신용평가 전문가시스템의 원형개발

이 영숙 (경북대학교 대학원 경영학과 박사과정)
서 창교 (포항공과대학 산업공학과 경영정보시스템연구실)

제1장 서 론

1950년대 중반에 처음으로 상품화된 컴퓨터가 보급되기 시작한 이래, 정보기술(Information Technology)은 기업의 다양한 분야에 적용되어 왔다. 특히, 개인용컴퓨터(Personal Computer)와 관련 소프트웨어의 등장은 기업에서의 정보기술 잠재력을 더욱 크게 만들었고, 최근에는 전략적인 목적으로 정보기술을 활용하는 단계에 까지 이르렀다. 이러한 정보기술의 다양한 영역 中에서 비교적 최근에 등장했지만 가장 유망한 분야로 인정받고 있는 것 中의 하나가 전문가시스템(Expert System:ES)이다.[Tyran & George, 1993]

일명 지식베이스시스템(Knowledge-based System)이라고도 불리우는 전문가시스템은 어느 특정분야의 전문지식을 그 분야의 전문가로 부터 획득하여 체계적으로 컴퓨터에 저장해 둠으로써, 전문가가 아닌 많은 사람들이 전문가를 만나지 않고도 전문가와 같은 의사결정을 할 수 있도록 해주는 컴퓨터 프로그램이다.[Bonnet, Haton & Truong-Ngoc, 1988] 전문가시스템은 1950년대 부터 연구되어온 인공지능(Artificial Intelligence:AI)의 한 응용분야로 발전되어 왔으며, 최근에는 의사결정지원시스템(Decision Support System:DSS)의 한 분야로 연구되어지고 있다.

전문가시스템은 경영, 화학, 컴퓨터시스템, 전자, 엔지니어링, 군사, 지질학, 의학 등 다양한 영역의 문제에 적용되고 있으며(표 1 참조), 전문가시스템이 행하는 기본적인 활동(Basic Activities)은 해석(Interpretation), 예측(Prediction), 진단(Diagnosis), 설계(Design), 계획수립(Planning), 감시(Monitoring), 디버깅(Debugging), 수리(Repair), 지시(Instruction), 통제(Control) 등의 범주로 나누어 볼 수 있다.(표 2 참조)[Waterman, 1986]

< 표 1 > 전문가시스템의 적용분야와 시스템 例

적 용 영 역	시스템 例
경 영	CORPTAX, K-FOLIO, DAS
화 학	CRYSALIS, DENDRAL, MOLGEN, SPEX, SECS, TQMSTUNE
컴퓨터시스템	PTRANS, BDS, DART, XCON, YES/MVS
전 자 공 학	ACE, IN-ATE, NDS, TALIB, PALLADIO, REDESIGN, CADHELP
엔지니어링	REACTOR, DELTA, SACON, STEAMER
군 사	ADEPT, ASTA, HANNIBAL, RTC, KNOBS, I & W, HASP/SIAP
지 질 학	ELAS, LITHO, HYDRO, MUD, PROSPECTOR, DRILLING ADVISOR
의 학	MYCIN, VM, GUIDON, PUFF, SPE, ANNA, ATTENDING

< 표 2 > 전문가시스템의 기본활동과 시스템 例

기본활동	시스템 例
해 석	DENDRAL, CRYSALIS, TQMSTUNE, REACTOR, ELAS, VM, SPE, ASTA
예 측	I & W
진 단	PTRANS, DART, ACE, IN-ATE, HYDRO, MUD,SPE, PUFF, MYCIN
설 계	SPEX, SECS, XCON, PALLIDIO, ACES
계획수립	TATR, KNOBS, TALIB, PTRANS
감 시	PTRANS, YES/MVS, REACTOR, ANNA, VM
디 버 깅	MYCIN, VM, DRILLING ADVISOR, DELTA, ACE, PTRANS
수 리	TQMSTUNE
지 시	CADHELP, SOPHIE, STEAMER, GUIDON, ATTENDING
통 제	VM, PTRANS, YES/MVS

전문가시스템을 개발하는 방법으로는 Prolog, Lisp, C 등의 범용 프로그래밍 언어를 사용하여 처음부터 문제의 성격에 맞는 전문가시스템을 구축할 수도 있고, 그렇지 않으면 지식베이스(knowledge base)를 제외하고는 전문가시스템이 갖추어야 할 모든 기능을 갖추고 있는 전문가시스템 틀(expert system shell)을 이용하여 구축할 수도 있다. 전자의 개발방법이 문제성격에 맞는 시스템을 개발할 수 있다는 장점이 있는 반면에, 후자는 시간, 비용, 노력면에서 경제적이라는 장점이 있다.

본 연구는 Turbo-Prolog를 통해 은행의 대부계에서 활용할 수 있는 신용평가전문가시스템을 개발하려 한다. 이러한 목적을 위해 제 1 장. 서론에 이어 제 2 장. 신용평가전문가시스템의 개발에서는 신용평가전문가시스템의 개발과정을 설명하고, 제 3 장. 신용평가전문가시스템의 실행 例에서는 개발한 신용평가전문가시스템을 사용하는 방법을 예제를 통해 살펴본다. 마지막으로 제 4 장. 결론에서는 신용평가전문가시스템의 한계 및 앞으로의 개선방향에 대해 제시한다.

제 2 장 신용평가전문가시스템의 개발

일반적으로 전문가시스템은 다음 세단계를 거쳐 개발한다.[Mockler, 1989]

- 의사결정상황을 분석(analyzing)한다(1 단계)
- 의사결정상황을 재구성(decomposing)한다(2 단계)
- 시스템을 구현(implementing)한다(3 단계)

따라서 본장의 이후 부분에서는 신용평가전문가시스템의 개발과정을 위의 각 단계별

로 살펴본다. 단, 1 단계인 의사결정상황의 분석단계는 가상적인 시나리오[1]임을 밝혀둔다.

