재구성 가능한 메쉬에서 결정적 유한 자동장치 문제에 대한 상수시간 알고리즘

김 영 학†

요 약

유한 자동장치는 이산적인 입력과 출력을 갖는 시스템을 표현하기 위한 수학적인 모델이다. 유한 자동장치는 텍스트 편집기, 구문 분석기, 스위칭 회로와 같은 문제들의 해결을 위해 유용한 설계 도구로 이용된다. 본 논문에서는 길이 n인 입력 문자열과 m개의 상태를 갖는 결정적 유한 자동장치가 주어질 때, $\lceil nm/2 \rceil \times 2m$ 크기의 재구성 가능한 메쉬에 유한 자동장치의 전이상태를 표현하고 상수시간에 입력 문자열의 승인 여부를 결정하는 병렬 알고리즘을 제안한다.

A Constant Time Algorithm for Deterministic Finite Automata Problem on a Reconfigurable Mesh

Young-Hak Kim†

ABSTRACT

Finite automaton is a mathematical model to represent a system with discrete inputs and outputs. Finite automata are a useful tool for solving problems such as text editor, lexical analyzer, and switching circuit. In this paper, given a deterministic finite automaton of an input string of length n and m states, we propose a constant time parallel algorithm that represents the transition states of finite automata and determines the acceptance of an input string on a reconfigurable mesh of size $\lceil nm/2 \rceil \times 2m$.

1. 서 론

최근에 메쉬 구조를 갖는 병렬 컴퓨터에서 병렬 알고리즘을 설계할 경우에 발생되는 긴 거리의 통신 지름(diameter)을 줄이기 위해, 각 처리기(processor)가 알고리즘 수행 동안 동적으로 스위치를 변경하여 처리기들 간의 통신 지름을 '1'로 하는 재구성 가능한 버스 시스템(reconfigurable bus system)을 갖는 처리기 배열들이 등장하게 되었다. 이러한 재구성 가능한 버스 개념을 적용한 대표적인 병렬 컴퓨터 구조들은 RMESH[1], RN[2], PARBS[3] 등이 있으며, 이들과 유사한 구조를 가지면서 각 처리기에 시프트 스위치 기능을 추가한 REBSIS[4], CRAP[5] 등의 컴퓨터 모델이 다수의 문헌에서 제안되었다. 또한 이러한 컴퓨터 모델을 이용하여 이론적인 측면에서 정렬, 트리, 이미지 인식, 그래프, 행렬 문제들에 대한 효율적인 병렬 알고리즘들이 제안되었다[6, 7, 8, 9].

본 논문에서는 결정적 유한 자동장치 문제(deterministic finite automata)를 해결하기 위해서 재구성 가능한 버스를 갖는 처리기 배열들 중에서 가장 일반적이고 널리 알려진 RMESH(reconfigurable mesh)를

※ 본 연구는 1999년도 금오공과대학교 학술연구비 지원에 의해서 연구되었음.
† 종신회원 : 금오공과대학교 컴퓨터공학부 교수

계산 모델로 사용한다. 유한 자동장치는 이산적인 입력과 출력을 갖는 시스템을 표현하기 위한 수학적인 모델이다. 이러한 장치에서 모델된 시스템은 유한한 상태(state)를 가지며 시스템의 행위는 현재의 상태와 현재의 입력에 따라 결정된다. 전산학 분야에서 유한 자동장치는 텍스트 편집기, 구문 분석기, 스위칭 회로와 같은 문제들의 해결을 위해 유용한 설계 도구로 이용된다[10].

Chomsky는 형식 언어의 표현 부류를 4가지로 구분하였다. 형식-0의 부류는 비제한(unrestricted) 언어이고, 형식-1의 부류는 문맥 인식(context-sensitive) 언어이다. 형식-2의 부류는 문맥 자유(context-free) 언어이고, 형식-3의 부류는 정규 언어이다. 이들 부류 중에서 형식-3의 부류인 정규 언어는 결정적 유한 자동장치에 의해서 인식될 수 있음이 증명되었다[10]. 따라서 본 논문에서 고려한 RMESH 상에서 유한 자동장치 알고리즘은 형식 3의 부류에 속하는 모든 정규언어들을 인식할 수 있다.

현재 재구성 가능한 버스 시스템을 갖는 처리기 배열에서 이러한 언어의 부류에 대한 인식 문제는 활발히 연구되지 않고 있다. 이러한 언어의 인식 부류는 주어진 입력 문자와 현재 상태에 따라 다음 상태가 결정되는 것과 같은 문제 자체에 내재된 고유한 순차적인 특성으로 인하여, 기존의 병렬 컴퓨터 모델 상에서 효율적으로 수행되는 알고리즘의 설계가 좀처럼 쉽지 않다. Jain 등은 길이 n의 입력 문자열이 주어질 때 $O(n^3)$ 크기를 갖는 3차원 PARBS(RMESH와 유사) 상에서 $O(\log n)$ 시간에 형식의 2의 부류인 문맥 자유 언어를 인식하는 알고리즘을 제안하였다[11]. 이동훈은 길이 n의 입력 문자열과 m개의 상태로 구성된 결정적 유한 자동장치가 주어질 때 3차원 $n\times m\times m$ PARBS에서 상수시간에 이 문제를 해결하는 알고리즘을 제안하였다[12].

