3 Solving Nonlinear systems

ModelingToolkit.jl is not just for differential equations. It can be used for any mathematical target that is representable by its IR. For example, let's solve a rootfinding problem `F(x)=0`. What we do is define a nonlinear system and generate a function for use in NLsolve.jl

```julia
@variables x y z
@parameters σ ρ β

# Define a nonlinear system
eqs = [0 ~ σ*(y-x),
       0 ~ x*(ρ-z)-y,
       0 ~ x*y - β*z]
ns = NonlinearSystem(eqs, [x,y,z], [σ,ρ,β])
nlsys_func = generate_function(ns)
```

```
(:(function (var"##arg#1070", var"##arg#1071")
      #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28aa57c6c6ea/
packages/SymbolicUtils/9iQGH/src/code.jl:
282 =#
      #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28aa57c6c6ea/
packages/SymbolicUtils/9iQGH/src/code.jl:
283 =#
      let x = #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28
aa57c6c6ea/packages/SymbolicUtils/9iQGH/src/
code.jl:169 =# @inbounds(var"##arg#1070"[1]), y = #= /root/.cache/julia-buildkite-plugin/
depots/a6029d3a-f78b-41ea-bc97-28aa57c6c6
ea/packages/SymbolicUtils/9iQGH/src/code.jl:169 =# @inbounds(var"##arg#1070"[2]), z = #=
/root/.cache/julia-buildkite-plugin/depot
s/a6029d3a-f78b-41ea-bc97-28aa57c6c6ea/packages/SymbolicUtils/9iQGH/src/code.jl:169 =#
@inbounds(var"##arg#1070"[3]), σ = #= /root
/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28aa57c6c6ea/packages/
SymbolicUtils/9iQGH/src/code.jl:169 =# @inboun
ds(var"##arg#1071"[1]), ρ = #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b
-41ea-bc97-28aa57c6c6ea/packages/SymbolicUt
ils/9iQGH/src/code.jl:169 =# @inbounds(var"##arg#1071"[2]), β = #= /root/.cache/julia-
buildkite-plugin/depots/a6029d3a-f78b-41ea-b
c97-28aa57c6c6ea/packages/SymbolicUtils/9iQGH/src/code.jl:169 =# @inbounds(var"##arg#1071
"[3])
          #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28
aa57c6c6ea/packages/SymbolicUtils/9iQGH/src/code
.jl:375 =#
          (SymbolicUtils.Code.create_array)(typeof(var"##arg#1070"), nothing, Val{1}(),
Val{(3,)}(), (*)(σ, (+)(y, (*)(-1, x))), (
+)((*)(x, (+)(ρ, (*)(-1, z))), (*)(-1, y)), (+)((*)(x, y), (*)(-1, z, β)))
      end
  end), :(function (var"##out#1072", var"##arg#1070", var"##arg#1071")
      #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28aa57c6c6ea/
packages/SymbolicUtils/9iQGH/src/code.jl:
282 =#
      #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28aa57c6c6ea/
packages/SymbolicUtils/9iQGH/src/code.jl:
283 =#
      let x = #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28
aa57c6c6ea/packages/SymbolicUtils/9iQGH/src/
code.jl:169 =# @inbounds(var"##arg#1070"[1]), y = #= /root/.cache/julia-buildkite-plugin/
depots/a6029d3a-f78b-41ea-bc97-28aa57c6c6
ea/packages/SymbolicUtils/9iQGH/src/code.jl:169 =# @inbounds(var"##arg#1070"[2]), z = #=
/root/.cache/julia-buildkite-plugin/depot
s/a6029d3a-f78b-41ea-bc97-28aa57c6c6ea/packages/SymbolicUtils/9iQGH/src/code.jl:169 =#
@inbounds(var"##arg#1070"[3]), σ = #= /root
/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28aa57c6c6ea/packages/
SymbolicUtils/9iQGH/src/code.jl:169 =# @inboun
ds(var"##arg#1071"[1]), ρ = #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b
-41ea-bc97-28aa57c6c6ea/packages/SymbolicUt
ils/9iQGH/src/code.jl:169 =# @inbounds(var"##arg#1071"[2]), β = #= /root/.cache/julia-
```

```
buildkite-plugin/depots/a6029d3a-f78b-41ea-b
c97-28aa57c6c6ea/packages/SymbolicUtils/9iQGH/src/code.jl:169 =# @inbounds(var"##arg#1071
"[3])
          #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28
aa57c6c6ea/packages/Symbolics/h8kPL/src/build_fu
nction.jl:331 =#
          #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97-28
aa57c6c6ea/packages/SymbolicUtils/9iQGH/src/code
.jl:329 =# @inbounds begin
                  #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97
-28aa57c6c6ea/packages/SymbolicUtils/9iQGH/
src/code.jl:325 =#
                  var"##out#1072"[1] = (*)(σ, (+)(y, (*)(-1, x)))
                  var"##out#1072"[2] = (+)((*)(x, (+)(ρ, (*)(-1, z))), (*)(-1, y))
                  var"##out#1072"[3] = (+)((*)(x, y), (*)(-1, z, β))
                  #= /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc97
-28aa57c6c6ea/packages/SymbolicUtils/9iQGH/
src/code.jl:327 =#
                  nothing
              end
      end
  end))
```

