

Mar 08, 04 18:17

FA.java

Page 1/6

```

package dRegAut;
import java.util.*;
import java.io.*;

/**
 * Deterministic finite state automaton. [Martin, Def. 3.2]
 */
public class FA
{
    /** Set of {@link State} objects (Q). */
    public Set<State> states;

    /** The automaton alphabet (). */
    public Alphabet alphabet;

    /** Initial state (q<sub>0</sub>). Member of {@link #states}. */
    public State initial;

    /** Accept states (A). Subset of {@link #states}. */
    public Set<State> accept;

    /**
     * Transition function ().
     * This is a map from pairs of states and alphabet symbols to states
     * (:
     * Q x A -> Q).
     */
    public Map<StateSymbolPair, State> transitions;

    /**
     * Checks that this automaton is well-defined.
     * In particular, this method checks that the transition function is total.
     * This method should be invoked after each <tt>FA</tt> operation during tes
     ting.
     */
    @return this automaton
    @exception AutomatonNotWellDefinedException if this automaton is not well
    -defined
    */
    public FA checkWellDefined() throws AutomatonNotWellDefinedException
    {
        if (states==null || alphabet==null || alphabet.symbols==null ||
            initial==null || accept==null || transitions==null)
            throw new AutomatonNotWellDefinedException("invalid null pointer");
        if (!states.contains(initial))
            throw new AutomatonNotWellDefinedException("the initial state is not in the state set");
        if (!states.containsAll(accept))
            throw new AutomatonNotWellDefinedException("not all accept states are in the state set");
        for (State s : states) {
            for (Character c : alphabet.symbols) {
                if (c==NFA.LAMBDA.LAMBDA)
                    throw new AutomatonNotWellDefinedException("lambda (null) transition appears in transitions");
                State s2 = transitions.get(new StateSymbolPair(s, c));
                if (s2==null)
                    throw new AutomatonNotWellDefinedException("transition function is not total");
                if (!states.contains(s2))
                    throw new AutomatonNotWellDefinedException("there is a transition to a state that is not in state set");
            }
        }
        for (StateSymbolPair sp : transitions.keySet()) {
            if (!states.contains(sp.state))
                throw new AutomatonNotWellDefinedException("transitions refer to a state not in the state set");
            if (!alphabet.symbols.contains(sp.symbol))
                throw new AutomatonNotWellDefinedException("non-alphabet symbol appears in transitions");
        }
        return this;
    }

    /**
     * Constructs an uninitialized FA.
     * <tt>states</tt> and <tt>accept</tt> are set to empty sets,
     * <tt>transitions</tt> is set to an empty map.
     */
    public FA()
    {
        states = new HashSet<State>();
        accept = new HashSet<State>();
        transitions = new HashMap<StateSymbolPair, State>();
    }
}

```

Tuesday June 01, 2004

Mar 08, 04 18:17

FA.java

Page 2/6

```

* Constructs a new FA consisting of one reject state.
* @param a automaton alphabet
*/
public FA(Alphabet a)
{
    states = new HashSet<State>();
    accept = new HashSet<State>();
    alphabet = a;

    // make a state
    State s = new State();
    states.add(s);
    initial = s;

    // add a loop transition for each alphabet symbol
    transitions = new HashMap<StateSymbolPair, State>();
    for (Character c : alphabet.symbols) {
        transitions.put(new StateSymbolPair(s, c), s);
    }
}

/** Clones this automaton. */
public Object clone()
{
    FA f = new FA();
    f.alphabet = alphabet;
    Map<State, State> m = new HashMap<State, State>(); // map from old states
    to new states
    for (State p : states) {
        State s = (State)p.clone();
        f.states.add(s);
        m.put(p, s);
        if (accept.contains(p))
            f.accept.add(s);
    }
    f.initial = (State)m.get(initial);
    for (Map.Entry<StateSymbolPair, State> e : transitions.entrySet()) {
        StateSymbolPair ssp = e.getKey();
        State q = e.getValue();
        f.transitions.put(new StateSymbolPair(m.get(ssp.state), ssp.symbol),
        m.get(q));
    }
    return f;
}

/**
 * Returns <a href="http://www.research.att.com/sw/tools/graphviz/" target="
    _top">Graphviz Dot</a>
 * representation of this automaton.
 * (<tt>To convert a dot file to postscript, run '<tt>dot -Tps -o file.ps file.d
    ot</tt>'</tt>).
 */
public String toDot()
{
    StringBuffer b = new StringBuffer("digraph Automaton {\n");
    b.append(" rankdir LR;\n");
    Map<State, Integer> id = new HashMap<State, Integer>();
    for (State s : states) {
        id.put(s, id.size());
    }
    for (State s : states) {
        b.append(" ").append(id.get(s));
        if (accept.contains(s))
            b.append(" [shape=circle,label=\""+s.name+"\\"];
        else
            b.append(" [shape=circle,label=\""+s.name+"\\"];
        if (s==initial) {
            b.append(" in [shape=plaintext,label=\\"];
            b.append(" in -> ").append(id.get(s)).append("\\n");
        }
    }
    for (Map.Entry e : transitions.entrySet()) {
        StateSymbolPair ssp = (StateSymbolPair) e.getKey();
        State q = (State) e.getValue();
        b.append(" ").append(id.get(ssp.state)).append("-> ").append(id.get(
        q));
        b.append(" [label=\\"];
        char c = ssp.symbol.charValue();
        if (c>=0x21 & c<=0x7e & c!='\\' & c!='%')
            b.append(c);
        else {
            b.append("\\u");
            String s = Integer.toHexString((int) c);
            if (c<0x10)
                b.append("000").append(s);
            else if (c<0x100)
                b.append("00").append(s);
            else if (c<0x1000)
                b.append("0").append(s);
            else
                b.append(s);
        }
        b.append("\\\\n");
    }
}

