**算子知识图谱设计文档v2.3**

# 算子知识图谱的节点设计

## 1.1 框架节点

### 1.1.1 标签(label)

框架节点的标签命名格式为：“framework”。

### 1.1.2 属性(properties)

框架节点的属性命名包含两个键值对，其中，第一个键名为“name”，值为所采用的深度学习框架；第二个键名为“version”，值为该深度学习框架的版本。示例如下：{name: “Pytorch”, version: “1.5.0”}, {name: “MindSpore”, version: “1.5.0”}, {name: “PaddlePaddle”, version: “1.5.0”}。

## 1.2 类节点

### 1.2.1 标签(label)

类节点的标签命名格式为：“class”。

### 1.2.2 属性(properties)

类节点的属性命名包含两个键值对，其中，第一个键名为“framework”，值为所属的深度学习框架；第二个键名为“name”，值为相应的类名称。示例如下：{framework: “Pytorch”， name: “torch”}, { framework: “MindSpore”， name: “nn”}。

## 1.3 算子节点

### 1.3.1 标签(label)

算子节点的标签命名格式为：“operator”。

### 1.3.2 属性(properties)

算子节点的属性命名包含三个键值对，其中，第一个键名为“framework”，值为所属的深度学习框架；第二个键值对的键名为“name”，值为相应的算子名称；第三个键值对的键名为“full\_name”，值为该算子的完整路径名称。示例如下：{ framework: “Pytorch”， name: “Conv2d”， full\_name：“torch.nn.Conv2d” }。

## 1.4 参数节点

### 1.4.1 标签(label)

参数节点根据该参数是否有多个可选类型可以进一步分为父参数节点和子参数节点：如果该参数只有一种可选类型，则只需构建一个父参数节点；如果该参数有多种可选类型，则需要构建一个父参数节点与若干个连接到父参数节点的子参数节点。其中，父参数节点的标签命名格式为：“parameter”；子参数节点的标签命名格式为“childParameter”。

### 1.4.2 属性(properties)

父参数节点的属性命名包含六个必需键值对和一个非必需键值对，子参数节点的属性在父参数节点属性的基础上还包含一个额外的必需键值对。

父参数节点第一个必需的键值对的键名为“framework”，值为该参数所属的深度学习框架；第二个必需的键值对的键名为“operator”，值为该参数所处的算子名称；第三个必需键值对的键名为“parameter\_order”，值为该参数在所处算子中的顺序序号；第四个必需键值对的键名为“name”，值为相应的参数名称；第五个必需键值对的键名为“dtype”，名为该参数的类型名称；第六个必需键值对的键名为“default”，值为该参数的默认参数，若该参数没有默认值，则该值为“Null”。示例如下：{framework：“PyTorch”， operator: “Conv2d”, parameter\_order: 1, name: “in\_channels”, dtype: “int”， default: “Null”}。

若该参数有多个可选类型，则构建等同于可选类型数目的子参数节点。其中父参数节点的“dtype”属性的值重定义为参数可选类型的数目，子参数节点在父参数节点属性的基础上再添加一维属性，键名为“dtype\_order”，值为该类型在类型列表中所处的顺序序号。示例如下：若父参数节点有两种可选属性“int”和“tuple”，则父参数节点的属性为{ framework：“PyTorch”， operator: “Conv2d”, parameter\_order: 3, name: “kernel\_size”, dtype: 2， default: “Null”}，两个子参数节点的属性分别为{ framework：“PyTorch”， operator: “Conv2d”, dtype\_order: 1, parameter\_order: 3, name: “kernel\_size”, dtype : “int” ， default: “Null”}, { framework：“PyTorch”， operator: “Conv2d”, dtype\_order: 2, parameter\_order: 3, name: “kernel\_size”, dtype : “tuple” ， default: “Null”}。