We can then tell ModelingToolkit.jl to compile this function for use in NLsolve.jl, and then numerically solve the rootfinding problem:

```
nl_f = @eval eval(nlsys_func[2])
# Make a closure over the parameters for for NLsolve.jl
f2 = (du,u) -> nl_f(du,u,(10.0,26.0,2.33))

using NLsolve
nlsolve(f2,ones(3))
```

```
Results of Nonlinear Solver Algorithm
 * Algorithm: Trust-region with dogleg and autoscaling
 * Starting Point: [1.0, 1.0, 1.0]
 * Zero: [2.2228042243306243e-10, 2.2228042243645056e-10, -9.990339599422887e-11]
 * Inf-norm of residuals: 0.000000
 * Iterations: 3
 * Convergence: true
   * |x - x'| < 0.0e+00: false
   * |f(x)| < 1.0e-08: true
 * Function Calls (f): 4
 * Jacobian Calls (df/dx): 4
```

0.0.4 Library of transformations on mathematical systems

The reason for using ModelingToolkit is not just for defining performant Julia functions for solving systems, but also for performing mathematical transformations which may be required in order to numerically solve the system. For example, let's solve a third order ODE. The way this is done is by transforming the third order ODE into a first order ODE, and then solving the resulting ODE. This transformation is given by the `ode_order_lowering` function.

```
@derivatives D3'''~t
@derivatives D2''~t
@variables u(t), x(t)
```

```
eqs = [D3(u) ~ 2(D2(u)) + D(u) + D(x) + 1
       D2(x) ~ D(x) + 2]
de = ODESystem(eqs, t, [u,x], [])
de1 = ode_order_lowering(de)
```

$$\frac{dutt(t)}{dt} = 1 + \mathrm{ut}\left(t\right) + \mathrm{xt}\left(t\right) + 2\mathrm{utt}\left(t\right) \tag{2}$$
$$\frac{dxt(t)}{dt} = 2 + \mathrm{xt}\left(t\right) \tag{3}$$
$$\frac{dut(t)}{dt} = \mathrm{utt}\left(t\right) \tag{4}$$
$$\frac{du(t)}{dt} = \mathrm{ut}\left(t\right) \tag{5}$$
$$\frac{dx(t)}{dt} = \mathrm{xt}\left(t\right) \tag{6}$$

```
de1.eqs
```

$$\frac{dutt(t)}{dt} = 1 + \mathrm{ut}\left(t\right) + \mathrm{xt}\left(t\right) + 2\mathrm{utt}\left(t\right) \tag{7}$$
$$\frac{dxt(t)}{dt} = 2 + \mathrm{xt}\left(t\right) \tag{8}$$
$$\frac{dut(t)}{dt} = \mathrm{utt}\left(t\right) \tag{9}$$
$$\frac{du(t)}{dt} = \mathrm{ut}\left(t\right) \tag{10}$$
$$\frac{dx(t)}{dt} = \mathrm{xt}\left(t\right) \tag{11}$$

This has generated a system of 5 first order ODE systems which can now be used in the ODE solvers.