```

FA.java

Mar 08, 04 18:17

FA.java

Page

```

return b.append("\\n").toString();
}

/** Returns number of states in this automaton. */
public int getNumberOfStates()
{
    return states.size();
}

/**
 * Sets a transition in the transition function.
 * =p
    ,c)=p
 */
public void setTransition(State q, char c, State p)
{
    transitions.put(new StateSymbolPair(q, c), p);
}

/**
 * Looks up transition in transition function.
 * @return (q,c)
 * @exception IllegalArgumentException if <tt>c</tt> is not in the al
    phabet
 */
public State delta(State q, Character c) throws IllegalArgumentException
{
    if (!alphabet.symbols.contains(c))
        throw new IllegalArgumentException("symbol '"+c+"' not in alphabet");
    return (State) transitions.get(new StateSymbolPair(q, c));
}

/**
 * Performs transitions in extended transition function. [Martin, Def.
    3.2]
 * @return (q,s)
 * @exception IllegalArgumentException if a symbol in <tt>s</tt> is n
    ot in the alphabet
 */
public State deltaStar(State q, String s) throws IllegalArgumentException
{
    for (int i = 0; i<s.length(); i++)
        q = delta(q, new Character(s.charAt(i)));
    return q;
}

protected enum ProductOperation {UNION, INTERSECT, MINUS};

/**
 * Performs the core FA product operation.
 * @param f1 The 'left' operand.
 * @param f2 The 'right' operand.
 * @param kind Either 'u', 'i' or 'm' depending on whether a
    union, intersection or minus operation should be performed.
 */
protected static FA product(FA f1, FA f2, ProductOperation kind)
throws IllegalArgumentException {
    FA r = new FA();
    f1.checkWellDefined(); f2.checkWellDefined();
    if (!f1.alphabet.equals(f2.alphabet))
        throw new IllegalArgumentException("alphabets do not match");
    r.alphabet = f1.alphabet;
    HashMap<StatePair, State> statemap = new HashMap<StatePair, State>();
    /* create new states */
    for (State s1 : f1.states)
        for (State s2 : f2.states) {
            State sn = new State(s1.name + s2.name);
            r.states.add(sn);
            statemap.put(new StatePair(s1, s2), sn);
        }
    //System.err.println("Created state " + s1.name + s2.name);
    /* check accept state */
    switch (kind) {
        case UNION:
            if (f1.accept.contains(s1) || f2.accept.contains(s2))
                r.accept.add(sn);
            break;
        case INTERSECT:
            if (f1.accept.contains(s1) & f2.accept.contains(s2))
                r.accept.add(sn);
            break;
        case MINUS:
            if (f1.accept.contains(s1) & !f2.accept.contains(s2))
                r.accept.add(sn);
            break;
    }
}

```

Mar 08, 04 18:17

FA.java

Page 4/6

```

    }
}
/* create new transitions */
for (State s1 : f1.states) {
    for (State s2 : f2.states) {
        State src = statemap.get(new StatePair(s1, s2));
        for (char c : r.alphabet.symbols)
            r.setTransition(src, c,
                statemap.get(new StatePair(f1.delta(s1, c),
                    f2.delta(s2, c))));
    }
}

/* set initial state */
r.initial = statemap.get(new StatePair(f1.initial, f2.initial));

return r;
}

/**
 * Runs the given string on this automaton. [Martin, Def. 3.4]
 * @param s a string
 * @return true iff the string is accepted
 * @exception IllegalArgumentException if a symbol in <tt>s</tt> is not in t
he alphabet
 */
public boolean accepts(String s) throws IllegalArgumentException
{
    /* IllegalArgumentException is thrown by delta() */
    return this.accept.contains(deltaStar(this.initial, s));
}

/** Pair of states. Used in product construction and in construction of regu
lar expression. */
private static class StatePair
{
    State s1, s2;

    /** Constructs a new pair. */
    StatePair(State s1, State s2)
    {
        this.s1 = s1;
        this.s2 = s2;
    }

    /** Checks whether two pairs are equals. */
    public boolean equals(Object obj)
    {
        if (!(obj instanceof StatePair))
            return false;
        StatePair ss = (StatePair) obj;
        return s1==ss.s1 & s2==ss.s2;
    }

    /** Computes hash code for this object. */
    public int hashCode()
    {
        return s1.hashCode()*3 + s2.hashCode()*2;
    }
}

/** Converts this automaton into an equivalent {@link RegExp} regular expres
sion. [Martin, Th. 4.5] */
public RegExp toRegExp()
{
    throw new UnsupportedOperationException("method not implemented yet!");
}

/**
 * Constructs a new automaton that accepts the complement of the language of
this automaton. [Martin, p. 110]
 * The input automaton is unmodified.
 */
public FA complement()
{
    FA f = (FA) clone();
    Set<State> s = new HashSet<State>();
    s.addAll(f.states);
    s.removeAll(f.accept);
    f.accept = s;
    return f;
}

/** Finds the set of states that are reachable from the initial state. */
public Set findReachableStates()
{
    throw new UnsupportedOperationException("method not implemented yet!");
}

/**
 * Constructs a new automaton with the same language as this automaton but w
ithout unreachable states. [Martin, Exercise 3.29]

