CHINA

自动机建模工具

自动机大作业说明报告

第 1 题 第 1 组
马晟、王玥、郭鑫
2010-1-8

本文档总结了自动机项目的设计和实现过程，最终说明交付产品的功能和体系结构，并总结了项目可扩展的方式。

目录

1. 自动机建模工具整体说明

1.1. 建模工具开发环境

1. Eclipse 3.2+
2. SVN Google Code Team Work
3. Java （JDK 1.6）
4. Eclipse RCP 平台框架
5. 开源 SVN 内容（http://code.google.com/p/automatonmodeling/）

1.2. 项目设计介绍

本项目完成了自动机第一题目的相关内容，为了便于项目的扩展性，我们把相关内容划分为三个部分，每个部分均可以进行扩展和优化或者内容补充。如图 1-1 所示，包括基础数据结构部分，XML 解析和读写部分，图形用户界面部分。

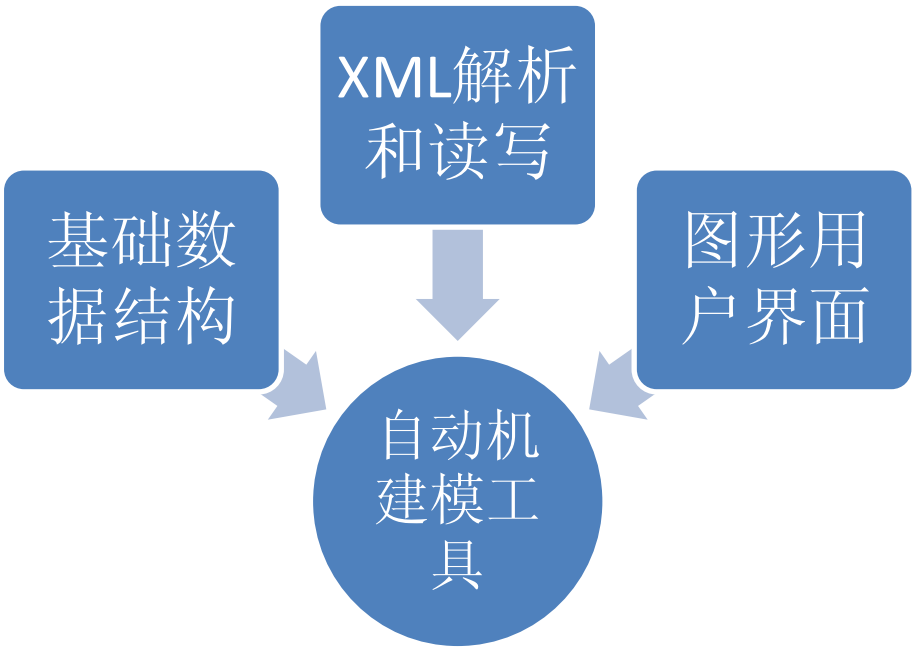

图 1-1：自动机项目体系结构图

2. 基础数据结构

在基础数据结构部分，这部分的数据结构将应用于自动机相关算法上。本项目包括 DFA、NFA 和 PDA，倘若未来需要拓展开发图灵机或者其他模型工具，可以在这个部分增加内容。