0.0.5 Linear Algebra... for free?

Let's take a look at how to extend ModelingToolkit.jl in new directions. Let's define a Jacobian just by using the derivative primatives by hand:

```
@parameters t σ ρ β
@variables x(t) y(t) z(t)
@derivatives D'~t Dx'~x Dy'~y Dz'~z
eqs = [D(x) ~ σ*(y-x),
       D(y) ~ x*(ρ-z)-y,
       D(z) ~ x*y - β*z]
J = [Dx(eqs[1].rhs) Dy(eqs[1].rhs) Dz(eqs[1].rhs)
 Dx(eqs[2].rhs) Dy(eqs[2].rhs) Dz(eqs[2].rhs)
 Dx(eqs[3].rhs) Dy(eqs[3].rhs) Dz(eqs[3].rhs)]
```

```
3×3 Matrix{SymbolicUtils.Term{Real, Nothing}}:
 Differential(x(t))(σ*(y(t) - x(t)))         …  Differential(z(t))(σ*(y(t) - x(t)))
 Differential(x(t))(x(t)*(ρ - z(t)) - y(t))     Differential(z(t))(x(t)*(ρ - z(t)) - y(t)
)
 Differential(x(t))(x(t)*y(t) - (β*z(t)))       Differential(z(t))(x(t)*y(t) - (β*z(t)))
```

Notice that this writes the derivatives in a "lazy" manner. If we want to actually compute the derivatives, we can expand out those expressions:

```
J = expand_derivatives.(J)
```

```
3×3 Matrix{Any}:
 -σ          σ     0
 ρ - z(t)  -1      -x(t)
 y(t)        x(t)  -β
```

Here's the magic of ModelingToolkit.jl: **Julia treats ModelingToolkit expressions like a Number, and so generic numerical functions are directly usable on ModelingToolkit expressions!** Let's compute the LU-factorization of this Jacobian we defined using Julia's Base linear algebra library.

```
using LinearAlgebra
luJ = lu(J,Val(false))
```

```
Error: MethodError: no method matching oneunit(::Type{Any})
Closest candidates are:
  oneunit(::Type{Union{Missing, T}}) where T at missing.jl:105
  oneunit(::Type{T}) where T at number.jl:319
  oneunit(::T) where T at number.jl:318
  ...
```

```
luJ.L
```

```
Error: UndefVarError: luJ not defined
```

and the inverse?

```
invJ = inv(luJ)
```

```
Error: UndefVarError: luJ not defined
```

Thus ModelingToolkit.jl can utilize existing numerical code on symbolic codes Let's follow this thread a little deeper.

0.0.6 Automatically convert numerical codes to symbolic

Let's take someone's code written to numerically solve the Lorenz equation:

```
function lorenz(du,u,p,t)
 du[1] = p[1]*(u[2]-u[1])
 du[2] = u[1]*(p[2]-u[3]) - u[2]
 du[3] = u[1]*u[2] - p[3]*u[3]
end
```

```
lorenz (generic function with 1 method)
```

Since ModelingToolkit can trace generic numerical functions in Julia, let's trace it with Operations. When we do this, it'll spit out a symbolic representation of their numerical code:

```
u = [x,y,z]
du = similar(u)
p = [σ,ρ,β]
lorenz(du,u,p,t)
du
```

$$\begin{bmatrix} \sigma\left(y\left(t\right)-x\left(t\right)\right) \\ x\left(t\right)\left(\rho-z\left(t\right)\right)-y\left(t\right) \\ x\left(t\right)y\left(t\right)-\beta z\left(t\right) \end{bmatrix} \tag{12}$$

We can then perform symbolic manipulations on their numerical code, and build a new numerical code that optimizes/fixes their original function!

```
J = [Dx(du[1]) Dy(du[1]) Dz(du[1])
     Dx(du[2]) Dy(du[2]) Dz(du[2])
     Dx(du[3]) Dy(du[3]) Dz(du[3])]
J = expand_derivatives.(J)
```

$$\begin{bmatrix} -\sigma & \sigma & 0 \\ \rho-z\left(t\right) & -1 & -x\left(t\right) \\ y\left(t\right) & x\left(t\right) & -\beta \end{bmatrix} \tag{13}$$