```

Mar 08, 04 18:17

FA.java

Page 5/6

```

 * The input automaton is unmodified.
 */
public FA removeUnreachableStates()
{
    throw new UnsupportedOperationException("method not implemented yet!");
}

/**
 * Constructs a new minimal automaton with the same language as this automa
ton. [Martin, Sec. 5.2]
 * The input automaton is unmodified.
 * Note: this textbook algorithm is simple to understand but not very effici
ent
 * compared to other existing algorithms.
 */
public FA minimize()
{
    throw new UnsupportedOperationException("method not implemented yet!");
}

/** Checks whether the language of this automaton is finite. [Martin, Sec. 5
.4] */
public boolean isFinite()
{
    throw new UnsupportedOperationException("method not implemented yet!");
}

/** Checks whether the language of this automaton is empty. [Martin, Sec. 5.
4] */
public boolean isEmpty()
{
    throw new UnsupportedOperationException("method not implemented yet!");
}

/** Checks whether the language of this automaton is a subset of the languag
e of the given automaton. [Martin, Sec. 5.4] */
public boolean subsetOf(FA f)
{
    throw new UnsupportedOperationException("method not implemented yet!");
}

/**
 * Computes hash code for this object.
 * (When {@link #equals(Object)} is implemented, <tt>hashCode</tt> must also
be there.)
 */
public int hashCode()
{
    return getNumberOfStates(); // a very simple but valid hash code
}

/** Checks whether the language of this automaton is equal to the language o
f the given automaton. [Martin, Sec. 5.4] */
public boolean equals(Object obj)
{
    throw new UnsupportedOperationException("method not implemented yet!");
}

/**
 * Returns a shortest string that is accepted by this automaton.
 * @return a (not necessarily unique) shortest example string, null if the l
anguage of this automaton is empty
 */
public String getAShortestExample()
{
    throw new UnsupportedOperationException("method not implemented yet!");
}

/**
 * Constructs a new automaton whose language is the intersection of the lang
uage of this automaton
 * and the language of the given automaton. [Martin, Th. 3.4]
 * The input automata are unmodified.
 * @exception IllegalArgumentException if the alphabets of <tt>f</tt> and th
is automaton are not the same
 */
public FA intersection(FA f) throws IllegalArgumentException
{
    return product(this, f, ProductOperation.INTERSECT);
}

/**
 * Constructs a new automaton whose language is the union of the language of
this automaton
 * and the language of the given automaton. [Martin, Th. 3.4]
 * The input automata are unmodified.
 * @exception IllegalArgumentException if the alphabets of <tt>f</tt> and th
is automaton are not the same
 * @see NFALambda#union
 */
public FA union(FA f) throws IllegalArgumentException
{
    return product(this, f, ProductOperation.UNION);
}

```

Mar 08, 04 18:17

FA.java

Page

```

/**
 * Constructs a new automaton whose language is equal to the language
s automaton
 * minus the language of the given automaton. [Martin, Th. 3.4]
 * The input automata are unmodified.
 * @exception IllegalArgumentException if the alphabets of <tt>f</tt>
is automaton are not the same
 */
public FA minus(FA f) throws IllegalArgumentException
{
    return product(this, f, ProductOperation.MINUS);
}

/**
 * Returns an NFAM representation of this automaton.
 */
public NFA toNFA() {
    NFA f = new NFA();
    f.states = this.states;
    f.alphabet = this.alphabet;
    f.accept = this.accept;
    f.initial = this.initial;

    f.transitions = new HashMap<StateSymbolPair, Set<State>>();
    for (Map.Entry<StateSymbolPair, State> e : transitions.entrySet())
        Set<State> set = new HashSet<State>();
        set.add(e.getValue());
        f.transitions.put(e.getKey(), set);
}

return f;
}
}

```

Feb 23, 04 21:45

NFA.java

Page 1/6

```

package dRegAut;
import java.util.*;
import java.io.*;

/**
 * Nondeterministic finite state automaton. [Martin, Def. 4.1]
 */
public class NFA
{
    /** Set of {@link State} objects (Q). */
    public Set<State> states;

    /** The automaton alphabet (). */
    public Alphabet alphabet;

    /** Initial state (q<sub>0</sub>). Member of {@link #states}. */
    public State initial;

    /** Accept states (A). Subset of {@link #states}. */
    public Set<State> accept;

    /**
     * Transition function ().
     * This is a map from pairs of states and alphabet symbols to sets of states
     * ().
     *
     * 
     * alt="x" 
     * 
     * <sup>Q</sup></sup>.
     */
    public Map<StateSymbolPair, Set<State>> transitions;

    /**
     * Checks that this automaton is well-defined.
     * This method should be invoked after each <tt>NFA</tt> operation during te
     sting.
     * @return this automaton
     * @exception AutomatonNotWellDefinedException if this automaton is not well
     -defined
     */
    public NFA checkWellDefined() throws AutomatonNotWellDefinedException
    {
        if (states==null || alphabet==null || alphabet.symbols==null ||
            initial==null || accept==null || transitions==null)
            throw new AutomatonNotWellDefinedException("invalid null pointer");
        Iterator i = states.iterator();
        while (i.hasNext()) {
            Object o = i.next();
            if (!(o instanceof State))
                throw new AutomatonNotWellDefinedException("non-State object appears in s
                tate set");
        }
        i = alphabet.symbols.iterator();
        while (i.hasNext()) {
            Object o = i.next();
            if (!(o instanceof Character))
                throw new AutomatonNotWellDefinedException("non-Character object appears
                in alphabet");
        }
        if (!states.contains(initial))
            throw new AutomatonNotWellDefinedException("the initial state is not in the state set
        ");
        if (!states.containsAll(accept))
            throw new AutomatonNotWellDefinedException("not all accept states are in the state
        set");
        i = states.iterator();
        while (i.hasNext()) {
            State s = (State) i.next();
            if (transitions.get(new StateSymbolPair(s, NFALambda.LAMBDA))!=null)
                throw new AutomatonNotWellDefinedException("lambda (null) transition
                appears in transitions");
            Iterator j = alphabet.symbols.iterator();
            while (j.hasNext()) {
                Character c = (Character) j.next();
                Object o = transitions.get(new StateSymbolPair(s, c));
                if (o!=null) {
                    if (!(o instanceof Set))
                        throw new AutomatonNotWellDefinedException("non-Set object a
                        ppears in transitions");
                    Set ps = (Set) o;
                    Iterator k = ps.iterator();
                    while (k.hasNext()) {
                        Object o2 = k.next();
                        if (!(o2 instanceof State))
                            throw new AutomatonNotWellDefinedException("non-State
                        object appears in transitions");
                    }
                    if (!states.contains(o2))
                        throw new AutomatonNotWellDefinedException("there is a tr
                ansition to a state that is not in state set");
                }
            }
        }
    }
}