1. 의사결정상황의 분석단계 (1 단계)

D은행 본점 대부계의 계장인 홍길동씨는 요즘 과중한 업무에 시달리고 있다. 최근, 시중의 자금사정이 어려워지자 D은행에 대한 이 지역기업들의 대부신청이 급격히 늘어났고, 대부결정을 하기 위해서는 대부신청 기업들의 신용상태를 평가해야 하는데, 이러한 신용평가업무를 모두 홍길동씨가 책임지고 있기 때문이다. 홍길동씨는 자신의 업무를 분담해 줄 인원충원도 생각해 보았지만, 일반적으로 신입행원들이 신용평가를 할 수 있을 정도의 판단력과 지식을 갖기 위해서는 최소한 3년 이상의 경험이 필요하기 때문에 이는 당장의 해결방법이 되지 못했다.

어느날, 우연히 홍길동씨는 자신의 고충을 전산부장인 가나다씨에게 이야기하게 되었다. 그러자, 가나다씨는 홍길동씨에게 신용평가전문가시스템을 개발할 것을 제의하게 되었다. 만일 신용평가에 대한 홍길동씨의 전문지식을 잘 구현한 전문가시스템을 개발한다면, 신입행원들이 이 시스템을 통해 마치 홍길동씨와 같이 신용평가를 잘 할 수 있을 것이고, 그러면 홍길동씨의 업무부담도 줄어들 것이라는 것이 가나다씨의 주장이었다. 홍길동씨도 가나다씨의 주장이 타당하다고 믿었고, 따라서 두사람은 신용평가전문가시스템을 개발하기로 결정하였다.

〈 표 3 〉 신용평가전문가시스템 개발프로젝트의 개요

목 표	신입행원이 신용평가를 할 수 있게 한다
의사결정기준	재무비율이 업체평균 이상인 기업에게 대부를 한다. 대부신청 기업에게 개선해야 할 사항이 있으면 알려준다.
요 구 사 항	1. 대부기간동안 대부신청기업이 도산하지 않아야 한다. 2. 대부신청기업은 대부상환능력이 있어야 한다. 3. 대부신청기업의 효율성 향상을 지원해 준다.
문 제 특 성	대부신청기업의 재무비율을 업체평균값과 비교한다.
데 이 터	대부신청기업의 재무제표, 업체평균데이터
해 결 안	1. 대부신청기업의 도산가능성을 알 수 있는 전문가시스템을 설계한다. 2. 재무비율분석을 할 수 있도록 설계한다. 3. 대부승인을 하기전에 대부신청기업에게 개선해야 할 사항을 알려줄 수 있도록 시스템을 설계한다.

이후 일주일간 가나다씨는 상황분석을 위해 홍길동씨와 계속 이야기를 나누었고, 그에 따라 가나다씨는 신용평가전문가시스템 개발프로젝트의 개요를 표 3과 같이 정리하였다. 그리고 가나다씨는 홍길동씨와의 면담을 통해 홍길동씨가 신용평가를 할 때 사용하는 지식(knowledge)이나 판단기준(judgement criteria) 등을 알아내었고, 이를 79개의 룰(rule)로 정리하였다.

2. 의사결정상황의 재구성단계 (2 단계)

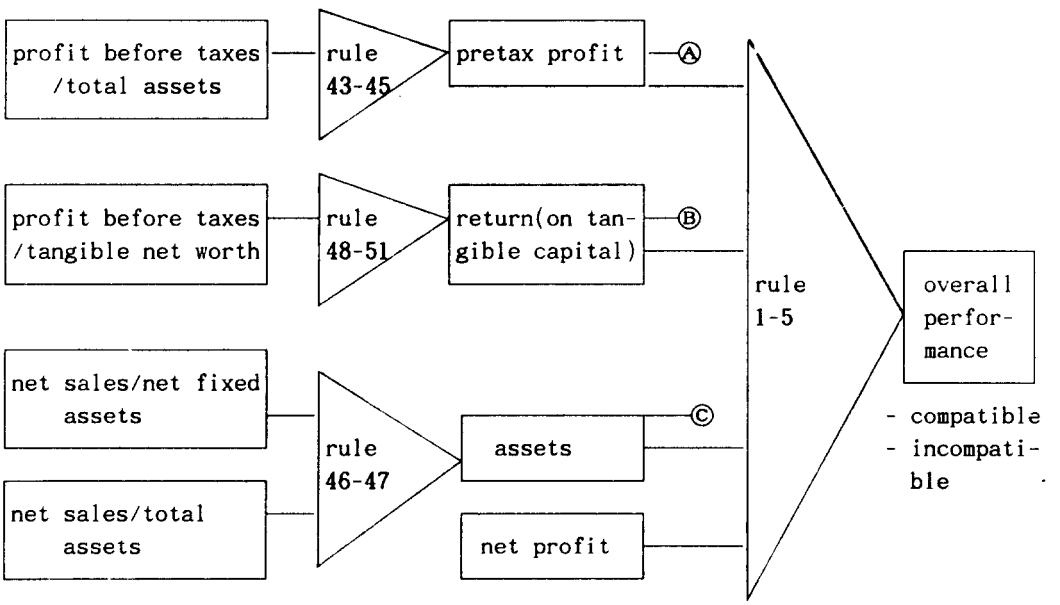
상황분석이 끝나고 나면, 다음으로는 시스템개발에 적합하도록 상황을 재구성한다. 상황의 재구성을 효과적으로 하기 위해서 여러가지 도식적인 도구(graphical tools)를 사용하는데, 이러한 도구(tool)로는 구조적 상황도(structured situation diagram), 종속도(dependency diagram), 의사결정도(decision chart) 등을 들 수 있다. 이 中 본 연구에서는 종속도와 의사결정도를 사용한다.