그러나 불행하게도 이동훈의 알고리즘은 PARBS에서 각 처리기가 유한한 기억장소를 갖는다는 특성을 위배하였고, 또한 3차원 형태의 재구성 가능한 처리기 배열은 2차원의 경우에 비해서 복잡하고 현실성이 떨어진다. 실제 3차원 $n\times m\times m$ 크기를 갖는 재구성 가능한 메쉬에서 수행되는 알고리즘들은 2차원 $2nm\times 2nm$의 크기의 RMESH에서 같은 시간에 수행될 수 있음이 증명되었다[13]. 이러한 경우에 2차원 재구성 가능한 메쉬에서 수행되는 알고리즘의 처리기의 수는 $O(n^2m^2)$개가 된다.

따라서 본 논문에서는 현재 재구성 가능한 메쉬에서 연구가 되고 있지 않은 결정적 유한 자동장치 문제를 고려하며, 길이 n인 입력 문자열과 m개의 상태를 갖는 결정적 유한 자동장치가 주어질 때 2차원 $\lceil nm/2\rceil\times 2m$ RMESH에 유한 자동장치의 전이상태를 표현하고 상수시간에 입력 문자열의 승인 여부를 결정하는 효율적인 병렬 알고리즘을 제안한다. 또한 이러한 알고리즘은 형식 3의 부류인 정규언어의 인식 문제에 그대로 적용될 수 있음을 설명한다.

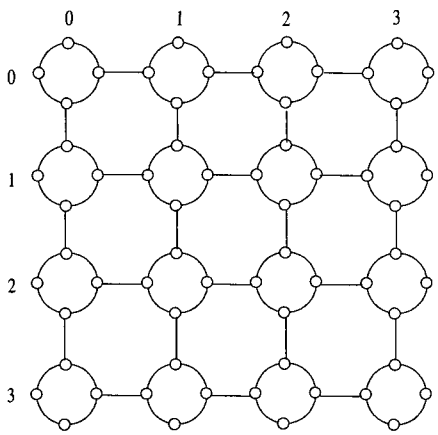
본 논문의 2장에서는 RMESH 구조와 결정적 유한 자동장치 문제를 간략히 설명하고, 3장에서는 2차원 RMESH 상에 결정적 유한 자동장치의 전이상태를 표현하는 방법과 입력 문자열의 승인 여부를 결정하는 상세한 알고리즘을 설명한다. 마지막으로 4장에서는 결론과 앞으로 추가적으로 연구해야할 과제를 설명한다.

2. 계산 모델 및 문제 정의

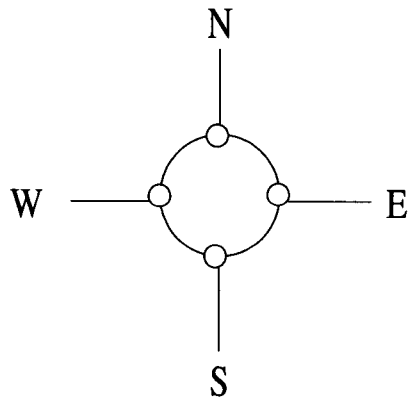
2.1 계산 모델

먼저 재구성 가능한 버스 시스템을 갖는 처리기 배열들 중에서 현재 가장 널리 알려진 RMESH의 기본적인 기능을 알아본다[1]. 2차원 $m\times n$ RMESH는 m개의 행과 n개의 열에서 메쉬 형태로 연결된 mn개의 처리기 배열로 구성된다. (그림 1)은 2차원 4×4 RMESH의 예를 보여준다. 각 처리기는 4개의 포트(W, E, N, S)를 가지며 (그림 2)에 표기 예가 보여졌다. $PE(i,j)$를 i번째 행과 j번째 열에 위치한 처리기로 정의한다. 처리기 $PE(i,j)$는 자신의 인덱스 (i,j)를 알고 있고 외부 경계에 위치한 처리기를 제외한 모든 처리기들은 4개의 인접한 처리기에 연결된다.

각 처리기는 기본적인 산술연산과 논리연산을 수행할 수 있으며 $\Theta(\log n)$ 비트의 크기를 갖는 $O(1)$개의 기억 장소를 갖는다. 알고리즘 수행 동안 각 처리기는 4개의 포트를 동적으로 연결하여 다양한 형태의 버스를 구성할 수 있다. 예로서 $\{WE, NS\}$는 W와 E 포트를 연결하고 N과 S 포트를 연결함을 의미한다. 만일 모든 처리기의 포트 설정을 $\{WE, N, S\}$로 설정하면 각 행에서 모든 처리기들은 같은 버스에 의해서 연결된다. 각 처리기에 대한 다양한 형태의 포트 설정은 Miller 등[1]의 결과를 참조하길 바란다.



(그림 1) 4×4 RMESH



(그림 2) 포트

RMESH에서 버스의 폭은 $\Theta(logn)$ 비트를 갖는다. 처리기들간의 자료 전송은 버스를 사용하여 이루어지고, 충돌이 발생하지 않는다면 다수의 처리기가 자료를 동시에 서로 다른 버스에 전송할 수 있다. 또한 같은 버스에 연결된 처리기들은 동시에 자료를 읽기가 가능하며, 같은 버스에 연결된 처리기들에 대한 자료 전송은 $O(1)$ 시간에 이루어진다. 이러한 형태의 RMESH 구조는 규칙적인 특성 때문에 VLSI 구현에 적합하며, SIMD 형식으로 작동한다.