参数节点属性的第一个非必需键值对的键名为“optional”，值为该参数的可选性，若该参数在相应的算子中是一个可选项，则该值为“True”。示例如下：父参数节点的属性为{ framework：“PyTorch”， operator: “Conv2d”, parameter\_order: 4, name: “stride”, dtype: 2, default: 1, optional: “True”}，两个子参数节点的属性分别为{ framework：“PyTorch”， operator: “Conv2d”, dtype\_order: 1, parameter\_order: 4, name: “stride”, dtype: “int”, default: 1, optional: “True”}，{ framework：“PyTorch”， operator: “Conv2d”, dtype\_order: 2, parameter\_order: 4, name: “stride”, dtype: “tuple”, default: “Null”， optional: “True”}。

## 1.5 输出节点(Output Nodes)

### 1.5.1 标签(label)

算子的输出节点根据该输出是否有多个可选类型可以进一步分为父输出节点和子输出节点：如果该参数只有一种可选类型，则只需构建一个父输出节点；如果该输出有多种可选类型，则需要构建一个父输出节点与若干个连接到父输出节点的子输出节点。其中，父输出节点的标签命名格式为：“output”；子输出节点的标签命名格式为“childOutput”。

### 1.5.2 属性(properties)

父输出节点的属性命名包含四个必需键值对，子输出节点的属性在父参数节点属性的基础上还包含一个额外的必需键值对。

父输出节点第一个必需的键值对的键名为“framework”，值为该输出所属的深度学习框架；第二个必需的键值对的键名为“operator”，值为该输出所处的算子名称；第三个必需键值对的键名为“output\_order”，值为该输出在所处算子中的返回顺序序号；第四个必需键值对的键名为“dtype”，名为该输出的类型名称。示例如下：{framework：“PyTorch”， operator: “Conv2d”, output\_order: 1, dtype: “Tensor”}。

若该输出有多个可选类型，则构建等同于可选类型数目的子输出节点。其中父输出节点的“dtype”属性的值重定义为参数可选类型的数目，子输出节点在父输出节点属性的基础上再添加一维属性，键名为“dtype\_order”，值为该类型在类型列表中所处的顺序序号。

## 1.6 成员函数节点(Member Function Nodes)

### 1.5.1 标签(label)

成员函数节点的标签命名格式为：“member\_function”。

### 1.5.2 属性(properties)

成员函数节点的属性命名包含三个键值对，其中，第一个键名为“framework”，值为所属的深度学习框架；第二个键值对的键名为“name”，值为相应的成员函数名称。示例如下：{ framework: “Pytorch”， name: “forward”}，{ framework：“MindSpore”， name: “construct”}。

# 算子知识图谱框架内的关系设计

## 2.1 框架节点与类节点之间的关系

### 2.1.1 类型(type)

框架节点与类节点之间关系的类型命名格式为：“classOfFramework”。

### 2.1.2 属性(properties)

框架节点与算子类节点之间关系的属性命名包含一个键值对，其中键名为“name”，值为所连接的类节点的名称。示例如下：{name: “torch”}。

## 2.2类节点与类节点之间的关系

### 2.2.1 类型(type)

类节点与类节点之间关系的类型命名格式为：“subClassOfClass”。

### 2.2.2 属性(properties)

类节点与类节点之间关系的属性命名包含一个键值对，其中键名为“name”，值为所连接的类节点的名称。示例如下：{name: “nn”}。

## 2.3类节点与算子节点之间的关系

### 2.3.1 类型(type)

类节点与算子节点之间关系的类型命名格式为：“operatorOfClass”。

### 2.3.2 属性(properties)

类节点与算子节点之间关系的属性命名包含一个键值对，其中键名为“name”，值为所连接的算子节点的名称。示例如下：{name: “Conv2d”}。

## 2.4算子节点与父参数节点之间的关系

### 2.4.1 类型(type)

算子节点与父参数节点之间关系的类型命名格式为：“parameterOfOperator”。

### 2.4.2 属性(properties)

算子节点与参数节点之间关系的属性命名包含两个键值对，其中第一个键名为“parameter\_order”，值为所连接的参数节点在当前算子的参数列表中所处的顺序；第二个键名为“name”，值为所连接的参数节点的名称。示例如下：{parameter\_order: 1, name: “in\_channels”}。