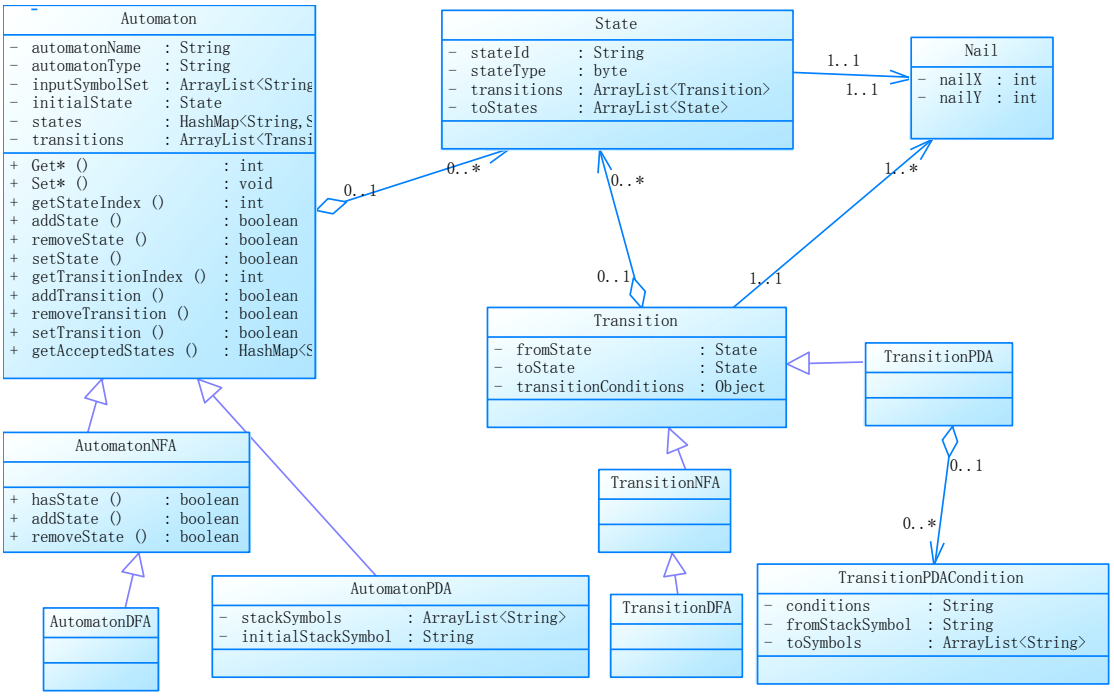

图 2-1 基础数据结构类图

在 AutomatonFactory 类中，我们定义了可以读写 XML 的静态方法，也定义了将自动机对象和图形界面的模型对象相互转化的方法。也定义了将模型中的对象转化为我们数据结构的对象的方法。

3. XML 解析

在 XML 解析和读写这个部分，我们项目中有 2 个包 xml 和 xml2，xml 中是开始定义的 DTD 文件和相应的 XML 解析，由于后来我们对 DTD 的内容有所改进，合并了 NFA 和 DFA，且改善了 XML 的格式，所以在 xml2 中做了新的 XML 读写，本项目现在使用的就是 xml2 中的读写。

解析的过程就是读取类中相应的数据内容，并根据节点内容生成 XML 文件，以及从 XML 文件中得到数据转化成相应的类。在 automaton 包中的 AutomatonXmlInterface 中定义了解析的接口。如图 3-1 所示类图。