2.2 문제 정의

유한 자동장치는 유한한 상태를 가지며 한 상태에서 다른 상태로의 전이는 현재 상태의 정보와 입력 심벌에 따라 결정된다. 각 입력 심벌에 대하여 정확하게 하나의 전이 상태만을 갖는 것을 결정적 유한 자동장치라 한다. 하나의 결정적(deterministic) 유한 자동장치 M={Q, Σ, q_0, δ, F}은 다섯 개의 객체로 구성되며 각 객체에 대한 구성은 다음과 같다. ① 공집합이 아닌 유한 상태의 집합 $Q=\{q_0, q_1, \cdots, q_{m-1}\}$. 여기서 q_i는 상태를 의미하고 m은 상태의 수를 의미한다. ② 입력 문자열에서 허용되는 심벌들의 공집합이 아닌 유한 집합 Σ. 집합 Σ는 M의 입력 알파벳이라 한다. ③ 하나의 지정된 상태 $q_0 \in Q$는 초기상태라 한다. ④ 모든 상태 $q_i \in Q$와 모든 $a \in \Sigma$에 대해서 정의된 전이 함수 $\delta(q_i, a)$. 여기서 $\delta: Q \times \Sigma \rightarrow Q$이다. ⑤ 종결 상태의 집합 $F \subseteq Q$. F를 승인 상태의 집합이라고 한다.

또한 정규 언어의 인식 문제는 임의의 입력 문자열이 주어지면 이 문자열이 정규 문법에 의해서 생성될 수 있는지를 결정한다. 정규 언어의 인식 문제를 좀더 구체적으로 설명하기 위해서 Σ를 입력 문자열에서 사용될 수 있는 문자들의 유한 집합으로 정의하고, Σ^*는 Σ에 포함된 문자들로 구성된 문자열로 정의한다. 정규 문법 G와 임의의 입력 문자열 $\sigma \in \Sigma^*$가 주어질 때, $\sigma \in L(G)$ 인지를 결정하는 문제를 정규 언어의 인식이라 한다. 여기서 $L(G)$는 문법 G에 의해 생성될 수 있는 모든 문자열의 집합을 의미한다.

Ullman 등[10]은 주어진 문법 G에 대해서 $L(M)=L(G)$인 결정적 유한 자동장치 M이 존재함을 증명하였다. 이러한 정리에 의해서 $\sigma \in L(G)$를 결정하는 문제는 $L(M)=L(G)$인 결정적 유한 자동장치 M을 구현하는 문제로 축약될 수 있다. 따라서 본 논문에서 고려한 결정적 유한 자동장치 M에 대한 2차원 RMESH 상의 알고리즘은 정규 언어의 인식 문제에 그대로 적용될 수 있다.

일반적으로 유한 자동장치는 상태 다이어그램 혹은 상태표에 의해서 표현될 수 있다. 예로서 $Q=\{q_0, q_1, q_2, q_3\}$, $\Sigma=\{0,1\}$, $F=\{q_0\}$이고 전이 함수 δ는 <표 1>과 같이 표현될 수 있다. 이러한 유한 자동장치에 입력 문자열 σ=110101이 주어질 때, 전이 상태는 $q_0 \rightarrow q_1 \rightarrow q_0 \rightarrow q_2 \rightarrow q_3 \rightarrow q_1 \rightarrow q_0$가 된다. 따라서 주어진 입력 문자열 σ는 $L(M)$에 포함되므로 유한 자동장치 M에 의해서 인식될 수 있다. 본 논문에서는 RMESH 상에 이러한 전이 상태의 경로를 표현함으로서 알고리즘을 구현한다.

〈표 1〉 전이 함수의 예

상태 \ 입력	0	1
q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

3. 알고리즘

3.1 {0,1}로 구성된 입력 문자열의 인식

이제 위에서 설명한 유한 자동장치를 RMESH 상에 표현하는 방법과 주어진 입력 문자열의 승인 여부를 결정하는 방법을 설명한다. 주어진 유한 자동장치에서 상태의 수를 $|Q|=m$ 이라 한다. 그러면 이러한 상태들은 0에서 $m-1$ 사이의 정수 값으로 표현될 수 있다. 일반성을 상실하지 않고 입력 문자열의 길이가 $|\sigma|=n$ 이라 하고 n은 짝수 값을 갖는다고 가정한다. 또한 입력 알파벳 Σ는 {0,1}로 제한하고 $m \geq 2$이라 가정한다. 후에 이러한 결과를 일반성을 갖도록 확장한다.

먼저 2차원 $m \times m$ 크기를 갖는 RMESH를 고려한다. 초기에 첫 번째와 두 번째 열의 처리기들에 <표 1>과 같은 상태 표가 저장되어 있다고 가정한다. 즉 첫 번째 열의 처리기 $PE(j,0)$ $0 \leq j \leq m-1$이 $\delta(q_j,0)=q_k$에 대한 전이 함수의 인덱스 값 k를 가지며, 두 번째 열의 처리기 $PE(j,1)$ $0 \leq j \leq m-1$이 $\delta(q_j,1)=q_r$에 대한 전이 함수의 인덱스 값 r을 갖는다. 이러한 인덱스 값에 대한 적재는 전적으로 RMESH의 입출력 부분에 의존하기 때문에 본 논문에서는 인덱스 적재 시간은 고려하지 않는다.

