電気通信 第5巻4号·1983. 12.

M10CN電子交換機 状態遷移 Table의 再構成에 対한 研究

李潤馥 · 金顯禹 / 交換技術 1 室

〈Abstract〉

Sine call handling for stored program control exchange can be regarded as finite state machine model of sequential process, the state transition concept is applied to describe its functions.

Coding method of the state transition table applied to M10CN ESS call processing and concerned topics was discussed and proposed another method for reduction memory usage.

Ⅰ. 序　言

1979年 以来 우리나라에 導入 運用되고 있는 M10CN 電子交換機의 소프트웨어는 交換機 相互間의 信号整合問題, 새로운 서비스의 增設및 維持保守 機能의 強化, 現地 適用 프로그램의補完및 修正等에 따라 3回의 全面的 改編을 거쳐 現在에 이르고 있다.

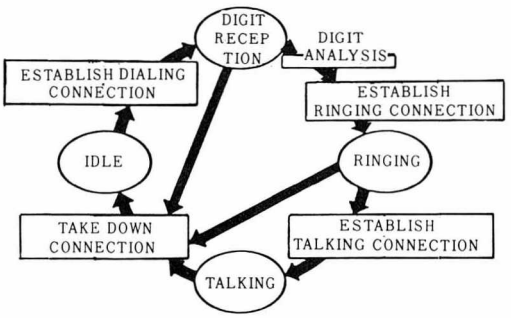
勿論, 이러한 改編은 交換소프트웨어의 根本的인 디자인에 対한 修正은 아니고 基本 概念의 具体化 過程및 方法, 各種 데이타 베이스와 테이블등에 対해 修正이 주로 이루어진 것이다.

本稿에서는 이런 M10CN電子交換機 呼処理프로그램中의 一部인 状態遷移 테이블의 再構成에 따른 메모리 使用의 效率性에 対해서 考察해 본 것이다.

Ⅱ. 電子交換機의 呼処理 概念

蓄積프로그램 制御方式 電子交換機의 呼処理 소프트웨어의 基本原理는 sequential process, concurrent process, finite state machine의 概念으로 생각할 수 있다.

1. Sequential Process



〈그림 1〉 Sequential Process 의 一例

Sequential process는 짜여진 順序에 依하여 一回에 発生되는 動作의 連続됨이라 表現할 수 있다.

〈그림 1〉은 呼処理 制御 過程에서의 sequential process의 한 例를 図示한 것이다. 이 sequential process는 다음의 主要 特性을 갖고 있다.

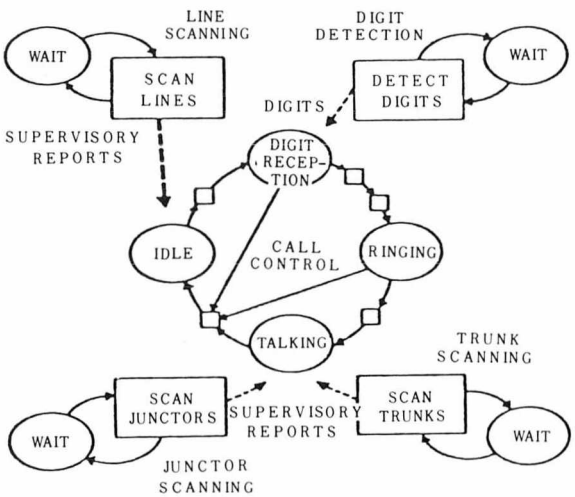
—Sequential program이 遂行됨으로써 얻어지는 結果는 그 program遂行 速度와는 無関하고 단지 定해진 遂行 速度로 한 瞬間에는 한 번만의 動作이 遂行되며 또한 프로그램 遂行 前과 後에는 어떤 関係를 갖는다.

Sequential program의 이러한 特性은 program作成時 operating system의 scheduling policy나 processor와 周辺装置에서 遂行되는 動作의 正確한 timing等을 計算하지 않아도 되는 長点을 갖게 된다.