```

Tuesday June 01, 2004

Feb 23, 04 21:45

NFA.java

Page 2/6

```

    }
    }
    i = transitions.keySet().iterator();
    while (i.hasNext()) {
        StateSymbolPair sp = (StateSymbolPair) i.next();
        if (!states.contains(sp.state))
            throw new AutomatonNotWellDefinedException("transitions refer to a state not
        in the state set");
        if (!alphabet.symbols.contains(sp.symbol))
            throw new AutomatonNotWellDefinedException("non-alphabet symbol appears
        in transitions");
        return this;
    }

    /**
     * Constructs an uninitialized NFA.
     * <tt>states</tt> and <tt>accept</tt> are set to empty sets,
     * <tt>transitions</tt> is set to an empty map.
     */
    public NFA()
    {
        states = new HashSet<State>();
        accept = new HashSet<State>();
        transitions = new HashMap<StateSymbolPair, Set<State>>();
    }

    /**
     * Constructs a new NFA consisting of one reject state.
     * @param a automaton alphabet
     */
    public NFA(Alphabet a)
    {
        states = new HashSet<State>();
        accept = new HashSet<State>();
        alphabet = a;
        State s = new State();
        states.add(s);
        initial = s;
        transitions = new HashMap<StateSymbolPair, Set<State>>();
    }

    /**
     * Returns <a href="http://www.research.att.com/sw/tools/graphviz/" target="
     _top">Graphviz Dot</a>
     * representation of this automaton.
     * (To convert a dot file to postscript, run <tt>dot -Tps -o file.ps file.d
     ot</tt>.)
     */
    public String toDot()
    {
        StringBuffer b = new StringBuffer("digraph Automaton (\n");
        b.append(" rankdir=LR\n");
        Map<State, Integer> id = new HashMap<State, Integer>();
        Iterator i = states.iterator();
        while (i.hasNext()) {
            State s = (State) i.next();
            id.put(s, id.size());
        }
        i = states.iterator();
        while (i.hasNext()) {
            State s = (State) i.next();
            b.append(" ").append(id.get(s));
            if (accept.contains(s))
                b.append(" [shape=doublecircle,label="+s.name+"\n");
            else
                b.append(" [shape=circle,label="+s.name+"\n");
            if (s==initial)
                b.append(" in [shape=plaintext,label="\n");
            b.append(" in-> ").append(id.get(s)).append("\n");
        }
        for (Map.Entry<StateSymbolPair, Set<State>> e: transitions.entrySet()) {
            StateSymbolPair ssp = e.getKey();
            for (State q: e.getValue()) {
                b.append(" ").append(id.get(ssp.state)).append(" -> ").append(id.
                get(q));
                b.append(" [label=");
                char c = ssp.symbol.charValue();
                if (c>=0x21 & c<=0x7e & c!='\ ' & c!='%')
                    b.append(c);
                else {
                    b.append("\u");
                    String s = Integer.toHexString((int) c);
                    if (c<=0x10)
                        b.append("000").append(s);
                    else if (c<=0x100)
                        b.append("00").append(s);
                    else if (c<=0x1000)
                        b.append("0").append(s);
                    else
                        b.append(s);
                }
            }
        }
    }
}

```

NFA.java

Feb 23, 04 21:45

NFA.java

Page

```

        b.append("\n\n");
    }
    }
    return b.append("(").append(")\n").toString();
}

/** Returns number of states in this automaton. */
public int getNumberOfStates()
{
    return states.size();
}

/**
 * Looks up transition in transition function.
 * @return 
 * (q, c), the result is never null and it should never be modified
 */
public Set<State> delta(State q, Character c) throws IllegalArgumentException
{
    if (!alphabet.symbols.contains(c))
        throw new IllegalArgumentException("symbol " + c +
        " not in alphabet");
    Set<State> set = transitions.get(new StateSymbolPair(q, c));
    if (set==null)
        set = new HashSet<State>();
    return set;
}

/**
 * Performs transitions in extended transition function. [Martin, Def. 4.3]
 * @return 
 * (q, s) (the result is never null)
 * @exception IllegalArgumentException if a symbol in <tt>s</tt> is n
 */
public Set<State> deltaStar(State q, String s) throws IllegalArgumentException
{
    Set<State> set = new HashSet<State>();
    set.add(q);
    for (int i=0; i<s.length(); i++) {
        Set<State> nset = new HashSet<State>();
        for (State p: set)
            nset.addAll(delta(p, s.charAt(i)));
        set = nset;
    }
    return set;
}

/**
 * Runs the given string on this automaton. [Martin, Def. 4.3]
 * @param s a string
 * @return true iff the string is accepted
 * @exception IllegalArgumentException if a symbol in <tt>s</tt> is n
 */
public boolean accepts(String s) throws IllegalArgumentException
{
    Set<State> set = deltaStar(initial, s);
    set.retainAll(accept);
    return !set.isEmpty();
}

/**
 * Constructs a new minimal automaton with the same language as
 * this automaton. [Martin, Sec. 5.2] The input automaton is
 * unmodified.
 * @see FA#minimize
 */
public FA minimize()
{
    return this.toNFAM().reverse().determine().toNFA().toNFAM().rev
    erse().determine();
}

/**
 * Converts this NFA into an equivalent (<link NFALambda>).
 * Note: this is trivial since an NFA is just a special kind of NFAL
 * ee [Martin, Th. 4.3]).
 */
public NFALambda toNFALambda()
{
    NFALambda f = new NFALambda();
    f.alphabet = alphabet;
    Map<State, State> m = new HashMap<State, State>(); // map from old
    to new states
    for (State p: states) {
        State s = (State) p.clone();
        f.states.add(s);
    }
}