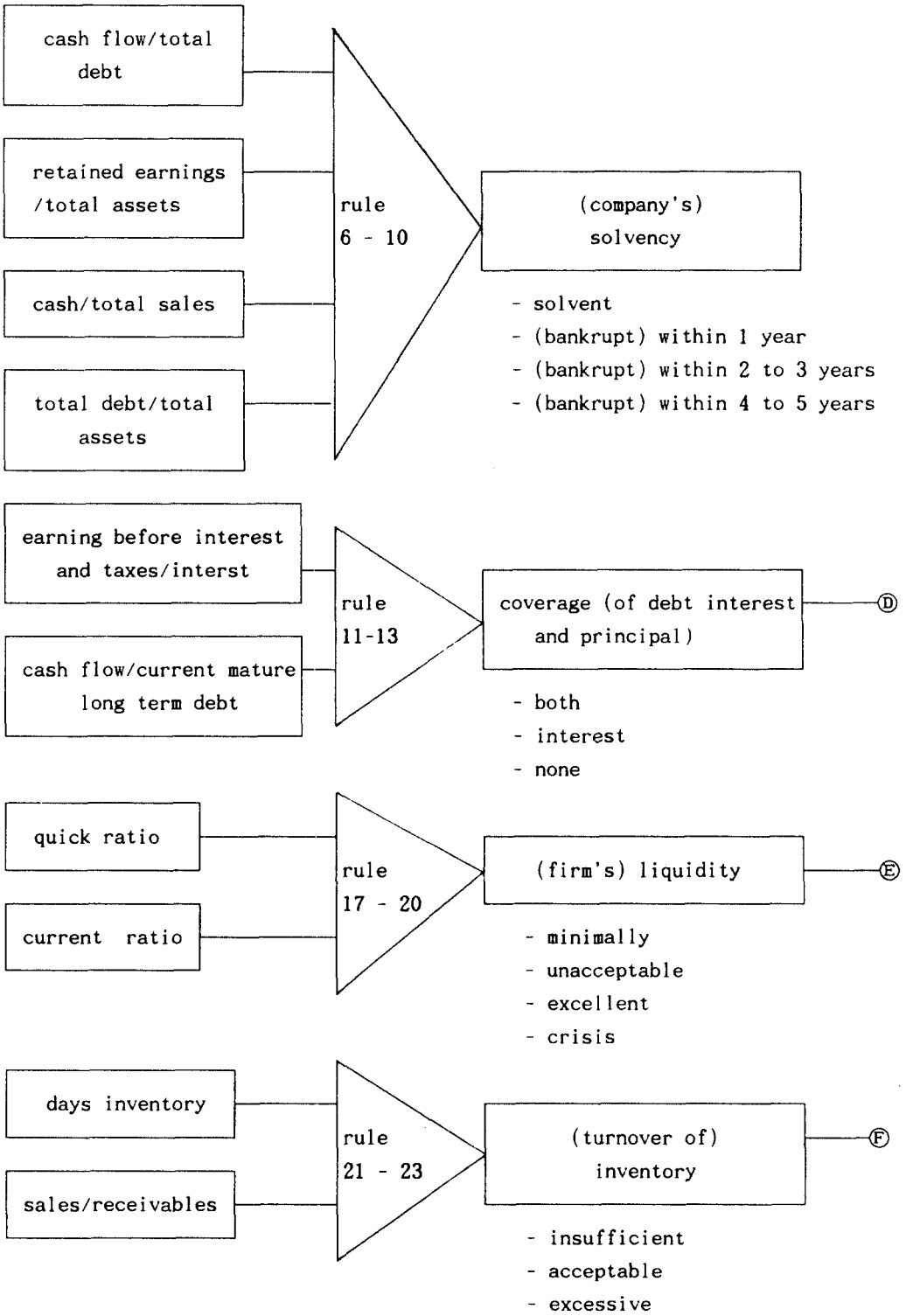
1) 종속도(dependency diagram)