—Sequential program은 같은 入力 data가 주어질 때의 遂行한 結果는 언제나 같게 된다. 바로 이 같은 reproducible behaviour는 program의 validation이나 debugging에 있어서 特히 重要하다.

2. Concurrent Process

Concurrent process는 program遂行이 同一時間内에서 重復되어 遂行됨을 말하며 여러 個의 많은 独立된 sequential program의 遂行이 同時에 일어난다. 이에 対한 典型的인 例로는 交換機에서 数 千個의 呼가 同時 処理過程에 있을 경우이다.〈그림 2〉는 電子交換機에서 遂行되는 concurrent processing의 一部를 図示한 것이다.



〈그림 2〉 ESS 에서의 Concurrent Process

이들 concurrent processing의 大部分은 連続的인 遂行 過程을 갖는다.

交換機로 入力되는 各種 信号들을 check하기 위해 週期的으로 遂行되는 여러 個의 走査 過程(Scanning process)이 한 例이다. 또다른 concurrent process의 一部는 呼処理 制御過程과 같은 event driven process로서 이들은 連続的인 scanning process에서 処理한 出力을 그 入力으로 하여 遂行된다.

3. Finite State Machine

Finite state machine의 概念은 한 時点에는 한 様相에만 関與함으로써, 또 安定한 system에 内在하는 virtual machine을 자세하게 定義함으로써 매우 複雜한 機能을 分析하는 技法中의 하나이다. 이 virtual machine이란 어떤 무엇이 発生되기를 기다리는 동안에 갖는 어느 程度安定한 状態라고 생각될 수 있다. 이 어떤 무엇은 event라고 불리우며 적당한 処理過程을 줌으로써 現在의 状態를 다음의 새로운 状態로 遷移를 일으킨다.

Finite state machine은 event driven sequential process를 model化 하는데 有用하다. 特히 電子交換機의 呼処理 過程은 finite state machine의 概念과 그 定義에 使用 되어진 用語로 表現할 수 있으며 通常的으로 定義되어 있는 finite state machine은 다음으로 構成되어 있는 抽象的인 model이다.

- 初期 状態를 포함하는 状態들의 有限 個의 集合
- 有限個인 入力들의 集合
- 出力 函数
- 다음 状態의 決定 函数

이 finite state machine은 다음 式으로 表現할 수 있다.

$$S' = f(E_a,\ E_b,\ E_c,\ \cdots\cdots E_n)$$

$$S' = f(S,\ E_n)$$

但, S' : 새로운 状態

S : 現在 状態

$E_a \cdots\cdots E_n$: Event

Finite state machine의 定義에 使用된 用語들은 電子交換機에서는 다음과 같이 定義하여 使用한다.

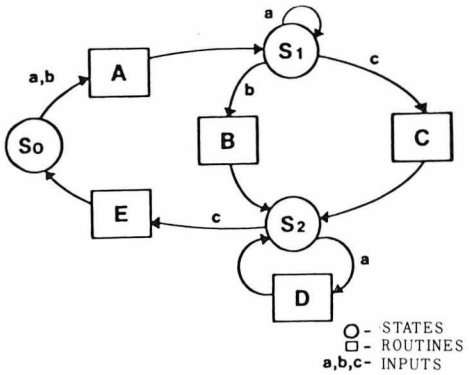
• 状態 : 어떤 安定 状態에 있는 呼의 段階

• 入力 : 加入者의 off-hook, on-hook, 또는 address signal같은 external event, 周辺装置에 対하여 順序的으로 내린 命令에 대한 完了信号, time-out의 終了같은 internal event.

• 出力函数 : 呼의 状態를 더 進行시켜 주는 routine들의 集合

• 다음状態決定函数 : 다음 状態로 遷移시켜주는 状態遷移 routine.

〈그림3〉은 sequential process의 finite state machine의 한 model을 例示한 것이다. 여기에는 状態, 処理routine, 各 状態에 주어지는 入力들을 나타내 보이고 있다.