본 논문의 기본적인 아이디어는 다음과 같다. 입력 문자열 σ=110101이 주어질 때, <표 1>에 의한 전이 상태는 $q_0 \rightarrow q_1 \rightarrow q_0 \rightarrow q_2 \rightarrow q_3 \rightarrow q_1 \rightarrow q_0$이 되며 6개의 연속적인 $m(=4) \times m$ 크기를 갖는 서브메쉬에서 각각 (q_0,q_1), (q_1,q_0), (q_0,q_2), (q_2,q_3), (q_3,q_1), (q_1,q_0)의 경로를 연결하는 버스를 구성한다. 그러면 첫 번째 서브메쉬에서 q_0를 갖는 처리기에 '*'를 전송하면 마지막 서브메쉬에서 이 값을 받는 처리기가 종결 상태일 경우 주어진 입력 문자열은 유한 자동장치에 의해서 인식될 수 있다. 따라서 길이가 n인 입력 문자열 $\sigma = x_0x_1 \cdots x_{n-1}$이 주어질 경우 n개의 $m \times m$ 크기의 서브메쉬가 필요하며, i번째 서브메쉬에서 입력 문자 x_i에 대한 전이 함수를 버스 경로로 구성할 수 있다.

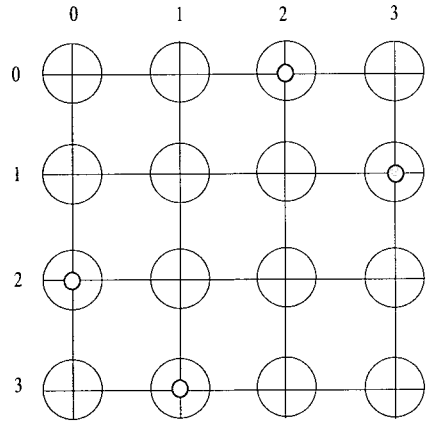
이러한 아이디어의 구현을 위해서 $m \times m$ 크기의 서브메쉬에서 입력 문자 Σ={0,1}에 대한 전위 함수를 어떻게 표현할 수 있는지를 다음 보조정리에서 보인다.

[보조정리 1] 입력 문자 x에 대한 전위 함수 $\delta(q_i,x)=q_k$ $0 \leq i \leq m-1$는 $m \times m$ 크기의 RMESH 상에서 $O(1)$ 시간에 결정될 수 있다. x는 0 혹은 1의 값을 갖는다.

[증 명] 초기의 가정에 의해서 첫 번째와 두 번째 열에는 각각 입력 문자 0과 1에 대한 상태 표가 저장되어 있다. 먼저 입력 문자 x가 0인 경우를 고려한다. 처리기 $PE(i,0)$ $0 \leq i \leq m-1$에 저장된 $\delta(q_i,0)$에 대한 전위 함수 값은 처리기 $PE(i,i)$를 경유하여 첫 번째 행의 처리기 $PE(0,i)$에 $O(1)$ 시간에 전송된다. 다음에 처리기 $PE(0,i)$ $0 \leq i \leq m-1$가 i번째 열의 모든 처리기에 전송 받은 값, 즉 $\delta(q_i,0)$의 전위 함수 값을 전송한다. 각 처리기 $PE(j,i)$ $0 \leq j \leq m-1$는 전송 받은 전위 함수의 값이 인덱스 j와 같은 경우는 {WENS}, 그렇지 않은 경우는 {WE, NS}로 포트를 연결한다. 이러한 연결에 의한 포트 설정 예는 (그림 3)을 참조한다. 그러면 첫 번째 행에서의 각 처리기 $PE(0,i)$가 $\delta(q_i,0)=q_k$에 대한 전위 함수에서 q_i를 가지며, 가장 오른쪽 열의 처리기 $PE(m-1,k)$가 q_i의 다음 전위 상태인 q_k의 값을 갖는다. 입력 문자 x가 1인 경우는 두 번째 열에 저장된 상태 정보를 이용하여 위에서 설명한 것과 같은 방법을 적용한다. 이러한 RMESH에서 한 순간에 한 개의 상태 q_i(첫 번째 행)와 이에 대한 전위 함수 값 q_k(맨 오른쪽 열)만을 허용하므로 자료 전송시 충돌이 발생하지 않으며, $O(1)$ 시간에 수행된다. □

이제 보조정리 1의 결과를 이용하여 m개의 상태와 n개의 입력 문자열을 갖는 유한 자동장치가 주어질 경우, 주어진 유한 자동장치를 RMESH에 임베딩한 후에 입력 문자열을 상수시간에 인식하는 방법을 설명한다. 먼저 주어진 유한 자동장치를 임베딩하기 위해서 2차원 $\lceil nm/2 \rceil \times 2m$ 크기를 갖는 RMESH를 고려한다. 이러한 크기를 갖는 RMESH는 (그림 4)에서 보여진 것과 같이 n개의 m×m 크기를 갖는 서브블록으로 분할될 수 있으며 각 서브블록의 명칭은 행의 순에 의해서 BK(0), BK(1), BK(2), …, BK(n-1)로 정의한다. 초

기에 BK(0)의 첫 번째와 두 번째 열에는 각각 <표 1>과 같은 전위 함수의 표가 저장되어 있다고 가정한다

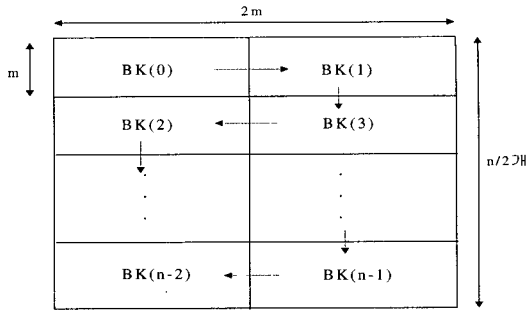


(그림 3) x=0인 경우에 각 처리기의 스위치 구성

다음 보조정리는 모든 블록 BK(i) 0≤i≤n-1가 BK(0)와 같은 전위 함수의 표를 가질 수 있음을 보인다.