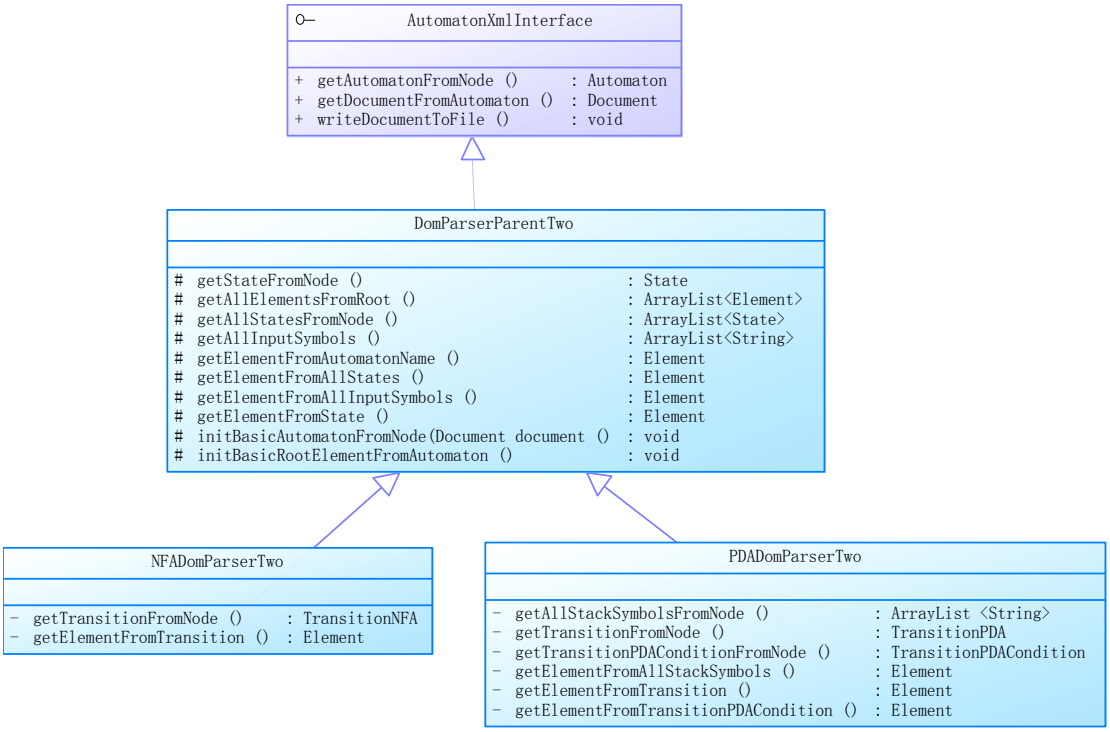

图 3-1 XML 解析类图

未来的拓展方法，可以使用新的 DTD 定义格式或者使用新的 XML 解析方法，比如使用新的 XML 解析类，比如用 Apache 的一些 XML 解析类等等，可以重新扩展 xml3，xml4……

对于 XML 文件的 DTD 定义如下面对话框

3.1. NFA 的 DTD 定义

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE NFA[
    <!ELEMENT NFA(States,InputSymbols,Transitions)>
        <!ATTLIST NFA type (DFA | NFA) #REQUIRED>
        <!ATTLIST NFA name #PCDATA "unnamed">
        <!ATTLIST NFA initialState #PCDATA "P1">

    <!ELEMENT States(State+)>
        <!ELEMENT State(StateId,StateType,Nail)>
            <!ELEMENT StateId(#PCDATA)>
            <!ELEMENT StateType(#PCDATA)>
            <!ELEMENT Nail(#PCDATA)>
                <!-- X,Y information is required -->
            <!ATTLIST Nail x #PCDATA>
            <!ATTLIST Nail y #PCDATA>
    <!ELEMENT InputSymbols(InputSymbol*)>
        <!ELEMENT InputSymbol(#PCDATA)>
    <!ELEMENT Transitions(NFATransition*)>
        <!ELEMENT
NFATransition(FromState,NFAConditions,ToState,Nails)>
            <!ELEMENT FromState(#PCDATA)>
            <!ELEMENT NFAConditions(NFACondition+)>
            <!ELEMENT ToState(#PCDATA)>
            <!ELEMENT Nails(Nail*)>
]>
```

DFA 其实跟 NFA 的区别只在于是否有 Eplison 转移，故实现过程中，我们将 DFA 继承 NFA 的几乎所有内容。

3.2. PDA 的 DTD 定义

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE PDA[
    <!ELEMENT
PDA(States,InputSymbols,StackSymbols,InitialStackSymbol,Transition
s)>
    <!ATTLIST NFA type (PDA) #REQUIRED>
    <!ATTLIST NFA name #PCDATA "unnamed">
    <!ATTLIST NFA initialState #PCDATA "P1">
    <!ELEMENT States(State+)>
        <!ELEMENT State(StateId,StateType,Nail)>
        <!ELEMENT StateId(#PCDATA)>
        <!ELEMENT StateType(#PCDATA)>
        <!ELEMENT Nail(#PCDATA)>
            <!-- X,Y information is required -->
        <!ATTLIST Nail x #PCDATA>
        <!ATTLIST Nail y #PCDATA>
    <!ELEMENT InputSymbols(InputSymbol+)>
        <!ELEMENT InputSymbol(#PCDATA)>
    <!ELEMENT StackSymbols(PDAStackSymbol+)>
        <!ELEMENT PDAStackSymbol(#PCDATA)>
    <!ELEMENT InitialStackSymbol(#PCDATA)>
    <!ELEMENT Transitions(PDATransition)>
        <!ELEMENT PDATransition(FromState,PDAConditions,ToState)>
            <!ELEMENT FromState(#PCDATA)>
            <!ELEMENT PDAConditions(PDACondition)>
                <!ELEMENT
PDACondition(ConditionSymbol,FromStackSymbol,ToStackSymbol)>
                <!ELEMENT ConditionSymbol(InputSymbol)>
                <!ELEMENT FromStackSymbol(#PCDATA)>
                <!ELEMENT ToStackSymbol(PDAStackSymbol+)>
            <!ELEMENT ToState(#PCDATA)>
]>
```

