Universidade Federal de Santa Catarina.
Departamento de Informática de Estatística.

Disciplina: INE5415 – Teoria da Computação.
Nome: Gustavo Tarciso da Silva.
Data: 26/05/2016

Relatório de Implementação do Trabalho 1

O trabalho realizado tem como objetivo escrever um programa que transforme automatos finitos não determinísticos em automatos finitos deterministicos, partindo disso, foi escolhido fazer o trabalho em C++ por questão de afinidade com a linguagem.

As propriedades do automato estão definidas na classe automato, nela possui um estado inicial definido depois da geração do automato finito deterministico, um conjunto de estados utilizando uma estrutura de lista encadeada, para que possam ser adicionados novos estados com uma alocação dinâmica no final da lista. As funções de transição, e quem são os estados de aceitação, são conhecidos pelos próprios estados, onde o estado informa para o automato se ele é um estado de aceitação ou não.

Para converter o automato finito não determinístico em determinístico sem transições por epsilon, o programa percorre a lista de estados, para cada estado até o final da lista, ele percorre suas transições e verifica se ela é igual a algum estado já existente, caso não seja, o automato cria um novo estado cujo o nome é o nome da transição, verifica quais estados do automato não determinístico tem seu nome contido no nome do novo estado, e concatena as transições desses estados por cada simbolo para criar as transições para o novo estado.

Já para converter o automato finito não determinístico com transições por epsilon, em um automato finito determinístico, o procedimento é parecido com o descrito acima, porém, com alguns ajustes. Primeiramente se extrai os epsilon fecho de cada estado do automato não determinístico, depois, cria-se uma lista auxiliar vazia para colocar os estados gerados, e preenche o primeiro estado com a criação de um novo estado a partir do estado inicial do automato finito não determinístico, onde o nome do estado inicial se mantem, e suas transições são geradas a partir do epsilon fecho dele, e conforme é feito para o automato sem transições por epsilon, é verificado quais estados tem o nome contido no epsilon fecho, e concatena-se as transições por cada simbolo. Após adicionar o primeiro estado, o programa percorre a lista auxiliar, adicionando de forma dinâmica os estados na lista auxiliar, e gerando novos estados de forma semelhante a do automato sem transições por epsilon, porém, ele verifica quais estados tem o nome contido no nome do novo estado, verifica quais são os estados contidos nos epsilon fecho de cada estado contido no nome, e gera as transições pelos simbolos concatenando as transições dos estados nos epsilon fecho. Para finalizar, a lista é transformada na lista auxiliar.

Para gerar o epsilon fecho, primeiramente ele assume o nome do estado, e depois é concatenado o nome dos estados contidos nas transições por epsilon, e em seguida verificado as transições por epsilon do estado cujo nome foi concatenado para verificar se ele transita por epsilon, e por onde ele transita por epsilon, para assim ir adicionando ao fecho até que o processo termine.

Para a conversão do automato finito por transição epsilon foi utilizado uma lista auxiliar para que não sobreescrevesse os estados do automato finito não determinístico, uma vez que eles são necessários para gerar o novo automato, e não são utilizados no novo automato determinístico, e por isso, ela é transformada depois na lista auxiliar.

Pseudo código da conversão do automato finito não determinístico sem transições por epsilon:

```
list states; // lista de estados do automato
list states_heads; // lista de nomes dos estados do automato
list finalStates; // lista com os estados de aceitação do automato não deterministico
int nstatesFNA; // número de estados do FNA - estático
int nstatesFinal; // número de estados finais
int nsymbol; // número de simbolos
int nstates; // número de estados - aumenta conforme são adicionados novos estados
generateDFA() {
	for states to end
		transitions = state.getTransitions();
		for i = 0 to nsymbol
			int cont = 0;
			if transitions[i] == "-"
				cont = 1;
			for j = 0 to nstates
				if transitions[i] == states_heads[j]
					cont++;
		if cont == 0
			head = transitions[i] // nome do novo estado
			states_heads[nstates] = head
			nstates++;
			newTransitions[nsymbols];
			for j = 0 to nsymbols
				for k = 0 to nstatesFNA
					if head find states_heads[k]
						transitions = states[k].getTransitions();
						newTransitions[j] += transitions[j];
			states.push_back(State(head, newTransitions));

	for states to end
		for i = 0 to nstatesFinal
			if states.getHead() find finalStates[i]
				state.setFinal(true);
}
```

Pseudo código da conversão do automato finito não determinístico com transições por epsilon:

```
list states; // lista de estados do automato
list states_heads; // lista de nomes dos estados do automato
list epsilonClosure; // lista com os epsilon fecho de cada estado
list finalStates; // lista de estados de aceitação
int nstatesFNA; // número de estados do FNA - estático
int nstatesFinal; // número de estados finais
int nsymbol; // número de simbolos
int nstates; // número de estados - aumenta conforme são adicionados novos estados
generateDFA() {
	list statesDFA;
	head = epsilonClosure[0];
	for j = 0 to nsymbols
				for k = 0 to nstatesFNA
					if epsilonClosure[0] find states[k]
						transitions = states[k].getTransitions();
						newTransitions[j] += transitions[j];
	statesDFA.push_back(State(head, newTransitions));
	for states to end
		transitions = state.getTransitions();
		for i = 0 to nsymbol
			int cont = 0;
			if transitions[i] == "-"
				cont = 1;
			for j = 0 to nstates
				if transitions[i] == states_heads[j]
					cont++;
		if cont == 0
			head = transitions[i] // nome do novo estado
			states_heads[nstates] = head
			nstates++;
			newTransitions[nsymbols];
			auxiliar;
			for j = 0 to nstatesFNA
				if head find states_heads[k]
					auxiliar += epsilonClosure[k];
			for j = 0 to nsymbols
				for k = 0 to nstatesFNA
					if auxiliar find states_heads[k]
						transitions = states[k].getTransitions();
						newTransitions[j] += transitions[j];
			statesDFA.push_back(State(head, newTransitions));
	states = statesDFA;
	for states to end
		for i = 0 to nstatesFinal
			if states.getHead() find finalStates[i]
				state.setFinal(true);
}
```

Pseudo código da geração do epsilon fecho:

```
int nstatesFNA; // número de estados do FNA – estático
int nsymbols; // número de simbolos
list states; // lista de estados do automato
generateEpsilonClosure() {
        for i = 0 to nstatesFNA
                epsilonClosure[i] += states[i].getHead();
                for states to nstatesFNA
                        transitions = state.getTransitions();
                        if transitions[nsymbols] == "-"
                                return;
                        for j = 0 to nstatesFNA
                                if transitions[nsymbols] find states[j].getHead();
                                        epsilonClosure[i] += states[j].getHead();
}
```