设计思路

考虑到字典不会特别大，但可能是比较底层的库会被频繁调用，所以应尽可能的提高库的效率。

对于扁平化过程，作如下假设:

1. 字典的格式设计良好，如：列表元素格式统一
2. 不涉及复杂逻辑，如：某键的取舍与其它键不存在依赖关系

则对于输入字典：

1. 如果只有一层（即不考虑嵌套的情况），有如下操作：
    - 键被重命名
    - 键值对被舍弃
2. 如果只有两层（即字典内嵌套一层字典），除了 1 的情况，有如下操作：
    - 第2层的键值对被提高到第1层，例如 D['a']['b'] => D['b']
3. 对于多层的情况，根据 1、2 有如下结论：
    - 第 n 层的键值对可被提高到任意层，并被重命名（键）
4. 如果某键值对的值是单纯的列表（没有内嵌字典或列表），则：
    - 列表可被视作单纯的值
5. 如果 4 中列表L的元素是列表L2，则：
    - L2可被去掉，其中的元素可被放入L中，即与 3 类似，多层列表可被任意压缩
6. 如果 4 中列表L的元素是字典D，则：
    - 对于字典D中的某键值对，有 len(L) 条访问路径，与 2、3 有唯一一条访问路径不同
    - 如果只取D中的某一个键值对，则D仍然可以提级

综合上面分析：

1. 字典是一等公民，列表被视为普通值
2. 处于n层的某键值对可向上提升
3. 列表不可被消除
4. 嵌套列表可被合并

于是，可定义类似如下的模板格式：

```
key1: op1,param...; ...
        key2: op1,param...; ...
        key3: op1,param...; ...
```

例如，例子中的模板A，可定义为如下：

```
wlb_waybill_i_search_response: delete
    subscribtions: rename,subscribtions;
        waybill_apply_subscription_info: delete; list
            branch_account_cols: delete
                waybill_branch_account: rename,waybill_address; list
            shipp_address_cols: delete
                waybill_address: delete; list; compact
                    province:
                    city:
                    address_detail:
                    town:
                    area:
            cp_type: rename,express_type
            cp_code: rename,express_code
```

实现思路

考虑到字典的性质和字典嵌套深度不会太大的情况，可以选择用递归方法实现。

对于模板的格式，上面为了方便描述，采用了非标准格式，实际情况可用 json, yaml 或 xml 等。

库分为模板解析模块和字典生成模块。

模板解析模块：

解析模板解析模板，生成树状结构体（dict 或 class 等）。

字典生成模块：

字典生成模块遍历模板解析模块返回的结果，同时根据指令解析输入数据生成目标字典。