0.0.7 Automated Sparsity Detection

In many cases one has to speed up large modeling frameworks by taking into account sparsity. While ModelingToolkit.jl can be used to compute Jacobians, we can write a standard Julia function in order to get a spase matrix of expressions which automatically detects and utilizes the sparsity of their function.

```
using SparseArrays
function SparseArrays.SparseMatrixCSC(M::Matrix{T}) where {T<:ModelingToolkit.Expression}
    idxs = findall(!iszero, M)
    I = [i[1] for i in idxs]
    J = [i[2] for i in idxs]
    V = [M[i] for i in idxs]
    return SparseArrays.sparse(I, J, V, size(M)...)
end
sJ = SparseMatrixCSC(J)
```

```
Error: UndefVarError: Expression not defined
```

0.0.8 Dependent Variables, Functions, Chain Rule

"Variables" are overloaded. When you are solving a differential equation, the variable `u(t)` is actually a function of time. In order to handle these kinds of variables in a mathematically correct and extensible manner, the ModelingToolkit IR actually treats variables as functions, and constant variables are simply 0-ary functions (`t()`).

We can utilize this idea to have parameters that are also functions. For example, we can have a parameter σ which acts as a function of 1 argument, and then utilize this function within our differential equations:

```
@parameters σ(..)
eqs = [D(x) ~ σ(t-1)*(y-x),
       D(y) ~ x*(σ(t^2)-z)-y,
       D(z) ~ x*y - β*z]
```

$$\frac{dx(t)}{dt} = \sigma\left(-1+t\right)\left(y\left(t\right)-x\left(t\right)\right) \tag{14}$$
$$\frac{dy(t)}{dt} = x\left(t\right)\left(\sigma\left(t^2\right)-z\left(t\right)\right)-y\left(t\right) \tag{15}$$
$$\frac{dz(t)}{dt} = x\left(t\right)y\left(t\right)-\beta z\left(t\right) \tag{16}$$

Notice that when we calculate the derivative with respect to `t`, the chain rule is automatically handled:

```
@derivatives D_t'~t
D_t(x*(σ(t^2)-z)-y)
expand_derivatives(D_t(x*(σ(t^2)-z)-y))
```

$$x\left(t\right)\left(\frac{d\sigma(t^2)}{dt^2}-\frac{dz(t)}{dt}\right)+\frac{dx(t)}{dt}\left(\sigma\left(t^2\right)-z\left(t\right)\right)-\frac{dy(t)}{dt} \tag{17}$$

0.0.9 Hackability: Extend directly from the language

ModelingToolkit.jl is written in Julia, and thus it can be directly extended from Julia itself. Let's define a normal Julia function and call it with a variable:

```
_f(x) = 2x + x^2
_f(x)
```

$$\left(x\left(t\right)\right)^2+2x\left(t\right) \tag{18}$$

Recall that when we do that, it will automatically trace this function and then build a symbolic expression. But what if we wanted our function to be a primative in the symbolic framework? This can be done by registering the function.

```
f(x) = 2x + x^2
@register f(x)
```

Now this function is a new primitive:

```
f(x)
```

$$f\left(x\left(t\right)\right) \tag{19}$$

and we can now define derivatives of our function:

```
function ModelingToolkit.derivative(::typeof(f), args::NTuple{1,Any}, ::Val{1})
    2 + 2args[1]
end
expand_derivatives(Dx(f(x)))
```

$$2+2x\left(t\right) \tag{20}$$

0.1 Appendix

These tutorials are a part of the SciMLTutorials.jl repository, found at: https://github.com/SciML/SciML
For more information on high-performance scientific machine learning, check out the SciML Open Source Software Organization https://sciml.ai.