## 2.5父参数节点与子参数节点之间的关系

### 2.4.1 类型(type)

父参数节点与子参数节点之间关系的类型命名格式为：“oneOf”。

### 2.4.2 属性(properties)

参数节点与子参数节点之间关系的属性命名包含两个键值对，第一个键名为“dtype\_order”，值为所连接的子参数节点在当前参数节点的类型列表中所处的顺序；第二个键名为“dtype”，值为所连接的子参数节点的名称。示例如下：{dtype\_order: 1, dtype: “int”}。

## 2.6算子节点与父输出节点之间的关系

### 2.6.1 类型(type)

算子节点与父输出节点之间关系的类型命名格式为：“outputOfOperator”。

### 2.6.2 属性(properties)

算子节点与输出节点之间关系的属性命名包含一个键值对，其中第一个键名为“output\_order”，值为所连接的输出节点在当前算子的输出列表中所处的顺序。

## 2.7父输出节点与子输出节点之间的关系

### 2.4.1 类型(type)

父输出节点与子输出节点之间关系的类型命名格式为：“oneOf”。

### 2.4.2 属性(properties)

输出节点与子输出节点之间关系的属性命名包含两个键值对，第一个键名为“dtype\_order”，值为所连接的子输出节点在当前输出节点的列表中所处的顺序；第二个键名为“dtype”，值为所连接的子输出节点的名称。

## 2.8 框架节点与成员函数节点之间的关系

### 2.1.1 类型(type)

框架节点与类节点之间关系的类型命名格式为：“memberFunctionOfFramework”。

### 2.1.2 属性(properties)

框架节点与算子类节点之间关系的属性命名包含一个键值对，其中键名为“name”，值为所连接的成员函数节点的名称。示例如下：{name: “forward”}。

# 算子知识图谱框架间的关系设计

## 3.1 算子节点与算子节点之间的关系

### 3.1.1 类型(type)

一个框架的算子节点与另一个框架相应的算子节点之间关系的类型命名格式为：“equivalentOperator”。

### 3.1.2 属性(properties)

一个框架的算子节点与另一个框架相应的算子节点之间关系的属性包含一个键值对，其中，键名为“framework\_name”，值为所指向的算子节点所属框架的框架名称。示例如下：{framework\_name: “PyTorch”}，{framework\_name: “MindSpore”}，{framework\_name: “PaddlePaddle”}。

## 3.2 父参数节点与父参数节点之间的关系

### 3.1.1 类型(type)

一个框架算子的参数节点与另一个框架相应的参数节点之间关系的类型命名格式为：“equivalentParameter”。

### 3.1.2 属性(properties)

一个框架算子的参数节点与另一个框架相应的参数节点之间关系的属性包含两个键值对，其中，第一个键值对的键名为“framework\_name”，值为所指向的参数节点所属框架的框架名称。示例如下：{framework\_name: “PyTorch”}，{framework\_name: “MindSpore”}。

## 3.1 成员函数节点与成员函数节点之间的关系

### 3.1.1 类型(type)

一个框架的成员函数节点与另一个框架相应的成员函数节点之间关系的类型命名格式为：“equivalentMemberFunction”。

### 3.1.2 属性(properties)

一个框架的成员函数节点与另一个框架相应的成员函数节点之间关系的属性包含一个键值对，其中，键名为“framework\_name”，值为所指向的算子节点所属框架的框架名称。示例如下：{framework\_name: “PyTorch”}，{framework\_name: “MindSpore”}，{framework\_name: “PaddlePaddle”}。