[보조정리 2] 초기에 *BK(0)*의 첫 번째와 두 번째 열에 저장된 전위 함수의 표는 모든 서브블록 *BK(i) 1≤i≤n-1*에 *O(1)* 시간에 복사될 수 있다.

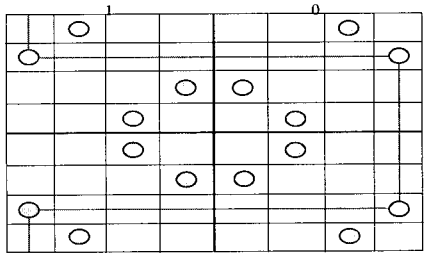
[증 명] 초기에 입력 문자 0과 1에 대한 전위 함수의 값이 *BK(0)*의 첫 번째와 두 번째 열에 저장되어있다. *BK(i).PE(j,k) 0≤i≤n-1, 0≤j,k≤m-1*를 *i*번째 서브블록에서 *j*번째 행과 *k*번째 열에 위치한 처리기로 정의한다. 먼저 *BK(0).PE(j,0)*와 *BK(0).PE(j,1)*에 저장된 전이 상태 값을 *BK(0).PE(j,j)*에 수집한다. 다음에 행의 버스를 이용하여 *BK(0).PE(j,j)*에 저장된 값을 *BK(1).PE(j,j)*에 전송하고, 다시 열의 버스를 이용하여 *BK(0).PE(j,j)*와 *BK(1).PE(j,j)*에 저장된 값을 각각 짝수와 홀수 블록들의 처리기 *BK(i).PE(j,j)*에 전송한다. 이제 각 블록에서 행의 버스를 이용하여 *BK(i).PE(j,j)*에 저장된 값 중에서 입력 문자 0의 상태 값은 *BK(i).PE(j,0)*에, 입력 문자 1의 상태 값은 *BK(i).PE(j,1)*에 전송한다. 그러면 모든 서브블록 *BK(i)*는 *BK(0)*에 저장된 것과 같은 전이 함수의 상태 값을 갖는다. 이러한 과정은 행과 열의 버스 상에서 두 개의 자료 값들만이 전송되므로 *O(1)* 시간에 수행된다. □



(그림 4) 크기 $m \times m$의 n개의 서브메쉬로 분할된 RMESH의 예

본 논문의 알고리즘에서 길이 n인 입력 문자열 $\sigma = x_0 x_1 \cdots x_{n-1}$의 각 x_i는 (그림 4)와 같은 RMESH에서 i번째 서브블록의 첫 번째 처리기인 *BK(i).PE(0,0)*에 저장되어 있다고 가정한다. 여기서 x_i는 0 혹은 1을 갖는다. 이제 유한 자동장치의 전이 상태의 값과 입력 문자열의 정보를 이용하여 (그림 4)와 같은 크기를 갖는 RMESH상에서 *O(1)* 시간에 주어진 문자열이 승인 가능한지를 판단하는 알고리즘을 설명한다. 먼저 독자들의 이해를 돕기 위해서 알고리즘의 개요를 한 예로서 설명하고 후에 각 단계별로 상세하게 기술한다.

<표 1>과 같은 전이 상태를 갖는 유한 자동장치와 입력 문자열 σ=1010이 주어졌을 경우의 예를 고려해 본다. 그러면 입력 문자열의 길이는 4이고 <표 1>에서 상태의 수는 4이므로 8×8 크기를 갖는 RMESH를 필요로 한다. (그림 5)에서 보여진 것과 같이 8×8 크기의 RMESH는 4개의 4×4 서브블록으로 구성된다. 각 서브블록 *BK(0), BK(1), BK(2), BK(3)*의 첫 번째 처리기에는 각각 입력 문자열 1, 0, 1, 0이 저장되어 있다. 또한 각 서브블록의 첫 번째와 두 번째 열에는 <표 1>의 전이 상태 값들이 저장되어 있다.



(그림 5) σ =1010에 대한 버스 구성

(그림 4)에서 점선 화살표로 도식한 경로를 따라 버스를 구성하기 위해서 위의 예에서 *BK(2)*의 첫 번째 처리기에 저장된 값 1과 *BK(3)*의 첫 번째 처리기에 저장된 값 0을 서로 교체한다. 각 서브블록의 첫 번째 처리기에 저장된 입력 문자는 간단한 라우팅에 의해서 각 서브블록의 모든 처리기들에 *O(1)* 시간에 전송될 수 있다. 이제 각 서브블록 *BK(i)*에서는 보조정리 1에서 설명한 연결 패턴을 생성할 수 있다. (그림 5)의 예에서 *BK(0)*와 *BK(3)*은 보조정리 1에서 설명한 방법을 그대로 따르고, *BK(1)*과 *BK(2)*는 보조정리 1에서 첫 번째 행을 첫 번째 열로 생각하여 연결 패턴을 설정한다. 또한 *BK(0)*와 *BK(3)*의 마지막 행의 처리기들은 아래쪽 서브블록에 버스가 연결되지 않도록 연결 패턴 {WENS}를 {WEN,S}로 재조정한다. (그림 5)에 최종적인 결과가 보여졌다.