```

Feb 23, 04 21:45

NFA.java

Page 4/6

```

        m.put(p, s);
        if (accept.contains(p))
            f.accept.add(s);
        if (p==initial)
            f.initial = s;
    }
    for (Map.Entry<StateSymbolPair, Set<State>> e: transitions.entrySet()) {
        StateSymbolPair ssp = e.getKey();
        Set<State> set = e.getValue();
        Set<State> ns = new HashSet<State>();
        Iterator j = set.iterator();
        while (j.hasNext()) {
            State q = (State) j.next();
            ns.add(m.get(q));
        }
        f.transitions.put(new StateSymbolPair((State) m.get(ssp.state), ssp.
symbol), ns);
    }
    return f;
}
/**
 * Determines this automaton by constructing an equivalent (@link FA). [Ma
rtin, Th. 4.1]
 * The input automaton is unmodified.
 * This implementation only constructs the reachable part of the subset stat
e space.
 */
public FA determinize()
{
    return this.toNFAM().determinize();
}

public String toReverseDot() {
    return this.toNFAM().reverse().toDot();
}

/**
 * Returns an NFAM representation of this automaton.
 */
private NFAM toNFAM() {
    NFAM f = new NFAM();
    f.states = this.states;
    f.alphabet = this.alphabet;
    f.accept = this.accept;
    f.transitions = this.transitions;

    f.initials = new HashSet<State>();
    f.initials.add(this.initial);

    return f;
}

private class NFAM {
    public Set<State> states;
    public Alphabet alphabet;
    public Set<State> initials;
    public Set<State> accept;
    public Map<StateSymbolPair, Set<State>> transitions;

    /**
     * Returns this automaton
     */
    public NFAM reverse() {
        NFAM f = new NFAM();

        f.states = new HashSet<State>();
        f.accept = new HashSet<State>();
        f.initials = new HashSet<State>();
        f.transitions = new HashMap<StateSymbolPair, Set<State>>();

        f.states.addAll(states);
        f.accept.addAll(accept);
        f.initials.addAll(initials);

        f.alphabet = this.alphabet;

        for (Map.Entry<StateSymbolPair, Set<State>> e: this.transitions.entry
Set())
            for (State s: e.getValue()) {
                StateSymbolPair ssp = new StateSymbolPair(s, e.getKey().symb
ol);

                if (!f.transitions.containsKey(ssp))
                    f.transitions.put(ssp, new HashSet<State>());

                f.transitions.get(ssp).add(e.getKey().state);
            }

        return f;
    }

    /**
     * @see NFA#determinize

```

Feb 23, 04 21:45

NFA.java

Page 5/6

```

    */
    public FA determinize()
    {
        FA f = new FA();
        f.alphabet = this.alphabet;

        Map<Set<State>, State> ssmap = new HashMap<Set<State>, State>();

        int state_count = 0;

        Set<State> current_state = new HashSet<State>();
        current_state = this.initials;
        f.initial = new State(" " + state_count++);
        f.states.add(f.initial);
        ssmap.put(current_state, f.initial);

        Stack<Set<State>> unprocessed_states = new Stack<Set<State>>();
        unprocessed_states.push(current_state);

        while (!unprocessed_states.isEmpty()) {
            current_state = unprocessed_states.pop();

            for (char c: alphabet.symbols) {
                Set<State> next_state = new HashSet<State>();
                for (State s: current_state) {
                    next_state.addAll(delta(s, c));
                    if (accept.contains(s))
                        f.accept.add(ssmap.get(current_state));
                }

                if (!ssmap.containsKey(next_state)) {
                    State n = new State(" " + state_count++);
                    ssmap.put(next_state, n);
                    f.states.add(n);
                    unprocessed_states.push(next_state);
                }

                f.setTransition(ssmap.get(current_state),
                    c, ssmap.get(next_state));
            }
        }

        return f;
    }

    /**
     * Returns <a href="http://www.research.att.com/sw/tools/graphviz/" target="
_top">Graphviz Dot</a>
     * representation of this automaton.
     * (To convert a dot file to postscript, run 'dot -Tps -o file.ps file.d
ot'.)
     */
    public String toDot()
    {
        StringBuffer b = new StringBuffer("digraph Automaton {\n");
        b.append(" rankdir = LR;\n");
        Map<State, Integer> id = new HashMap<State, Integer>();
        Iterator i = states.iterator();
        while (i.hasNext()) {
            State s = (State) i.next();
            id.put(s, id.size());
        }
        i = states.iterator();
        while (i.hasNext()) {
            State s = (State) i.next();
            b.append(" ").append(id.get(s));
            if (accept.contains(s))
                b.append(" [shape=doublecircle,label=\"" + s.name + "\"];");
            else
                b.append(" [shape=circle,label=\"" + s.name + "\"];");
            if (initials.contains(s)) {
                b.append(" in [shape=plaintext,label=\"" + "\n");
                b.append(" in -> ").append(id.get(s)).append(";\n");
            }
        }
        for (Map.Entry<StateSymbolPair, Set<State>> e: transitions.entrySet()) {
            StateSymbolPair ssp = e.getKey();
            for (State q: e.getValue()) {
                b.append(" ").append(id.get(ssp.state)).append(" -> ").append(id.
get(q));

                b.append(" [label=\""");
                char c = ssp.symbol.charValue();
                if (c >= '21' & c <= '7e' & c != '\\ ' & c != '\')
                    b.append(c);
                else {
                    b.append("\\u");
                    String s = Integer.toHexString((int) c);
                    if (s.length() < 4)
                        b.append("000").append(s);
                    else if (s.length() < 8)
                        b.append("00").append(s);
                    else if (s.length() < 10)
                        b.append("0").append(s);
                    else