To locally run this tutorial, do the following commands:

```
using SciMLTutorials
SciMLTutorials.weave_file("tutorials/ode_extras","01-ModelingToolkit.jmd")
```

Computer Information:

```
Julia Version 1.6.2
Commit 1b93d53fc4 (2021-07-14 15:36 UTC)
Platform Info:
  OS: Linux (x86_64-pc-linux-gnu)
  CPU: AMD EPYC 7502 32-Core Processor
  WORD_SIZE: 64
  LIBM: libopenlibm
  LLVM: libLLVM-11.0.1 (ORCJIT, znver2)
Environment:
  JULIA_DEPOT_PATH = /root/.cache/julia-buildkite-plugin/depots/a6029d3a-f78b-41ea-bc9
  JULIA_NUM_THREADS = 16
```

Package Information:

```
      Status `/var/lib/buildkite-agent/builds/5-amdci4-julia-csail-mit-edu/julialang/sc
  [f3b72e0c] DiffEqDevTools v2.27.2
  [0c46a032] DifferentialEquations v6.17.1
  [961ee093] ModelingToolkit v5.17.3
  [76087f3c] NLopt v0.6.2
  [2774e3e8] NLsolve v4.5.1
  [429524aa] Optim v1.3.0
  [1dea7af3] OrdinaryDiffEq v5.56.0
  [91a5bcdd] Plots v1.15.2
  [30cb0354] SciMLTutorials v0.9.0
  [37e2e46d] LinearAlgebra
  [2f01184e] SparseArrays
```