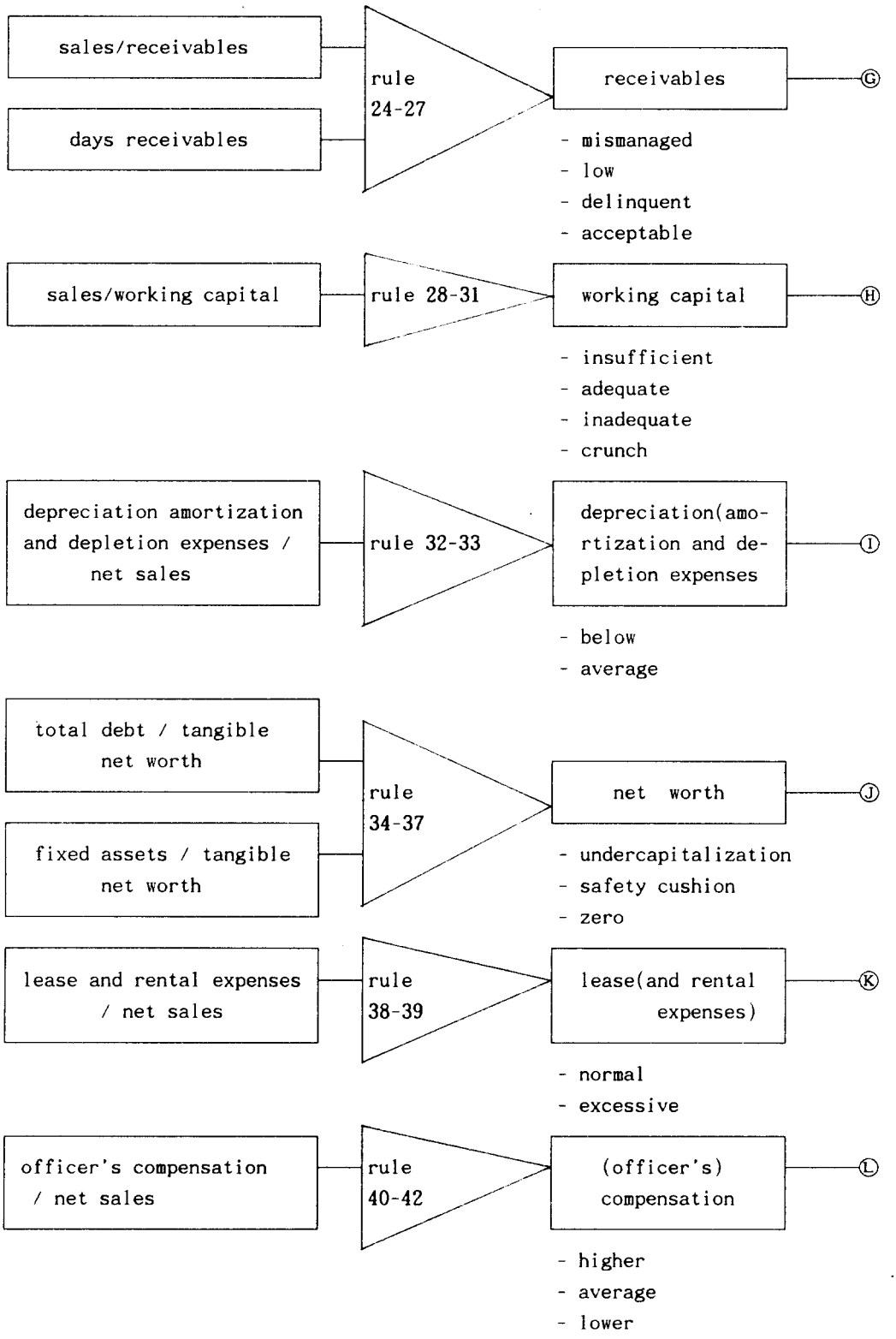
종속도(dependency diagram)란 시스템을 개발하고자 하는 문제영역의 의사결정과 관련있는 모든 대안(alternatives), 의사결정의 각 단계에서의 가능한 대안값(alternative values), 의사결정에 필요한 모든 지식항목, 지식의 처리에 사용되는 룰(rule) 등을 도식적으로 표현한 것이다. 따라서 종속도를 보면 그 문제영역과 관련있는 모든 지식, 지식간의 상호관련성, 추론과정 등을 알 수 있다.

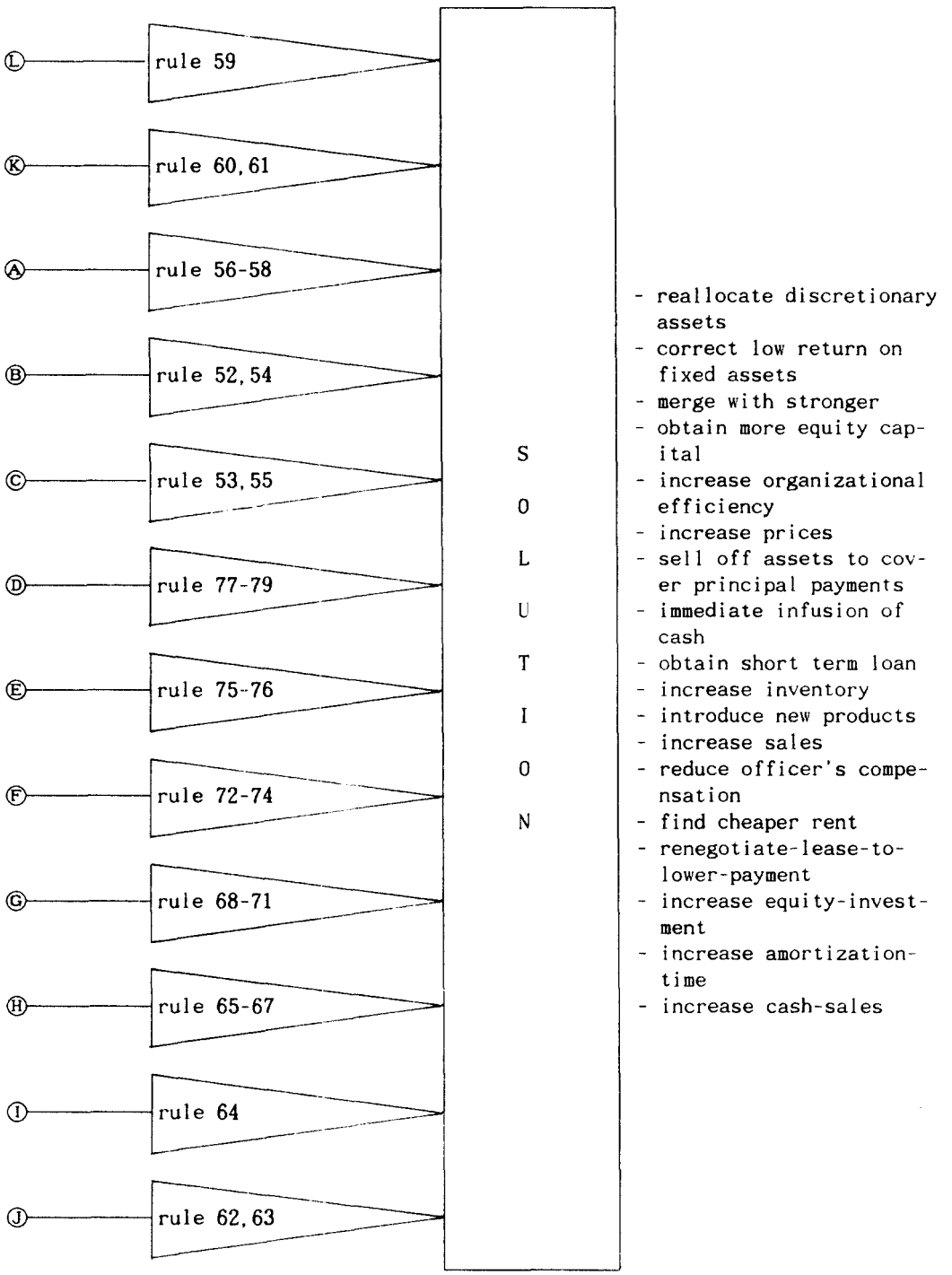
D은행에서 이루어지는 신용평가를 종속도로 나타내면 그림 1과 같다.