(그림 5)는 4개의 서브블록을 하나의 경로를 갖는 버스로 구성할 수 있음을 보인다. 예로서 유한 자동장치의 시작 상태가 q_0라고 하면 $q_0 \rightarrow q_1 \rightarrow q_3 \rightarrow q_2 \rightarrow q_0$의 전이 상태를 하나의 버스 경로로 구성해 준다. 따라서 q_0의 상태를 갖는 *BK(0)*의 처리기 *PE(0,0)*에서 '1' 값을 전송하였을 때 *BK(2)*의 마지막 열의 처리기가 이 값의 수신 여부에 따라 입력 문자열의 승인 여부를 결정한다. 즉 서브블록 *BK(0), BK(1), BK(3), BK(2)*에서 각각 (q_0, q_1), (q_1, q_3), (q_3, q_2), (q_2, q_0)의 전이 함수 경로를 생성해 준다. 예로써 <표 1>에서 주어진 유한 자동장치의 종결 상태가 q_0라고 가정하면 입력 문자열 1010은 유한 자동장치에 의해서 승인된다.

이제 위의 예에서 개략적으로 설명한 알고리즘을 각 단계별로 상세하게 기술한다. 이미 설명한 것과 같이 초기의 전이 상태의 값은 *BK(0)*의 첫 번째 열과 두 번째 열에 저장되어 있고, 입력 문자열 $\sigma = x_0x_1\cdots x_{n-1}$의 각 x_i가 *BK(i)*의 처리기 *PE(0,0)*에 저장되어 있다. 다음에 기술된 알고리즘에서 *RM(k)* $0 \le k \le \lceil n/2 \rceil - 1$는 $\lfloor j/2 \rfloor = k$ 인 *BK(j)*들로 구성된 서브블록을 의미한다.

[알고리즘 : FA_Recognition]

단계 1. *BK(0)*에 저장된 전이 상태 값을 모든 *BK(i)*에 복사한다.

단계 2. 각 *RM(k)* $0 \le k \le \lceil n/2 \rceil - 1$에 대해서 k가 홀수인 경우 *BK(2k)*의 *PE(0,0)*에 저장된 입력 문자 *BK(2k+1)*의 *PE(0,0)*에 저장된 입력 문자를 서로 교환한다.

단계 3. 각 *BK(i)* $0 \le i \le n-1$의 *PE(0,0)*에 저장된 입력 문자의 값을 *BK(i)*의 첫 번째 행과 열의 처리기들에 복사한다.

단계 4. 각 서브블록 *BK(i)*에서 함수 $\delta(q_j, x_i) = q_r$, $0 \le j,r \le m-1$의 전위 함수에 대한 버스 경로를 구성한다. 각 *RM(k)* $0 \le k \le \lceil n/2 \rceil - 1$에 대해서 k가 짝수인 경우는 *BK(2k)*에 보조정리 1을 직접 적용하고 *BK(2k+1)*에는 첫 번째 열을 기준으로 하여 보조정리 1을 적용한다. k가 홀수인 경우는 짝수의 반대 경우를 적용한다.

단계 5. *BK(0)*에서 시작 상태 q_i를 갖는 처리기 *PE(0,i)*의 N 포트에 '*'를 전송한다. *BK(n-2).PE(m-1,j)* $0 \le j \le m-1$가 전송된 값을 받았는지를 확인한다. 만일 이 값을 전송 받은 *BK(n-2).PE(m-1,j)*가 종결 상태일 경우 입력 문자열은 승인되고 그렇지 않으면 거부된다.

위의 알고리즘에서 단계 4는 (그림 4)에서 화살표 방향으로 표시된 각 전이 상태에 대한 버스 경로를 생성해 주고, 단계 5는 시작 상태가 q_0인 경우 *BK(0).PE(0,0)*의 N 포트에 '*' 값을 전송한다. 만일 q_i가 종결 상태라고 가정하면 *BK(n-2).PE(m-1,j)*가 '*' 값을 받았으면 입력 문자열은 승인되고 그렇지 않으면 거부된다. 다음 정리는 위의 알고리즘의 결과를 정리하며 각 단계가 상수시간에 수행될 수 있음을 보인다.

[정리 3] {0,1}로 구성된 길이 n의 짝수 개의 입력 문자열과 m개의 상태를 갖는 유한 자동장치가 주어질 때, 입력 문자열의 승인 여부는 2차원 $\lceil nm/2 \rceil \times 2m$ RMESH에서 *O(1)* 시간에 결정될 수 있다.