4. 用户界面及相关功能说明

我们项目复用 RCP 框架，结合 GEF 可视化编辑框架（即 Eclipse 里的 Graphical Editor Framework（GEF）创建图形化编辑器）实施开发自动机建模工具的图形界面。我们实现了项目要求的基本功能，如下说明：

4.1. 功能说明

- 创建新的自动机编辑文件（支持 PDA 和 NFA，DFA 三种形式自动机）
- 添加、删除、修改状态和转移，Delete 键删除状态或者转移
- 在属性栏中修改状态转移属性值，也可以设定状态类别（可接受的，初始状态或者普通状态）
- 右键修改状态和转移属性
- 增加转移中间结点，用于手动布局
- 支持 Undo 和 Redo 操作
- 将自动机保存成 XML 文件
- 从 XML 文件中读取自动机
- 放大、缩小模型

4.2. 使用说明

基本功能由功能说明里介绍的既可以使用，特殊说明如下：

- 对转移条件的判定符号用“;”分开，即如果在一个转移有多种转移条件可以写作“a;b;c”表示有 a,b,c 这三种符号时都转移。同理分隔 PDA 的多个转移条件。
- 调整转移的布局，每次增加一个 bending point，这个点是线段的中点，每次移动把转移分为两个部分。

4.3. 未来扩展建议

在开发过程中，我们也尝试了 GMF 的框架，这个框架是建立在 GEF 和 EMF 结合的基础上的。比 GEF 更加强大，而且更适合我们的自动机建模工具的开发。由于我们小组已经开发了部分内容，所以没有采用 GMF 的框架。倘若采用 XML 的框架，从开始定义自动机类的时候就可以开始运用这个框架了，这个框架会根据定义的类自动生成相应的可视化编辑框架，并生成相应的 XML 文件。倘若使用这个框架，我们项目的 XML 解析和开始的数据结构的内容就不需要写了。建议可以尝试这种开发模式，因为它比 GEF 内容更丰富。不过 GMF 的开发会有更多的依赖包，详细内容参考 Eclipse 官方网站插件。

此外，GMF 支持验证。这句话的含义是：如果我们要限定图形化模型，只许与每个模型元素有一个连接会怎样？只允许相似的元素相互连接，还是对可用

于图形的名称类型进行控制？GMF 完全能够支持这些类型的验证，甚至其他验证。对于验证，GMF 利用了 Eclipse Modeling Framework Technology（EMFT）来支持涉及使用 Java 代码和对象约束语言（Object Constraint Language，OCL）来定义验证器的情况。

5. 附录

5.1. 文档修改记录

修改时间	修改内容	修改者
2010-1-8	完成基本文档	马晟、王玥、郭鑫小组

5.2. 项目需求

1. 用 java 编写，保证可扩展性。
2. 有完整的友好的图形界面，支持鼠标操作，鼠标支持经过自动机元素时高亮，便捷的选择操作。有针对“状态”和“变迁”的属性修改页面。
3. 将自动机保存至 XML，并且对应好对应的 DTD。
4. 能够读取定义好的 XML 文件。
5. 定义好自动机的数据结构，并能够从某个接口将自动机结构从 XML 中读出。
6. 支持定义一些自动机的属性，比如是 DNF 还是 CNF 还是 PDA。
7. （Optional）支持对自动机进行自动的布局。