< 그림 1> 신용평가결정의 종속도(dependency diagram)




cash flow/total debt
retained earnings /total assets
cash/total sales
total debt/total assets
rule 6 - 10
(company's) solvency
- solvent
- (bankrupt) within 1 year
- (bankrupt) within 2 to 3 years
- (bankrupt) within 4 to 5 years
earning before interest and taxes/interst
cash flow/current mature long term debt
rule 11-13
coverage (of debt interest and principal)
Ⓓ
- both
- interest
- none
quick ratio
current ratio
rule 17 - 20
(firm's) liquidity
Ⓔ
- minimally
- unacceptable
- excellent
- crisis
days inventory
sales/receivables
rule 21 - 23
(turnover of) inventory
Ⓕ
- insufficient
- acceptable
- excessive


sales/receivables
days receivables
rule 24-27
receivables
Ⓖ
- mismanaged
- low
- delinquent
- acceptable
sales/working capital
rule 28-31
working capital
Ⓗ
- insufficient
- adequate
- inadequate
- crunch
depreciation amortization and depletion expenses / net sales
rule 32-33
depreciation(amortization and depletion expenses
Ⓘ
- below
- average
total debt / tangible net worth
fixed assets / tangible net worth
rule 34-37
net worth
Ⓙ
- undercapitalization
- safety cushion
- zero
lease and rental expenses / net sales
rule 38-39
lease(and rental expenses)
Ⓚ
- normal
- excessive
officer's compensation / net sales
rule 40-42
(officer's) compensation
Ⓛ
- higher
- average
- lower


Ⓛ
rule 59
Ⓚ
rule 60, 61
Ⓐ
rule 56-58
Ⓑ
rule 52, 54
Ⓒ
rule 53, 55
Ⓓ
rule 77-79
Ⓔ
rule 75-76
Ⓕ
rule 72-74
Ⓖ
rule 68-71
Ⓗ
rule 65-67
Ⓘ
rule 64
Ⓙ
rule 62, 63
S
O
L
U
T
I
O
N
- reallocate discretionary assets
- correct low return on fixed assets
- merge with stronger
- obtain more equity capital
- increase organizational efficiency
- increase prices
- sell off assets to cover principal payments
- immediate infusion of cash
- obtain short term loan
- increase inventory
- introduce new products
- increase sales
- reduce officer's compensation
- find cheaper rent
- renegotiate-lease-to-lower-payment
- increase equity-investment
- increase amortization-time
- increase cash-sales

2) 의사결정도(decision chart)

일단, 종속도(dependency diagram)가 작성되고 나면 의사결정의 각 단계별로 의사결정도(decision chart)를 그린다. 원칙적으로 의사결정도는 종속도에 나타난 삼각형의 숫자만큼 그리게 된다. 따라서, 그림 1의 종속도에 의하면 모두 26개의 의사결정도를 그려야 하나, 여기서는 전반적인 성과(overall performance) 평가단계에 대한 의사결정도만 제시한다.(표 4 참조)

< 표 4 > overall performance 평가단계의 의사결정도(decision chart)

Rule No	rule 1	rule 2	rule 3	rule 4	rule 5
pretax profit	effective	effective	ineffect-ive		
return on tan-gible capital	undercapi-talization	acceptable		deficit	under-employment
use of assets	acceptable	acceptable	unaccept-able		
net profit	acceptable	acceptable			
overall-performance	compatible	compatible	incompati-ble	incompati-ble	incompati-ble

