COMP235 : Logic and Computation

Supplementary notes on Kleene's Theorem

This material is not covered in the hand-out booklet.

1 Transition Graphs

A *transition graph* over the finite alphabet X is a directed graph T such that:

- each edge of T is labelled with a regular expression over X
- there is a unique starting vertex (= state)
- some vertices may be accepting states

The *language accepted by the transition graph* T, $L(T)$, is all strings over X $w = w_1 w_2 \cdots w_k$ where each $w_i \in L(R_i)$, R_i the label on edge e_i where $e_1, e_2, \cdots, e_k$ is a path from the starting state through to an accepting state.

(If T is an NFSA, this is the same as our earlier definition.)

Transition graphs T_1 and T_2 are *equivalent* if $L(T_1) = L(T_2)$.

An *elementary transition graph* (ETG) is a transition graph which has two states only, one starting and the other accepting, and at most one edge between them.

2 Finding an NFSA accepting R, a regular language

Given an ETG T, with single edge having label the regular expression R over X, we can convert T to an equivalent NFSA using the following step repeatedly:

- replace an edge from q to r labelled $R + S$ by two parallel edges from q to r, one labelled R and the other labelled S
- replace an edge from q to r labelled RS by an edge from q to the new state t labelled R and an edge from t to r labelled S
- replace an edge from q to r labelled R^* by an edge from q to the new state t labelled λ (= empty word), a loop on t labelled R, and an edge from t to r labelled λ

This can only be done finitely many times before all edge labels are single characters in X, or the empty word. By then we have a NFSA M such that $L(M) = R$, since each TG along the way is equivalent to the ETG we started with, which clearly accepts R.

3 Given M an NFSA: find a regular expression R such that $L(M) = R$,

Given an NFSA M, first of all:

1. introduce a new starting state q_0 and a new accepting state q_F
2. join q_0 to each starting state, and each accepting state to q_F, and label these new edges λ

Throughout the following process:

- eliminate parallel edges: if there are two edges from state q to state r, labelled R and S, combine as a single edge from q to r labelled $R + S$.

Eliminate states successively, other than q_0 and q_F. Eliminate state q as follows:

1. if there is a loop on q labelled R, and an edge from q to r with label S, replace that label by R^*S; do this for each edge coming out of q
2. once this is done, delete the loop on q
3. if there is an edge (p, q) from state p to state q, with label R, and an edge (q, r) with label S, add in an edge (p, r) with label RS
4. once this is done wherever possible, delete q and all edges coming in to or going out of it

This process stops when all states are deleted except q_0 and q_F, and there is at most one edge (q_0, q_F) remaining, with label R say. Each state deletion does not alter the language accepted, so $L(M) = R$. (If no edges remain at the end, it means M accepts the empty language (ie no strings are accepted).

Any comments or queries: please contact Tim Stokes (stokes@math.waikato.ac.nz).