```

Feb 23, 04 21:45

NFA.java

Page

```

                b.append(s);
            }
            b.append("\n");
        }
        return b.append("\n").toString();
    }
}

```

Feb 25, 04 13:05

NFALambda.java

Page 1/6

```

package dRegAut;
import java.util.*;
import java.io.*;

/**
 * Nondeterministic finite state automaton with
 *  tra
 nsitions.
 */
public class NFALambda
{
    /** Our representation of . */
    public static final Character LAMBDA = null; // tje null character isn't use
 d otherwise

    /** Set of {@link State} objects (Q). */
    public Set<State> states;

    /** The automaton alphabet (). */
    public Alphabet alphabet;

    /** Initial state (q<sub>0</sub>). Member of {@link #states}. */
    public State initial;

    /** Accept states (A). Subset of {@link #states}. */
    public Set<State> accept;

    /**
     * Transition function ().
     * This is a map from pairs of states and alphabet symbols to sets of states
     * ().
     *
     *  (  ())
     * 
 "; <sup>Q</sup>
     * We represent  by {@link #LAMBDA}.
     */
    public Map<StateSymbolPair, Set<State>> transitions;

    /**
     * Checks that this automaton is well-defined.
     * This method should be invoked after each <tt>NFALambda</tt> operation dur
 ing testing.
     * @return this automaton
     * @exception AutomatonNotWellDefinedException if this automaton is not well
 -defined
     */
    public NFALambda checkWellDefined() throws AutomatonNotWellDefinedException
    {
        if (states==null || alphabet==null || alphabet.symbols==null ||
            initial==null || accept==null || transitions==null)
            throw new AutomatonNotWellDefinedException("Invalid null pointer");
        for (Object o : states)
            if (!(o instanceof State))
                throw new AutomatonNotWellDefinedException("non-State object appears in s
 tate set");
        for (Object o : alphabet.symbols)
            if (!(o instanceof Character))
                throw new AutomatonNotWellDefinedException("non-Character object appears
 in alphabet");
        if (!states.contains(initial))
            throw new AutomatonNotWellDefinedException("the initial state is not in the state set
 ");
        if (!states.containsAll(accept))
            throw new AutomatonNotWellDefinedException("not all accept states are in the state
 set");
        for (State s : states) {
            ArrayList<Character> extended_symbols = new ArrayList<Character>(alp
 habet.symbols);
            extended_symbols.add(LAMBDA);
            for (Character c : extended_symbols) {
                Object o = transitions.get(new StateSymbolPair(s, c));
                if (o!=null) {
                    if (!(o instanceof Set))
                        throw new AutomatonNotWellDefinedException("non-Set object a
 ppears in transitions");
                    Set ps = (Set) o;
                    Iterator k = ps.iterator();
                    while (k.hasNext()) {
                        Object o2 = k.next();
                        if (!(o2 instanceof State))
                            throw new AutomatonNotWellDefinedException("non-State
 object appears in transitions");
                        if (!states.contains(o2))
                            throw new AutomatonNotWellDefinedException("there is a tr

```

Tuesday June 01, 2004

Feb 25, 04 13:05

NFALambda.java

Page 2/6

```

ansition to a state that is not in state set");
    }
}

    for (StateSymbolPair sp : transitions.keySet()) {
        if (!states.contains(sp.state))
            throw new AutomatonNotWellDefinedException("transitions refer to a state not
 in the state set");
        if (sp.symbol!=null & !alphabet.symbols.contains(sp.symbol))
            throw new AutomatonNotWellDefinedException("non-alphabet symbol appears
 in transitions");
    }
    return this;
}

/**
 * Constructs an uninitialized NFALambda.
 * <tt>states</tt> and <tt>accept</tt> are set to empty sets,
 * <tt>transitions</tt> is set to an empty map.
 */
public NFALambda()
{
    states = new HashSet<State>();
    accept = new HashSet<State>();
    transitions = new HashMap<StateSymbolPair, Set<State>>();
}

/** Clones this automaton. */
public Object clone()
{
    NFALambda f = new NFALambda();
    f.alphabet = alphabet;
    Map<State, State> m = new HashMap<State, State>(); // map from old states
 to new states
    for (State p : states) {
        State s = (State) p.clone();
        f.states.add(s);
        m.put(p, s);
        if (accept.contains(p))
            f.accept.add(s);
    }
    f.initial = m.get(initial);
    for (Map.Entry<StateSymbolPair, Set<State>> e : transitions.entrySet()) {
        StateSymbolPair ssp = e.getKey();
        Set<State> set = e.getValue();
        Set<State> ns = new HashSet<State>();
        for (State q : set)
            ns.add(m.get(q));
        f.transitions.put(new StateSymbolPair(m.get(ssp.state), ssp.symbol),
 ns);
    }
    return f;
}

/**
 * Constructs a new NFALambda that accepts the empty
 * language. [Martin, Th. 4.4]
 * @param a automaton alphabet
 */
public static NFALambda makeEmptyLanguage(Alphabet a)
{
    NFALambda f = new NFALambda();
    f.alphabet = a;
    State s = new State();
    f.states.add(s);
    f.initial = s;
    return f;
}

/**
 * Constructs a new NFALambda that accepts the language containing only the
 empty string. [Martin, Th. 4.4]
 * @param a automaton alphabet
 */
public static NFALambda makeEmptyString(Alphabet a)
{
    NFALambda f = new NFALambda();
    f.alphabet = a;
    State s = new State();
    f.states.add(s);
    f.initial = s;
    f.accept.add(s);
    return f;
}

/**
 * Constructs a new NFALambda that accepts the language containing only the
 given singleton string. [Martin, Th. 4.4]
 * @param a automaton alphabet
 * @param c the symbol in the singleton string
 */
public static NFALambda makeSingleton(Alphabet a, char c)
{