〈그림 3〉 Sequential Process 의 Finite State Machine 의 한 Model

Finite state machine을 具体化하는 데는, graphic, production language, transition matrix, transition table等의 여러가지 다른 形態로 表現할 수 있다.〈그림 3〉과 같은 graph에 依한 表現은 視覚 効果가 좋아 理解에 便利한 点은 있으나 컴퓨터나 交換機用 프로세서에 기억시켜 処理하기 위해서는 이를 table形態로 表現하여 널리 使用한다.

〈表 1〉과〈表2〉는〈그림3〉의 model을 出力函数 決定 table과 다음 状態 決定 table의 2個 table로 作成하여 본 것이다.

이 model의 現在 状態를 알아 이 table을 access하여 그 状態를 入力시키면 이 状態에 따라 遂行해야 할 routine을 불러내며 다음 状態를 알아낼 수 있다. 이 model은 入力条件에 適用될 出力 函数 routine과 다음 状態를 함께 定義함으로써 한 個의 table로도 表現할 수 있다. M-10CN ESS에서는 이를 状態遷移 table이라 하여 한 個의 table로 構成하였다.

CURRENT STATE	INPUT a	INPUT b	INPUT c
S_0	A	A	X
S_1	—	B	C
S_2	D	X	E

〈表 1〉 出力函数決定 table

CURRENT STATE	INPUT a	INPUT b	INPUT c
S_0	S_1	S_1	S_0
S_1	S_1	S_2	S_2
S_2	S_2	S_0	S_0

〈表 2〉 다음 状態決定 table

Ⅲ. M10CN ESS의 状態遷移 table의 構成方法

M10CN ESS의 呼処理 프로그램은 finite state machine에 根據를 둔 一般的인 table driven program으로 構成되어 있으며 여기서는 呼処理 프로그램 中 局間中継信号 処理에 適用시킨 finite state machine model과 이에 따른 状態遷移 table만을 考察하였다. 이 finite state machine에 入力되는 event로는,

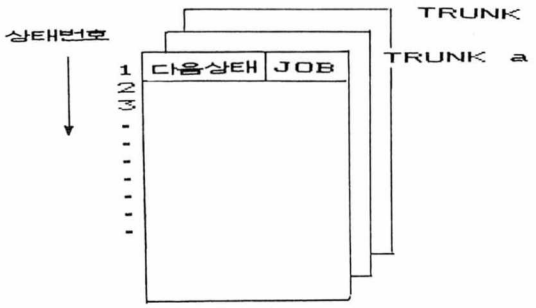
—mismatches of testpoints

—time-out의 終了

—기타 다른 프로그램에서의 command.

이때의 各 状態는 이들 入力 event와 이들 event의 発生 順序에 따른 函数関係를 갖게 된다. 이들 프로그램들을 標準化된 一般的이고 単一한 package로 꾸미기 為하여〈그림4〉와같은 3次元을 갖는 状態遷移테이블을 構成하였다.

〈그림4〉의 table을 보면 現在의 状態 番号와 入力 event 番号가 주어지면 다음 새로운 状態를



〈그림 4〉 狀態遷移 테이블의 構造

알아낼 수 있다. 이 새로운 狀態로 遂行해야 할 job을 定義해야 한다. 이 job은 다음과 같이 coding하였다. 各 狀態에 番号를 附與하여 이 狀態番号에 函数 関係를 갖는 job list table을 構成하여 이 table에서 指示된 job routine들을모두 遂行할 때까지 차례로 処理하는 경우이다.

이 狀態遷移table에서의 trunk回路의 種類에 따라 이들 상태에 番号를 附與하는 2가지 方法에 따른 table coding方法을 比較하여 보자.

• (方法 a)

回路의 種類別로 各各 番号를 附與하는 方法으로 回路 種類別로 狀態遷移table이 各各 한 個씩 存在하며 回路의 種類와 狀態番号로 addressing되는 table 構造를 갖는다.

• (方法 b)

한 回路에 附與한 狀態 番号에 連続하여 番号를 附與하는 方法으로 回路의 種類에 関係없이 한 個의 커다란 table이 存在하며 이 狀態 番号는 回路의 種類를 포함하므로 狀態 番号만으로 addressing되는 table構造를 갖는다.