And the full manifest:

```
      Status `/var/lib/buildkite-agent/builds/5-amdci4-julia-csail-mit-edu/julialang/sc
  [c3fe647b] AbstractAlgebra v0.16.0
  [1520ce14] AbstractTrees v0.3.4
  [79e6a3ab] Adapt v3.3.0
  [ec485272] ArnoldiMethod v0.1.0
```

```
[4fba245c] ArrayInterface v3.1.15
[4c555306] ArrayLayouts v0.7.0
[aae01518] BandedMatrices v0.16.9
[6e4b80f9] BenchmarkTools v1.0.0
[764a87c0] BoundaryValueDiffEq v2.7.1
[fa961155] CEnum v0.4.1
[00ebfdb7] CSTParser v2.5.0
[d360d2e6] ChainRulesCore v0.9.44
[b630d9fa] CheapThreads v0.2.5
[523fee87] CodecBzip2 v0.7.2
[944b1d66] CodecZlib v0.7.0
[35d6a980] ColorSchemes v3.12.1
[3da002f7] ColorTypes v0.11.0
[5ae59095] Colors v0.12.8
[861a8166] Combinatorics v1.0.2
[a80b9123] CommonMark v0.8.1
[38540f10] CommonSolve v0.2.0
[bbf7d656] CommonSubexpressions v0.3.0
[34da2185] Compat v3.30.0
[8f4d0f93] Conda v1.5.2
[187b0558] ConstructionBase v1.2.1
[d38c429a] Contour v0.5.7
[a8cc5b0e] Crayons v4.0.4
[9a962f9c] DataAPI v1.6.0
[864edb3b] DataStructures v0.18.9
[e2d170a0] DataValueInterfaces v1.0.0
[bcd4f6db] DelayDiffEq v5.31.0
[2b5f629d] DiffEqBase v6.62.2
[459566f4] DiffEqCallbacks v2.16.1
[f3b72e0c] DiffEqDevTools v2.27.2
[5a0ffddc] DiffEqFinancial v2.4.0
[c894b116] DiffEqJump v6.14.2
[77a26b50] DiffEqNoiseProcess v5.7.3
[055956cb] DiffEqPhysics v3.9.0
[163ba53b] DiffResults v1.0.3
[b552c78f] DiffRules v1.0.2
[0c46a032] DifferentialEquations v6.17.1
[c619ae07] DimensionalPlotRecipes v1.2.0
[b4f34e82] Distances v0.10.3
[31c24e10] Distributions v0.24.18
[ffbed154] DocStringExtensions v0.8.4
[e30172f5] Documenter v0.26.3
[d4d017d3] ExponentialUtilities v1.8.4
[e2ba6199] ExprTools v0.1.3
[c87230d0] FFMPEG v0.4.0
[7034ab61] FastBroadcast v0.1.8
[9aa1b823] FastClosures v0.3.2
[1a297f60] FillArrays v0.11.7
[6a86dc24] FiniteDiff v2.8.0
```

```
[53c48c17] FixedPointNumbers v0.8.4
[59287772] Formatting v0.4.2
[f6369f11] ForwardDiff v0.10.18
[069b7b12] FunctionWrappers v1.1.2
[28b8d3ca] GR v0.57.4
[5c1252a2] GeometryBasics v0.3.12
[42e2da0e] Grisu v1.0.2
[cd3eb016] HTTP v0.9.9
[eafb193a] Highlights v0.4.5
[0e44f5e4] Hwloc v2.0.0
[7073ff75] IJulia v1.23.2
[b5f81e59] IOCapture v0.1.1
[615f187c] IfElse v0.1.0
[d25df0c9] Inflate v0.1.2
[83e8ac13] IniFile v0.5.0
[c8e1da08] IterTools v1.3.0
[42fd0dbc] IterativeSolvers v0.9.1
[82899510] IteratorInterfaceExtensions v1.0.0
[692b3bcd] JLLWrappers v1.3.0
[682c06a0] JSON v0.21.1
[7d188eb4] JSONSchema v0.3.3
[98e50ef6] JuliaFormatter v0.13.7
[b964fa9f] LaTeXStrings v1.2.1
[2ee39098] LabelledArrays v1.6.1
[23fbe1c1] Latexify v0.15.5
[093fc24a] LightGraphs v1.3.5
[d3d80556] LineSearches v7.1.1
[2ab3a3ac] LogExpFunctions v0.2.4
[bdcacae8] LoopVectorization v0.12.23
[1914dd2f] MacroTools v0.5.6
[b8f27783] MathOptInterface v0.9.22
[fdba3010] MathProgBase v0.7.8
[739be429] MbedTLS v1.0.3
[442fdcdd] Measures v0.3.1
[e1d29d7a] Missings v1.0.0
[961ee093] ModelingToolkit v5.17.3
[46d2c3a1] MuladdMacro v0.2.2