```

NFALambda.java

Feb 25, 04 13:05

NFALambda.java

Page

```

NFALambda f = new NFALambda();
State s = new State(), p = new State();

    f.alphabet = a;
    f.states.add(s);
    f.states.add(p);
    f.initial = s;
    f.accept.add(p);

    Set<State> set = new HashSet<State>();
    set.add(p);
    f.transitions.put(new StateSymbolPair(s, c), set);

    return f;
}

/**
 * Returns <a href="http://www.research.att.com/sw/tools/graphviz/"
_top">Graphviz Dot</a>
 * representation of this automaton.
 * (To convert a dot file to postscript, run '<tt>dot -Tps -o file.ps
 ot</tt>'.)
 * Lambdas are written as '<tt>%</tt>'.
 */
public String toDot()
{
    StringBuffer b = new StringBuffer("digraph Automaton {\n");
    b.append(" rankdir=LR;\n");
    Map<State, Integer> id = new HashMap<State, Integer>();
    for (State s : states)
        id.put(s, new Integer(id.size()));
    for (State s : states) {
        b.append(" ").append(id.get(s));
        if (accept.contains(s))
            b.append(" [shape=doublecircle,label=\"" + s.name + "\"];\n");
        else
            b.append(" [shape=plaintext,label=\"" + s.name + "\"];\n");
        if (s==initial) {
            b.append(" in [shape=plaintext,label=\"" + s.name + "\"];\n");
            b.append(" in-> ").append(id.get(s)).append(";\n");
        }
    }
    for (Map.Entry<StateSymbolPair, Set<State>> e : transitions.entrySet())
        for (State q : e.getValue()) {
            b.append(" ").append(id.get(ssp.state)).append("-> ").app
            b.append(" [label=\"" + s.name + "\"];\n");
            if (ssp.symbol==LAMBDA)
                b.append(" ");
            else {
                char c = ssp.symbol.charValue();
                if (c>=0x21 & c<=0x7e & c!='\'' & c!='%')
                    b.append(c);
                else {
                    b.append("\\u");
                    String s = Integer.toHexString((int) c);
                    if (s.length()==1)
                        b.append("000").append(s);
                    else if (s.length()==2)
                        b.append("00").append(s);
                    else if (s.length()==3)
                        b.append("0").append(s);
                    else
                        b.append(s);
                }
            }
            b.append(");\n");
        }
    return b.append("}\n").toString();
}

/** Returns number of states in this automaton. */
public int getNumberOfStates()
{
    return states.size();
}

/**
 * Looks up transitions in transition function.
 * @return (q, c),
 * the result is never null and it should never be modified
 */
public Set<State> delta(State q, Character c) throws IllegalArgumentException
{
    if (!alphabet.symbols.contains(c) & c!=LAMBDA)
        throw new IllegalArgumentException("symbol '" + c +
 " not in alphabet");
    Set<State> set = transitions.get(new StateSymbolPair(q, c));

```

NFALambda.java

Feb 25, 04 13:05 **NFALambda.java** Page 4/6

```

if (set==null)
    set = new HashSet<State>();
return set;
}

/**
 * Computes the -closure of the given state set. [Martin,
 * Def. 4.6]
 * @param set set of {@link State} objects
 * @return set of {@link State} objects (the input set is
 * unmodified)
 */
public Set<State> lambdaClosure(Set<State> set)
{
    Set<State> res = new HashSet<State>(); // contains the states that are i
n
    // the closure and have been visited
    Set<State> pending = new HashSet<State>(); // contains the states that a
re
    // in the closure but have not
    // yet been visited
    pending.addAll(set);
    while (!pending.isEmpty()) {
        State q = pending.iterator().next();
        pending.remove(q);
        res.add(q);
        Set<State> s = new HashSet<State>(delta(q, LAMBDA));
        s.removeAll(res);
        pending.addAll(s);
    }
    return res;
}

/**
 * Performs transitions in extended transition function. [Martin, Def. 4.5]
 * @return (q,s)
 * (the result is never null)
 * @exception IllegalArgumentException if a symbol in <tt>s</tt> is not in t
he alphabet
 */
public Set<State> deltaStar(State q, String s) throws IllegalArgumentExcepti
on
{
    Set<State> set = new HashSet<State>();
    set.add(q);
    set = lambdaClosure(set);
    for (int i=0; i<s.length(); i++) {
        Set<State> nset = new HashSet<State>();
        Iterator j = set.iterator();
        while (j.hasNext()) {
            State p = (State) j.next();
            nset.addAll(delta(p, new Character(s.charAt(i))));
        }
        set = lambdaClosure(nset);
    }
    return set;
}

/**
 * Runs the given string on this automaton. [Martin, Def. 4.5b]
 * @param s a string
 * @return true iff the string is accepted
 * @exception IllegalArgumentException if a symbol in <tt>s</tt> is not in t
he alphabet
 */
public boolean accepts(String s) throws IllegalArgumentException
{
    Set<State> set = deltaStar(initial, s);
    set.retainAll(accept);
    return !set.isEmpty();
}

/**
 * Adds a  transition
 * @param p from state
 * @param q to state
 */
public void addLambda(State p, State q)
{
    this.setTransition(p, LAMBDA, q);
}

/**
 * Constructs a new automaton whose language is the union of the language of
this automaton
 * and the language of the given automaton. [Martin, Th. 4.4]
 * The input automata are unmodified.
 * @exception IllegalArgumentException if the alphabets of <tt>f</tt> and th
is automaton are not the same
 */