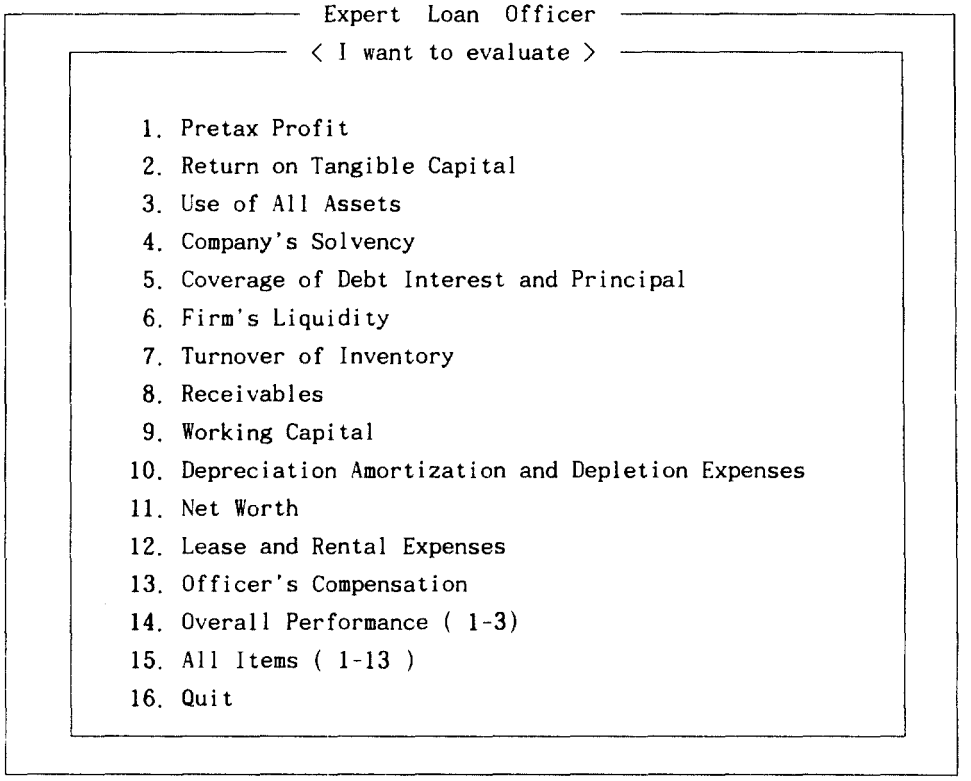
3 시스템구현단계

마지막 단계는 실제로 시스템을 구현하는 단계이다. 본 연구의 목표인 신용평가전문가시스템은 Turbo-Prolog 2.0을 이용하여 IBM PC 486에서 구현하였다. 사용자(신입행원)의 편이를 위해 사용자 인터페이스는 메뉴(menu) 방식을 채택하였고, 추론방법은 정방향 연결방법(forward chaining)을 채택하였다(소스 프로그램은 附錄을 참조하기 바람).

제 3 장 신용평가전문가시스템의 실행

신용평가전문가시스템의 실행을 위해서는 주요 프로그램인 "Loan.pro"와 메뉴를 위한 하위 프로그램인 "Menu.pro"[2], 그리고 실행 프로그램인 "Loan.exe"의 세가지 파일을 기본적으로 가지고 있어야 한다

< 그림 2 > 신용평가전문가시스템의 주 메뉴



시스템을 구동시키면 그림 2와 같이 16가지의 항목으로 구성된 주 메뉴가 나타난다. 사용자가 자신이 분석하고자 하는 항목을 주 메뉴에서 선택하여 Enter 키를 치면, 시스템은 선택된 항목의 분석을 위한 데이터를 입력할 것을 요구한다. 사용자가 시스템의 요구에 따라 적절하게 데이터를 입력하면, 시스템은 입력된 데이터를 처리(계산, 비교 등)하여 사용자에게 답을 제시해준다.

예를 들어 사용자가 기업의 지불능력(solvency)을 평가하고 싶은 경우에는, 먼저 주 메뉴에서 "4. Company's Solvency"를 선택한다. 그러면, 시스템은 기업의 지불능력(solvency)을 평가하는데 필요한 일련의 재무항목들에 대해서 질문을 하게 된다. 만일 평가하고자 하는 기업의 현금흐름이 5,000, 총부채가 50,000, 유보이익이 70,000, 총자산이 500,000인 경우의 사용자 입력 例는 그림 3과 같다. 이렇게 사용자가 입력

한 값들을 이용하여 비율(ratio)을 계산하고, 필요한 경우에는 업체평균과 비교한 다음, 시스템은 답("일년이내에 파산할 가능성이 있다")을 제시해준다.

시스템의 사용을 마치고 도스(DOS) 상태로 돌아가고자 하는 경우에는 주 메뉴에서 "16. Quit"를 선택한다.

< 그림 3 > 기업의 지불능력(solvency) 평가를 위한 입력 例

```
                    Expert Loan Officer

Input cash flow  :  5000
Input total debt  :  50000
Input retained earnings  :  70000
Input total assets  :  500000


—> The company could be bankrupt within one year.

                - Hit any key to continue -
```

제 4 장 결 론

본 연구에서는 Turbo-Prolog를 이용하여 신용평가전문가시스템을 개발하였다. 본 시스템에 구현된 지식은 은행의 대부계에서 대부신청기업의 신용상태를 평가할 때 사용하는 지식과 판단기준 등을 79가지의 룰(rule)로 변환한 것이다. 본 시스템은 재무제표상의 몇가지 값만을 입력해 주면, 신용평가에 필요한 비율계산이나 비교 등을 대신해주고 그에 따른 평가결과를 제시해 주기 때문에 신용평가에 대한 사전지식이나 경험이 없는 사용자들도 마치 전문가와 같이 신용평가를 할 수 있을 것이다.

그러나 본 시스템에는 다음과 같은 몇가지 문제점이 있다. 첫째, 전문가시스템이 기존의 정보시스템과 다른 점은 추론능력과 함께 설명능력을 갖고 있는 것이다. 그런데, 본 시스템은 설명능력을 갖추지 못하였다. 둘째, 오늘날은 금융여건이 계속 변화하고 있기 때문에 신용평가의 룰(rule)도 그에 따라 변해야 한다. 따라서 본 시스템은 룰 편집기능을 갖추는 것이 바람직하나, 현재로는 룰 편집기능을 갖추지 못하고 있다. 세째, 원래 신용평가는 기업의 재무적인 요소외에 비재무적인 요소도 포함되어야 한다. 그러나, 현재 은행계에서는 재무적인 요소만을 고려하고 있고, 본 시스템도 이러한 관행에 맞추어 재무적인 요소만을 평가하도록 설계되어 있다. 차후의 연구에서는 이러한 문제점들을 개선해야 할 것으로 사료된다.