[f9640e96] MultiScaleArrays v1.8.1
[ffc61752] Mustache v1.0.10
[d8a4904e] MutableArithmetics v0.2.19
[d41bc354] NLSolversBase v7.8.0
[76087f3c] NLopt v0.6.2
[2774e3e8] NLsolve v4.5.1
[77ba4419] NaNMath v0.3.5
[8913a72c] NonlinearSolve v0.3.8
[6fe1bfb0] OffsetArrays v1.9.0
[429524aa] Optim v1.3.0
[bac558e1] OrderedCollections v1.4.1
[1dea7af3] OrdinaryDiffEq v5.56.0
```

```
[90014a1f] PDMats v0.11.0
[65888b18] ParameterizedFunctions v5.10.0
[d96e819e] Parameters v0.12.2
[69de0a69] Parsers v1.1.0
[ccf2f8ad] PlotThemes v2.0.1
[995b91a9] PlotUtils v1.0.10
[91a5bcdd] Plots v1.15.2
[e409e4f3] PoissonRandom v0.4.0
[f517fe37] Polyester v0.3.1
[85a6dd25] PositiveFactorizations v0.2.4
[21216c6a] Preferences v1.2.2
[1fd47b50] QuadGK v2.4.1
[74087812] Random123 v1.3.1
[fb686558] RandomExtensions v0.4.3
[e6cf234a] RandomNumbers v1.4.0
[3cdcf5f2] RecipesBase v1.1.1
[01d81517] RecipesPipeline v0.3.2
[731186ca] RecursiveArrayTools v2.11.4
[f2c3362d] RecursiveFactorization v0.1.12
[189a3867] Reexport v1.0.0
[ae029012] Requires v1.1.3
[ae5879a3] ResettableStacks v1.1.0
[79098fc4] Rmath v0.7.0
[47965b36] RootedTrees v1.0.0
[7e49a35a] RuntimeGeneratedFunctions v0.5.2
[476501e8] SLEEFPirates v0.6.20
[1bc83da4] SafeTestsets v0.0.1
[0bca4576] SciMLBase v1.13.4
[30cb0354] SciMLTutorials v0.9.0
[6c6a2e73] Scratch v1.0.3
[efcf1570] Setfield v0.7.0
[992d4aef] Showoff v1.0.3
[699a6c99] SimpleTraits v0.9.3
[b85f4697] SoftGlobalScope v1.1.0
[a2af1166] SortingAlgorithms v1.0.0
[47a9eef4] SparseDiffTools v1.13.2
[276daf66] SpecialFunctions v1.4.1
[aedffcd0] Static v0.2.4
[90137ffa] StaticArrays v1.2.0
[82ae8749] StatsAPI v1.0.0
[2913bbd2] StatsBase v0.33.8
[4c63d2b9] StatsFuns v0.9.8
[9672c7b4] SteadyStateDiffEq v1.6.2
[789caeaf] StochasticDiffEq v6.34.1
[7792a7ef] StrideArraysCore v0.1.11
[09ab397b] StructArrays v0.5.1
[c3572dad] Sundials v4.4.3
[d1185830] SymbolicUtils v0.11.2
[0c5d862f] Symbolics v0.1.25
```

```
[3783bdb8] TableTraits v1.0.1
[bd369af6] Tables v1.4.2
[8290d209] ThreadingUtilities v0.4.4
[a759f4b9] TimerOutputs v0.5.9
[0796e94c] Tokenize v0.5.16
[3bb67fe8] TranscodingStreams v0.9.5
[a2a6695c] TreeViews v0.3.0
[5c2747f8] URIs v1.3.0
[3a884ed6] UnPack v1.0.2
[1986cc42] Unitful v1.7.0
[3d5dd08c] VectorizationBase v0.20.11
[81def892] VersionParsing v1.2.0
[19fa3120] VertexSafeGraphs v0.1.2
[44d3d7a6] Weave v0.10.8
[ddb6d928] YAML v0.4.6
[c2297ded] ZMQ v1.2.1
[a5390f91] ZipFile v0.9.3
[700de1a5] ZygoteRules v0.2.1
[6e34b625] Bzip2_jll v1.0.6+5
[83423d85] Cairo_jll v1.16.0+6
[5ae413db] EarCut_jll v2.1.5+1
[2e619515] Expat_jll v2.2.10+0
[b22a6f82] FFMPEG_jll v4.3.1+4
[a3f928ae] Fontconfig_jll v2.13.1+14
[d7e528f0] FreeType2_jll v2.10.1+5
[559328eb] FriBidi_jll v1.0.5+6
[0656b61e] GLFW_jll v3.3.4+0
[d2c73de3] GR_jll v0.57.2+0
[78b55507] Gettext_jll v0.21.0+0
[7746bdde] Glib_jll v2.68.1+0
[e33a78d0] Hwloc_jll v2.4.1+0
[aacddb02] JpegTurbo_jll v2.0.1+3
[c1c5ebd0] LAME_jll v3.100.0+3