[증 명] 단계 1은 보조정리 2에 의해서 *O(1)* 시간에 수행된다. 단계 2에서 각 *BK(j)* $0 \le j \le n-1$가 어느 *RM(k)* $0 \le k \le \lceil n/2 \rceil - 1$에 포함되는지는 $\lfloor j/2 \rfloor = k$ 를 계산하면 쉽게 확인할 수 있고, 각 k가 홀수인지 혹은 짝수인지는 *(k)mod(n)*의 값을 계산하여 0과 비교하면 된다. 또한 k가 홀수이면 *BK(2k)*의 *PE(0,0)*와 *BK(2k+1)*의 *PE(0,0)*에 저장된 입력 문자는 *RM(k)*에서 행의 버스를 사용하면 *O(1)* 시간에 서로 교환이 가능하다. 단계 3에서 각 *BK(i)* $0 \le i \le n-1$의 *PE(0,0)*에 저장된 입력 문자의 값은 각 서브블록에서 첫 번째 행과 열의 버스를 이용하면 *O(1)* 시간에 첫 번째 행과 열의 처리기들에 전송된다. 단계 4는 보조정리 1에 의해서 *O(1)* 시간에 수행된다. 단계 5는 단계 4에서 설정된 버스

경로에서 한 번의 자료 전송과 $BK(n-2)$의 마지막 열의 처리기들이 전송된 값을 받았는가의 여부만을 확인하므로 $O(1)$ 시간에 수행된다. 따라서 주어진 입력 문자열의 승인 여부는 2차원 $\lceil nm/2 \rceil \times 2m$ RMESH에서 $O(1)$ 시간에 결정된다. □

지금까지 우리는 입력 문자열의 길이 n이 짝수인 경우를 고려하였다. n이 홀수인 경우에는 위 알고리즘의 단계 5에서 $BK(n-2).PE(m-1,j)$ 부분을 $BK(n-1).PE(j,0)$ $0 \le j \le m-1$로 바꾼다. 즉 n이 홀수인 경우는 결과의 승인 여부를 서브블록 $BK(n-1)$의 첫 번째 열의 처리기들에서 확인한다. 따라서 이러한 사실과 정리 3에 의해서 아래와 같은 결과를 요약할 수 있다.

[따름정리 4] 길이 n의 홀수 개의 입력 문자열과 m개의 상태를 갖는 유한 자동장치가 주어질 때, 입력 문자열의 승인 여부는 2차원 $\lceil nm/2 \rceil \times 2m$ RMESH에서 $O(1)$ 시간에 결정될 수 있다.

3.2 일반적인 문자로 구성된 입력 문자열의 인식

본 논문의 2.1절에서는 입력 문자가 {0,1}로 제한된 경우에 대해서 유한 자동장치를 RMESH에 임베딩하여 입력 문자열을 인식하는 알고리즘을 설명하였다. 이제 본 논문에서 제안한 알고리즘 FA_Recognition이 일반적인 문자로 구성된 문자열이 주어질 경우 어떻게 확장될 수 있는지를 설명한다. 유한 자동장치의 5개 객체 중에서 입력 문자 알파벳 Σ는 임의의 문자로 주어지고 $|\Sigma| \le m$이라 한다. 즉 입력 문자들은 $\{a_0, a_1, \cdots, a_{|\Sigma|-1}\}$의 형식으로 표현될 수 있고, 여기서 각 원소 a_i는 임의의 문자로 주어진다. 이런 경우 <표 1>에서 보여진 것과 유사한 전이 함수의 상태 표는 $m \times |\Sigma|$ 크기를 갖는다. 따라서 이러한 일반적인 입력 문자열이 알고리즘 FA_Recognition에서 적절하게 수행되기 위해서 단계 1과 단계 3의 조정이 필요하다.

단계 1의 조정을 위해서 입력 문자 a_j를 Σ에서 j번째에 위치한 문자로 정의하고, 초기에 $m \times m$ 크기를 갖는 각 서브블록 $BK(k)$ $0 \le k \le n-1$의 j번째 열의 처리기 $PE(i,j)$ $0 \le i,j \le m-1$에 전이 함수 $\delta(q_i, a_j)$의 값이 주어졌다고 가정한다. 실제 이러한 전위 함수 값들은 초기에 서브블록 $BK(0)$에 적재된 후에 나머지의 서브블록들에 복사될 수 있다. 만일 $|\Sigma|$가 고정된(fixed) 크기를 갖는 경우는 각 서브블록에 대한 자료의 복사는 몇 번의 라우팅에 의해서 상수시간에 수행될 수 있다. 그러나 $|\Sigma|$가 m보다 적은 값을 가지면서 m에 비례하는 크기를 가질 경우 각 서브블록에 상수시간에 자료를 이동하는 문제는 쉽지 않다. 일반적인 문제에서 입력 문자는 고정된 크기로 제한되며 본 논문에서는 $|\Sigma|$가 ε인 상수의 경우를 고려한다. 따라서 알고리즘 FA_Recognition의 단계 1은 $O(1)$ 시간에 수행된다.

다음에 단계 3에서 보조정리 1을 적용하기 전에 각 서브블록 $BK(k)$ $0 \le k \le n-1$의 첫 번째 처리기 $PE(0,0)$에 저장된 입력 문자 x_k가 입력 문자 알파벳 Σ에서 어떤 위치에 포함되는지의 인덱스 값을 알아야 한다. 이러한 과정은 다음과 같은 단계에 의해서 계산될 수 있다. 먼저 각 서브블록 $BK(k)$ $0 \le k \le n-1$의 첫 번째 행에 Σ에 포함된 문자열을 순서적으로 배치한 후에 행의 버스를 구성한다. 다음에 각 서브블록 $BK(k)$의 $PE(0,0)$가 x_k를 첫 번째 행에 전송하면, 첫 번째 행의 처리기 $PE(0,j)$가 이 값과 자신이 저장한 문자를 비교하며 일치할 경우 인덱스 j를 $PE(0,0)$에 전송한다. 따라서 단계 3은 $O(1)$ 시간에 수행될 수 있다.