부 록

```
/************************************************/
/*  Prototype Expert Loan Officer               */
/*                                              */
/*  program id: loan.pro                        */
/*  date: 1993. 8. 15.                          */
/*                                              */
/************************************************/
code = 5000
nowarnings
/************************************************/
                  database
/************************************************/
is_fact(symbol, real)
/************************************************/
                  predicates
/************************************************/
start_loan_advisor
loan_advisor
process(integer)
   pretaxProfit(symbol)
      act_pretaxProfit(char,symbol)
      out_pretaxProfit(symbol)
      solution_pretaxProfit(symbol)
   return(symbol)
      act_return(char,symbol)
      out_return(symbol)
      solution_return(symbol)
   assets(symbol)
      act_assets(char,char,symbol)
      out_assets(symbol)
      solution_assets(symbol)
   solvency(symbol)
      getvalue(symbol,real)
      out_solvency(symbol)
   coverage(symbol)
      act_coverage(char,char,symbol)
      out_coverage(symbol)
      solution_coverage(symbol)
   liquidity(symbol)
      act_liquidity(char,char,symbol)
      out_liquidity(symbol)
      solution_liquidity(symbol)
   inventory(symbol)
      act_inventory(char,char,symbol)
      out_inventory(symbol)
      solution_inventory(symbol)
   receivables(symbol)
      act_receivables(char,char,symbol)
      out_receivables(symbol)
      solution_receivables(symbol)
   workingCapital(symbol)
```

```
      act_workingCapital(char,symbol)
      out_workingCapital(symbol)
      solution_workingCapital(symbol)
   depreciation(symbol)
      act_depreciation(char,symbol)
      out_depreciation(symbol)
      solution_depreciation(symbol)
   netWorth(symbol)
      act_netWorth(char,char,symbol)
      out_networth(symbol)
      solution_netWorth(symbol)
   lease(symbol)
      act_lease(char,symbol)
      out_lease(symbol)
      solution_lease(symbol)
   compensation(symbol)
      act_compensation(char,symbol)
      out_compensation(symbol)
      solution_compensation(symbol)
/* overall                              */
      netProfit(symbol)
      act_overall(symbol,symbol,symbol,symbol,symbol)
      out_overall(symbol)
output(symbol, symbol, symbol)
pauser
verify_ab(char)
verify_abc(char)
max(real, real, real)
min(real, real, real)
/*********************************/
/* Borland's add-on menu module. */
/*********************************/
include "menu.pro"
goal
   start_loan_advisor.
/************************************************/
                  clauses
/************************************************/
start_loan_advisor :-
   makewindow(1,112,7,"< Expert Loan Officer >",0,0,25,80),nl,nl,
   write("  This is a Prototype Expert System to advise the novice loan officer.
"), nl,nl,
   write("  You must answer a series of questions."),nl,nl,
   write("  Then, Expert Loan Officer will analyze the company's overall perform
ance"), nl,
   write("  and provide the recommendation for the better management."), nl, nl,
   write("  Press any key to start : "),
   readchar(_),repeat,shiftwindow(1),clearwindow,
   loan_advisor.

loan_advisor :- clearwindow,
   menu(4,12,7,7,["  1. Pretax Profit",
                  "  2. Return on Tangible Capital",
                  "  3. Use of All Assets",
                  "  4. Company's Solvency",
                  "  5. Coverage of Debt Interest and Principal",
```

```
                    "  6. Firm's Liquidity",
                    "  7. Turnover of Inventory",
                    "  8. Receivables",
                    "  9. Working Capital",
                    " 10. Depreciation Amortization and Depletion Expenses",
                    " 11. Net Worth",
                    " 12. Lease and Rental Expenses",
                    " 13. Officer's Compensation",
                    " 14. Overall Performace Only (1 - 3)",
                    " 15. All Items (1 - 13)",
                    " 16. Quit"], "< I want to evaluate >", 16, Choice),
   process(Choice), Choice = 16, !.
process( 1) :- pretaxProfit(Status),out_pretaxProfit(Status),
               solution_pretaxProfit(Status),!,pauser.
process( 2) :- return(Status),out_return(Status),
               solution_return(Status),!,pauser.
process( 3) :- assets(Status),out_assets(Status),
               solution_assets(Status),!,pauser.
process( 4) :- solvency(Status),out_solvency(Status),
               retractall(is_fact(_,_)),!,pauser.
process( 5) :- coverage(Status),out_coverage(Status),
               solution_coverage(Status),!,pauser.
process( 6) :- liquidity(Status),out_liquidity(Status),
               solution_liquidity(Status),!,pauser.
process( 7) :- inventory(Status),out_inventory(Status),
               solution_inventory(Status),!,pauser.
process( 8) :- receivables(Status),out_receivables(Status),
               solution_receivables(Status),!,pauser.
process( 9) :- workingCapital(Status),out_workingCapital(Status),
               solution_workingCapital(Status),!,pauser.
process(10) :- depreciation(Status),out_depreciation(Status),
               solution_depreciation(Status),!,pauser.
process(11) :- netWorth(Status),out_netWorth(Status),
               solution_netWorth(Status),!,pauser.
process(12) :- lease(Status),out_lease(Status),
               solution_lease(Status),!,pauser.
process(13) :- compensation(Status),out_compensation(Status),
               solution_compensation(Status),!,pauser.
process(14) :- pretaxProfit(Profit),out_pretaxProfit(Profit),
               solution_pretaxProfit(Profit),pauser,
               return(Return),out_return(Return),
               solution_return(Return),pauser,
               assets(Assets),out_assets(Assets),
               solution_assets(Assets),pauser,
               netProfit(NetProfit),
               write("\n\n   Based on your input,\n"),