```

Feb 25, 04 13:05 **NFALambda.java** Page 5/6

```

public NFALambda union(NFALambda f) throws IllegalArgumentException
{
    if (!f.alphabet.equals(this.alphabet))
        throw new IllegalArgumentException("alphabets are different");
    NFALambda f1 = (NFALambda) this.clone();
    NFALambda f2 = (NFALambda) f.clone();
    NFALambda n = new NFALambda();
    n.alphabet = alphabet;
    n.states.addAll(f1.states);
    n.states.addAll(f2.states);
    n.accept.addAll(f2.accept);
    n.accept.addAll(f1.accept);

    n.initial = new State();
    n.states.add(n.initial);
    n.addLambda(n.initial, f1.initial);
    n.addLambda(n.initial, f2.initial);

    n.transitions.putAll(f1.transitions);
    n.transitions.putAll(f2.transitions);

    return n;
}

/**
 * Constructs a new automaton whose language is the concatenation
 * of the language of this automaton and the language of the given
 * automaton. [Martin, Th. 4.4] The input automata are
 * unmodified.
 * @exception IllegalArgumentException if the alphabets of
 * <tt>f</tt> and this automaton are not the same
 */
public NFALambda concat(NFALambda f) throws IllegalArgumentException
{
    if (!f.alphabet.equals(this.alphabet))
        throw new IllegalArgumentException("alphabets are different");
    NFALambda f1 = (NFALambda) this.clone();
    NFALambda f2 = (NFALambda) f.clone();
    NFALambda n = new NFALambda();
    n.alphabet = alphabet;
    n.states.addAll(f1.states);
    n.states.addAll(f2.states);
    n.accept.addAll(f2.accept);
    n.initial = f1.initial;
    n.transitions.putAll(f1.transitions);
    n.transitions.putAll(f2.transitions);
    Iterator i = f1.accept.iterator();
    while (i.hasNext()) {
        State s = (State) i.next();
        n.addLambda(s, f2.initial);
    }
    return n;
}

/**
 * Constructs a new automaton whose language is the Kleene star of the langu
age of this automaton. [Martin, Th. 4.4]
 * The input automaton is unmodified.
 */
public NFALambda kleene()
{
    NFALambda n = (NFALambda) this.clone();
    State s = new State();
    n.states.add(s);
    n.addLambda(s, n.initial);
    n.initial = s;

    Iterator i = n.accept.iterator();
    while (i.hasNext()) {
        s = (State) i.next();
        n.addLambda(s, n.initial);
    }

    n.accept.clear();
    n.accept.add(n.initial);

    return n;
}

/**
 * Constructs an equivalent {@link NFA} by reducing all  transitions to other transitions. [Martin,
 * Th. 4.2] The input automaton is unmodified.
 */
public NFA removeLambdas()
{
    NFA f = new NFA();
    f.alphabet = alphabet;
    Map<State,State> m = new HashMap<State,State>(); // map from old states
to new states
    Iterator i = states.iterator();
    while (i.hasNext()) {
        State p = (State) i.next();

```

Feb 25, 04 13:05 **NFALambda.java** Page

```

    State s = (State) p.clone();
    f.states.add(s);
    m.put(p, s);
    if (accept.contains(p))
        f.accept.add(s);
}
f.initial = (State) m.get(initial);
Set<State> cl = new HashSet<State>();
cl.add(initial);
cl = lambdaClosure(cl);
cl.retainAll(accept);
if (!cl.isEmpty())
    f.accept.add(f.initial);
i = states.iterator();
while (i.hasNext()) { // this is quite naive but closely
// follows the mathematical definition
// in [Martin, Th. 4.2]
    State p = (State) i.next();
    Iterator j = alphabet.symbols.iterator();
    while (j.hasNext()) {
        Character c = (Character) j.next();
        Set<State> set = deltaStar(p, c.toString());
        Set<State> nset = new HashSet<State>();
        Iterator k = set.iterator();
        while (k.hasNext()) {
            State q = (State) k.next();
            nset.add(m.get(q));
        }
        f.transitions.put(new StateSymbolPair(m.get(p), c),
nset);
    }
}
return f;
}

/**
 * Sets a transition in the transition function.
 * (p,c)=p
 */
public void setTransition(State p, Character c, State q)
{
    StateSymbolPair ssp = new StateSymbolPair(p, c);
    Set<State> s = transitions.get(ssp);
    if (s==null) {
        s = new HashSet<State>();
        transitions.put(ssp, s);
    }
    s.add(q);
}
}

```