알고리즘 FA_Recognition의 나머지 단계들은 {0,1}로 구성된 입력 문자열의 경우와 똑 같이 적용되므로 다음의 정리가 성립된다.

[정리 5] Σ의 크기가 ε(고정된 상수)인 m개의 상태를 갖는 유한 자동장치와 길이 n의 입력 문자열이 주어질 때, 입력 문자열의 승인 여부는 2차원 $\lceil nm/2 \rceil \times 2m$ RMESH에서 $O(1)$ 시간에 결정될 수 있다.

4. 결 론

본 논문에서는 길이 n인 입력 문자열과 m개의 상태를 갖는 결정적 유한 자동장치가 주어질 때, $\lceil nm/2 \rceil \times 2m$ 크기의 RMESH 상에 유한 자동장치의 전이 상태를 임베딩하고 상수시간에 입력 문자열의 승인 여부를 결정하는 효율적인 병렬 알고리즘을 제안하였다. 유한 자동장치는 텍스트 편집기, 구문 분석기, 스위칭 회로와 같은 다양한 문제들의 해결을 위해 유용한 설계 도구로 이용되고 있다. 또한 이러한 결정적 유한 자동장치 알고리즘은 형식 3의 부류인 정규언어를 인식할 수 있다.

본 논문에서는 2차원 RMESH 상에서 결정적 유한

자동장치 문제만을 고려하였다. 앞으로 결정적 유한 자동장치에서 상태의 수를 줄이는 문제와 비결정적 유한 자동장치의 해결 문제가 추가적으로 연구되어야 할 것으로 기대된다.

참 고 문 헌

[1] R. Miller, V.K. Prasanna-Kumar, D. Resis, and Q.F. Stout, "Parallel computations on reconfigurable meshes," *IEEE Trans. Comput.*, Vol.42, No.6, pp.678-692, Jun. 1994.

[2] H.M. Alnuweiri, "Parallel constant-time connectivity algorithms on a reconfigurable network of processors," *IEEE Trans. Parallel Distribut. Systems*, Vol.6, No.1, pp.105-110, Jan. 1995.

[3] B.F. Wang and G.H. Chen, "Constant time algorithms for the transitive closure and some related graph problems on processor array with reconfigurable bus systems," *IEEE Trans. Parallel Distribut. Systems*, Vol.1, No.4, pp.500-507, Oct. 1990.

[4] R. Lin and S. Olariu, "Reconfigurable buses with shift switching : concepts and applications," *IEEE Trans. Parallel Distribut. Systems*, Vol.6, No.1, pp.93-104, Jan. 1995.

[5] T.W. Kao, S.J. Horng, Y.L. Wang, and H.R. Tsai, "Designing efficient parallel algorithms on a CRAP," *IEEE Trans. Parallel Distribut. Systems*, Vol.6, No.5, pp.554-560, May. 1995.

[6] T. Hayashi, K. Nakano, and S. Olariu, "Efficient list ranking on the reconfigurable mesh with applications," *Proc. 7th International Symposium on Algorithms and Computation*, pp.326-335, Dec. 1996.

[7] S. Olariu, J.L. Schwing, and J. Zhang, "Applications of reconfigurable meshes to constant-time computations," *Parallel Computing 19*, pp.229-237, 1993.

[8] R. Lin, "Fast algorithms for lowest common ancestors on a processor array with reconfigurable buses," *Inform. Processing Lett.*, Vol.40, pp.223-230, 1991.

[9] J.W. Jang and V.K. Prasanna, "An optimal sorting algorithm on reconfigurable meshes," *J. Parallel Distribut. Comput. 25*, pp.31-41, 1995.

[10] J.D. Ullman and J.E. Hopcroft, Introduction to automata theory, languages, and computation, Addison-Wesley Publishing Company, pp.13-76, 1979.

[11] A. Jain and N.S. Chaudhari, "Efficient parallel recognition of context-free languages," *Parallel Computing*, Vol.20, pp.1303-1321, 1994.

[12] 이동훈, 재구성 가능 버스 시스템으로 구성된 처리기 배열에서의 상수시간 언어 인식, 석사학위 논문, 서강대학교, 1994.

[13] 이광의, 재구성 가능 메쉬에서 집합의 순서열 계산과 리스트 랭킹 문제에 대한 상수시간 알고리즘에 관한 연구, 박사학위 논문, 서강대학교, 1997.

김 영 학

e-mail : yhkim@cespc1.kumoh.ac.kr

1984년 금오공과대학교 전자공학과 (전자계산기공학) 졸업

1989년 서강대학교 대학원 전자계산학과(공학석사)

1997년 서강대학교 대학원 전자계산학과(공학박사)

1989~1997 해군사관학교 전산과학과 교수

1998~1999 국립 여수대학교 산업정보학과 교수

1999~현재 금오공과대학교 컴퓨터공학부 교수

관심분야 : 병렬 알고리즘, 병렬 처리, 멀티미디어 등