[dd4b983a] LZO_jll v2.10.1+0
[dd192d2f] LibVPX_jll v1.9.0+1
[e9f186c6] Libffi_jll v3.2.2+0
[d4300ac3] Libgcrypt_jll v1.8.7+0
[7e76a0d4] Libglvnd_jll v1.3.0+3
[7add5ba3] Libgpg_error_jll v1.42.0+0
[94ce4f54] Libiconv_jll v1.16.1+0
[4b2f31a3] Libmount_jll v2.35.0+0
[89763e89] Libtiff_jll v4.1.0+2
[38a345b3] Libuuid_jll v2.36.0+0
[079eb43e] NLopt_jll v2.7.0+0
[e7412a2a] Ogg_jll v1.3.4+2
[458c3c95] OpenSSL_jll v1.1.1+6
[efe28fd5] OpenSpecFun_jll v0.5.4+0
[91d4177d] Opus_jll v1.3.1+3
[2f80f16e] PCRE_jll v8.44.0+0
```

```
[30392449] Pixman_jll v0.40.1+0
[ea2cea3b] Qt5Base_jll v5.15.2+0
[f50d1b31] Rmath_jll v0.3.0+0
[fb77eaff] Sundials_jll v5.2.0+1
[a2964d1f] Wayland_jll v1.17.0+4
[2381bf8a] Wayland_protocols_jll v1.18.0+4
[02c8fc9c] XML2_jll v2.9.12+0
[aed1982a] XSLT_jll v1.1.34+0
[4f6342f7] Xorg_libX11_jll v1.6.9+4
[0c0b7dd1] Xorg_libXau_jll v1.0.9+4
[935fb764] Xorg_libXcursor_jll v1.2.0+4
[a3789734] Xorg_libXdmcp_jll v1.1.3+4
[1082639a] Xorg_libXext_jll v1.3.4+4
[d091e8ba] Xorg_libXfixes_jll v5.0.3+4
[a51aa0fd] Xorg_libXi_jll v1.7.10+4
[d1454406] Xorg_libXinerama_jll v1.1.4+4
[ec84b674] Xorg_libXrandr_jll v1.5.2+4
[ea2f1a96] Xorg_libXrender_jll v0.9.10+4
[14d82f49] Xorg_libpthread_stubs_jll v0.1.0+3
[c7cfdc94] Xorg_libxcb_jll v1.13.0+3
[cc61e674] Xorg_libxkbfile_jll v1.1.0+4
[12413925] Xorg_xcb_util_image_jll v0.4.0+1
[2def613f] Xorg_xcb_util_jll v0.4.0+1
[975044d2] Xorg_xcb_util_keysyms_jll v0.4.0+1
[0d47668e] Xorg_xcb_util_renderutil_jll v0.3.9+1
[c22f9ab0] Xorg_xcb_util_wm_jll v0.4.1+1
[35661453] Xorg_xkbcomp_jll v1.4.2+4
[33bec58e] Xorg_xkeyboard_config_jll v2.27.0+4
[c5fb5394] Xorg_xtrans_jll v1.4.0+3
[8f1865be] ZeroMQ_jll v4.3.2+6
[3161d3a3] Zstd_jll v1.5.0+0
[0ac62f75] libass_jll v0.14.0+4
[f638f0a6] libfdk_aac_jll v0.1.6+4
[b53b4c65] libpng_jll v1.6.38+0
[a9144af2] libsodium_jll v1.0.20+0
[f27f6e37] libvorbis_jll v1.3.6+6
[1270edf5] x264_jll v2020.7.14+2
[dfaa095f] x265_jll v3.0.0+3
[d8fb68d0] xkbcommon_jll v0.9.1+5
[0dad84c5] ArgTools
[56f22d72] Artifacts
[2a0f44e3] Base64
[ade2ca70] Dates
[8bb1440f] DelimitedFiles
[8ba89e20] Distributed
[f43a241f] Downloads
[7b1f6079] FileWatching
[9fa8497b] Future
[b77e0a4c] InteractiveUtils
```

```
[b27032c2] LibCURL
[76f85450] LibGit2
[8f399da3] Libdl
[37e2e46d] LinearAlgebra
[56ddb016] Logging
[d6f4376e] Markdown
[a63ad114] Mmap
[ca575930] NetworkOptions
[44cfe95a] Pkg
[de0858da] Printf
[3fa0cd96] REPL
[9a3f8284] Random
[ea8e919c] SHA
[9e88b42a] Serialization
[1a1011a3] SharedArrays
[6462fe0b] Sockets
[2f01184e] SparseArrays
[10745b16] Statistics
[4607b0f0] SuiteSparse
[fa267f1f] TOML
[a4e569a6] Tar
[8dfed614] Test
[cf7118a7] UUIDs
[4ec0a83e] Unicode
[e66e0078] CompilerSupportLibraries_jll
[deac9b47] LibCURL_jll
[29816b5a] LibSSH2_jll
[c8ffd9c3] MbedTLS_jll
[14a3606d] MozillaCACerts_jll
[4536629a] OpenBLAS_jll
[bea87d4a] SuiteSparse_jll
[83775a58] Zlib_jll
[8e850ede] nghttp2_jll
[3f19e933] p7zip_jll
```