Inheritance

Different joints share many things:

- They are linked to a predecessor and successor body
- They have coordinate systems and locations for DP and DS
- The have functions to compute $\mathbf{A}_{IS}$, ${}_S\mathbf{r}_{IS}$, …

BUT:

- They differ in how the joint variable(s) are used to compute $\mathbf{A}_{DpDs}$, ${}_{Dp}\mathbf{r}_{DpDs}$, …
- This is a functional (not parametric) difference!
- How to implement this elegantly?

Advantage: Reuse code in a very clear structure (rather than copy & paste)

Example

Kinematics

$\vec{r}_{IB}$, $\vec{v}_B$, $\vec{a}_B$

A_{IB}, $\vec{\Omega}_B$, $\dot{\vec{\Omega}}_B$

integration Step

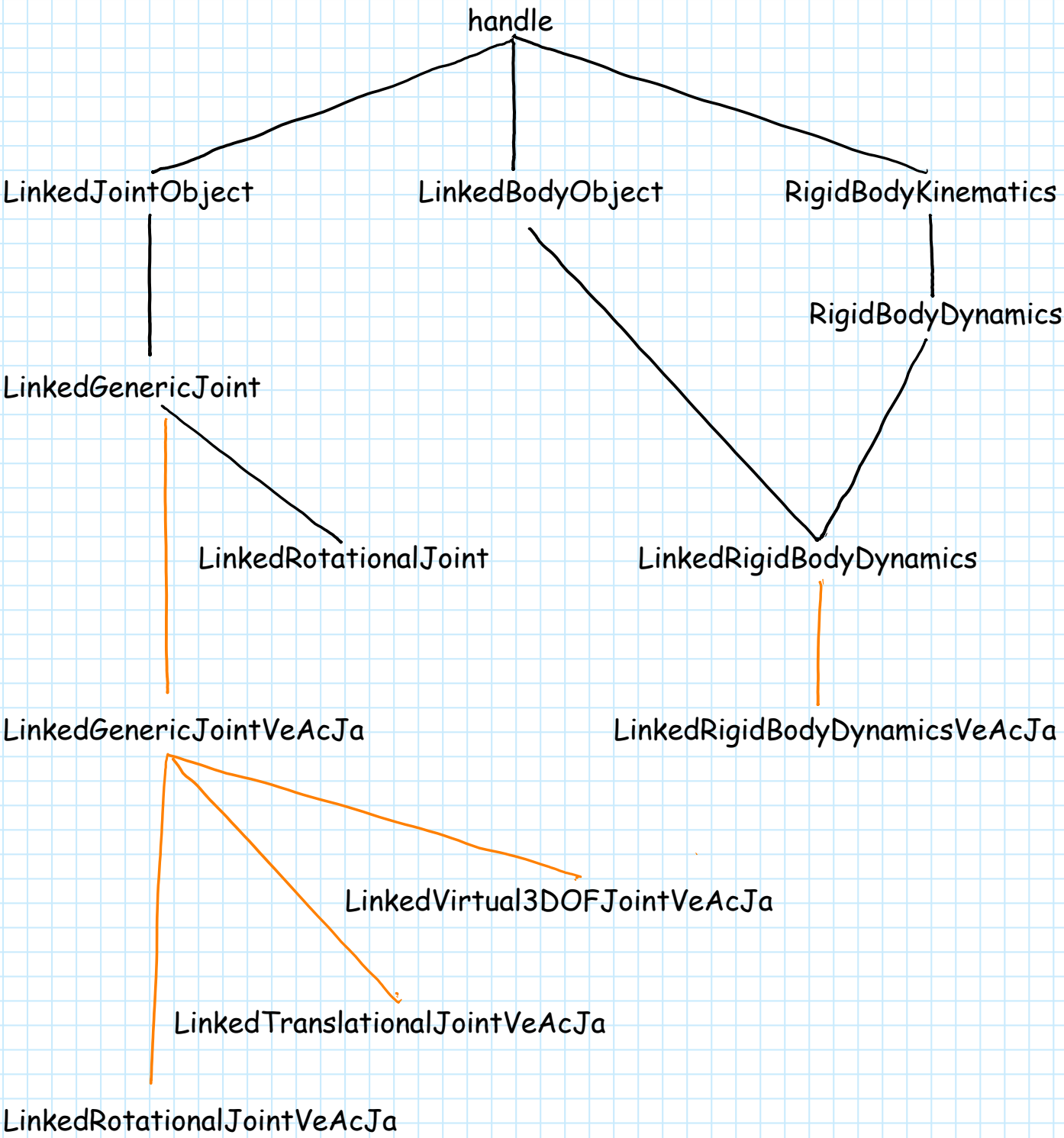
position Of Point

Dynamics

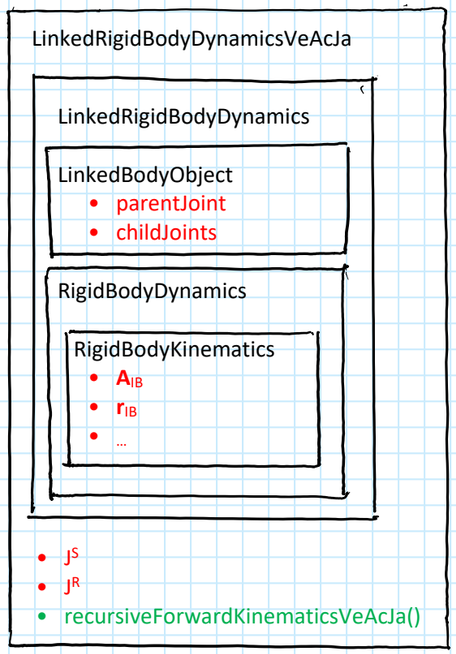
m_B, I_B

Compute Natural Dyn

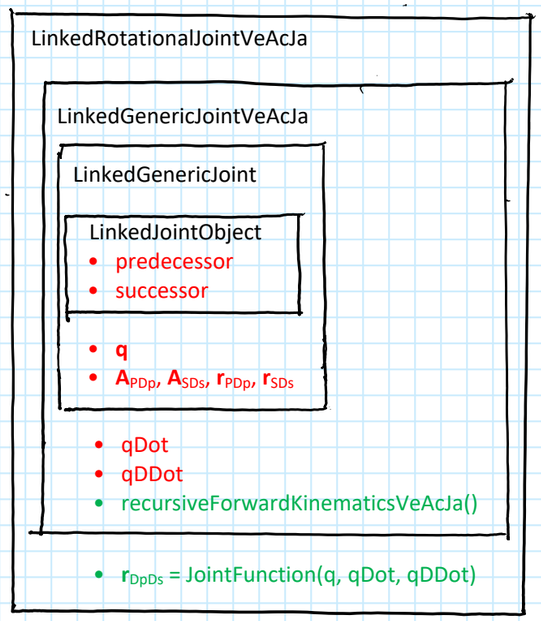
Inheritance creates a hirachy



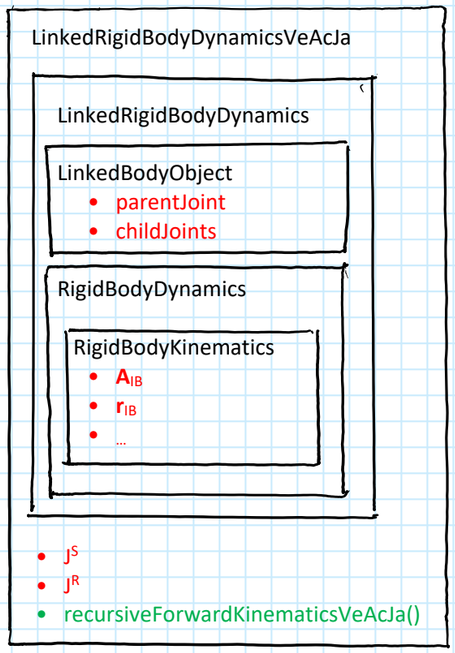
"Ground"



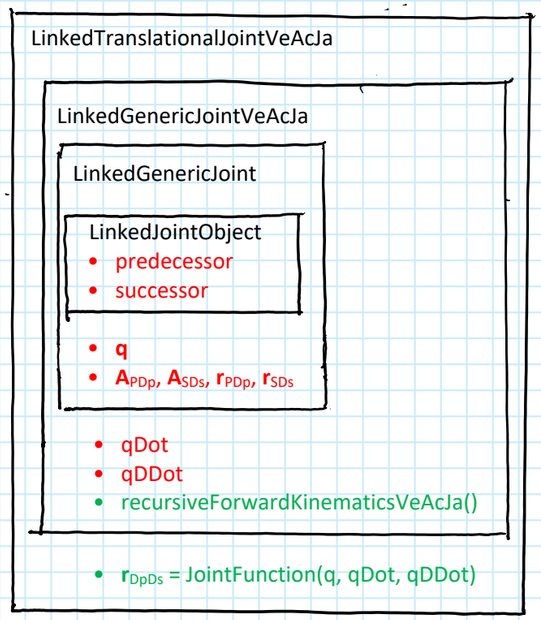
"Joint1", rotational



Link 1



"Joint2" translational




```
Ground = LinkedRigidBodyDynamicsVeAcJa(env);
Link1 = LinkedRigidBodyDynamicsVeAcJa(env);
Link2 = …

Joint1 = LinkedRotationalJointVeAcJa(env, Ground, Link1)
Joint2 = LinkedTranslationalJointVeAcJa(env, Link1, Link2)

Ground. RecursiveForwardKinematicsVeAcJa();
```