(方法 b)는 한 個의 긴 table만을 使用하므로 여러 個의 table을 access하기 為한 index 用 table이 不必要하고 狀態 番号의 追加에 対備하기 위한 予備 data用 memory도 일괄적으로 確保되어 (方法a)보다 table data用 memory를 덜 必要로 하나 回路의 種類에 따른 狀態 番号를 M10CN ESS 呼処理 프로그램 構造上 call buffer에 追加로 記憶시켜 두어야 하므로 call buffer에는 過多한 量의 data slice를 確保해야 한다. 現在의 M10CN 電子交換機에서는 (方法 a)를 採択하여 運用中에 있다.

Ⅳ. 狀態遷移테이블의 再構成案 提示

初期에 導入된 M10CN 電子交換機에서의 trunk回路의 種類는 20餘種에 不過하였으나 多様한 信号方式의 処理, 維持保守機能의 強化및 더많은 서비스의 제공을 위하여 trunk回路의 種類를 늘려 現在는 40餘種으로 構成되어 있다.

이와 같은 trunk回路種類의 增加는 Ⅲ章에서 본 바와 같이 (方法a)를 採択한 関係로 이에 正比例하여 狀態遷移table의 data用 memory의 增加를 가져오게 된다.

〈表7〉에서 보는 바와 같이 system当 20,000加入者, 加入者 平均通話量 0.2Erlang으로 설계하였을 경우 (方法 a)가 (方法 b)를 採択하였을 경우보다 導入 初期 基準으로 約 700bytes가량의 memory가 節約되나 現在의 경우는 거의 비슷한 量의 memory가 使用되어지고 있다.

앞으로 処理해야 할 trunk回路의 種類가 늘어가게 되면 (方法b)가 memory 使用 効率面에서 더욱 有利할 것으로 여겨진다.

Ⅴ. 結　　語

以上에서는 電子交換機의 呼処理 基本 概念과 呼処理 프로그램中 狀態遷移테이블의 理論的 考察과 이에 対해 M10CN 電子交換機에서는

소요 Memory 량	방법(a)		방법 b)를 사용할 경우 추정량	
	도입초기	현재	도입초기	현재
Table	약 2,000	약 3,000	약 1,200	약 1,500
Call Buffer	약22,000	약22,000	약23,500	약23,500
계	약24,000	약25,000	약24,700	약25,000

(단위 = Byte)

〈表 3〉 상태천이 Table 再構成에 따른 Memory 소요량의 비교

그 構成方式을 比較하여 記述하였다.

Ⅳ章에서 言及한 構成方式中 (方法b)는 状態遷移테이블의 메모리 節約의 利点外에도 addressing回数의 減小等으로 因하여 全般的인 프로세서의 overhead가 줄어드는 利点도 있다. 그러나 回路状態의 加減및 状態遷移의 変更에는 테이블 構造上 不変動 既存 回路 状態遷移 테이블에 対해서도 再assembly를 해야하는 短点을 가지고 있다.

局間 中継信号方式이 統一되는 等 그 동안 試行하였던 変動 要因이 거의 없어졌으므로 call buffer 体係의 一部 変更等 関聯 프로그램과의 関係를 신중히 考慮하여 (方法b)를 採択해 봄직도 하다.

参 考 文 献

1. Kawashima Hiroshi, Kazunori Futami, & Sadahiko Kano, "Functional Specification of Call Processing by State Transition Diagram," IEEE Trans. on Comm., Vol 19, No. 5, Oct., 1971.
2. Mills, Darid L.,"Communication Software," Proc. of the IEEE, Vol 60, No. 11, Nov., 1972.
3. Chang, H. Y, "Finite State Support Software System Overview," Bell Laboratories Technical Memorandum, TM 75-5331-6, July 1975.
4. Guido, Adams, "Metaconta 10CN Signalling Software Design Note," 144 ITT 63253 AS, BTMC, May, 1978.
5. Hansen, P. Brinch, Operating System Principles. Prentice-Hall, Inc, Chapter2 Sequential Processes and Chapter 3 Concurrent Processes 1973.