               act_overall(Profit,Return,Assets,NetProfit,Status),
               out_overall(Status),!,pauser.
process(15) :- process(14),process( 4),process( 5),process( 6),
               process( 7),process( 8),process( 9),process(10),
               process(11),process(12),process(13),!.
process(16) :- !.            /* end of consult */
/*********************************/
/*  process( 1) : pretax profit  */
/*********************************/
pretaxProfit(Status) :-
```

```
     write("  Percent profit before taxes to total assets\n\n"),
     write("    a: average or better\n"),
     write("    b: low\n"),
     write("    c: negative\n"), readchar(Response),
     act_pretaxProfit(Response,Status).
act_pretaxProfit('a',Status) :- Status = "effective",!.
act_pretaxProfit('b',Status) :- Status = "ineffective",!.
act_pretaxProfit('c',Status) :- Status = "deficit",!.
act_pretaxProfit(_,_) :- beep,nl,
     write("   Invalid response. Try again."), readchar(Response),
     act_pretaxProfit(Response,Status),!,nl.
out_pretaxProfit(effective) :-
     output("Pretax profit","shows","assets employed effectively"),!.
out_pretaxProfit(ineffective) :-
     output("Pretax profit","shows","assets not effectively employed"),!.
out_pretaxProfit(deficit) :-
     output("Pretax profit","indicates","deficit or losses"),!.
solution_pretaxProfit(ineffective) :-
     output("Solution","could be reallocate","discretionary assets"),
     output("Solution","could be","correct low return on fixed assets"),!.
solution_pretaxProfit(deficit) :-
     output("Solution", "could be merge","with stronger"),!.
solution_pretaxProfit(_) :- !.
/*********************************************/
/*  process( 2) : return on tangible capital  */
/*********************************************/
return(Status) :-
     write("  Percent profit before taxes to tangible net worth\n\n"),
     write("    a: high\n"),
     write("    b: average\n"),
     write("    c: low\n"),
     write("    d: negative\n"), readchar(Response),
     act_return(Response,Status).
act_return('a',Status) :- Status = "undercapitalization",!.
act_return('b',Status) :- Status = "acceptable",!.
act_return('c',Status) :- Status = "underemployment",!.
act_return('d',Status) :- Status = "deficit",!.
act_return(_,_) :- beep,nl,
     write("   Invalid response. Try again."), readchar(Response),
     act_return(Response,Status),!,nl.
out_return(undercapitalization) :-
     output("The return on tangible capital","is",
            "too high\n     indicating undercapitalization"),!.
out_return(acceptable) :-
     output("The return on tangible capital","is","acceptable"),!.
out_return(underemployment) :-
     output("The return on tangible capital","is",
            "too low\n     indicating underemployment of assets"),!.
out_return(deficit) :-
     output("The return on tangible capital","indicates","deficits or losses")
,!.
solution_return(undercapitalization) :-
     output("Solution","could be obtain","more equity capital"),!.
solution_return(underemployment) :-
     output("Solution", "could be increase","organizational efficiency"),!.
solution_return(_) :- !.
```

```
/*************************************/
/*  process( 3) : use of all assets  */
/*************************************/
assets(Status) :-
     write("  Net sales to net fixed assets is\n\n"),
     write("    a: average or greater\n"),
     write("    b: below average\n"),
     readchar(Fixed), verify_ab(Fixed),nl,
     write("  Net sales to total assets is\n\n"),
     write("    a: average or greater\n"),
     write("    b: below average\n"),
     readchar(Total), verify_ab(Total),
     act_assets(Fixed,Total,Status).
act_assets('b','b',Status) :- Status = "unacceptable",!.
act_assets('a','a',Status) :- Status = "acceptable",!.
act_assets( _ , _ ,Status) :- Status = "unknown",!.
out_assets(unacceptable) :-
     output("The productive use of all assets","is","unacceptable"),!.
out_assets(acceptable) :-
     output("The productive use of all assets","is","acceptable"),!.
out_assets(unknown) :-
     output("I can't evaluate the use of all assets\n",
            "     because","I have no knowledge for this case"),!.
solution_assets(unacceptable) :-
     output("Solution","could be increase",
           "organizational efficiency"),
     output("Solution","could be increase","prices"),!.
solution_assets(_) :- !.

/*************************************/
/*  process( 4) : company's solvency  */
/*************************************/
solvency(solvent) :-
     getvalue("cash flow",CashFlow),getvalue("total debt",TotalDebt),
     (CashFlow/TotalDebt) > 0.1309,
     getvalue("total assets",TotalAssets),
     (TotalDebt/TotalAssets) <= 0.6975,!.
solvency(solvent) :-
     getvalue("cash flow",CashFlow),getvalue("total debt",TotalDebt),
     (CashFlow/TotalDebt) <= 0.1309,
     getvalue("retained earnings",RetainedEarnings),
     getvalue("total assets",TotalAssets),
     (RetainedEarnings/TotalAssets) > 0.1453,
     getvalue(cash,Cash),getvalue("total sales",TotalSales),
     (Cash/TotalSales) > 0.025,!.
solvency(withinlyear) :-
     getvalue("cash flow",CashFlow),getvalue("total debt",TotalDebt),
     (CashFlow/TotalDebt) <= 0.1309,
     getvalue("retained earnings",RetainedEarnings),
     getvalue("total assets",TotalAssets),
     (RetainedEarnings/TotalAssets) <= 0.1453,!.
solvency(within2to3years) :-
     getvalue("cash flow",CashFlow),getvalue("total debt",TotalDebt),
     (CashFlow/TotalDebt) <= 0.1309,
     getvalue("retained earnings",RetainedEarnings),
     getvalue("total assets",TotalAssets),
     (RetainedEarnings/TotalAssets) > 0.1453,
     getvalue(cash,Cash),
     getvalue("total sales",TotalSales),
     (Cash/TotalSales) <= 0.025,!.
```

```
solvency(within4to5years) :-
     getvalue("cash flow",CashFlow),getvalue("total debt",TotalDebt),
     (CashFlow/TotalDebt) > 0.1309,
     getvalue("total assets",TotalAssets),
     (TotalDebt/TotalAssets) > 0.6975,!.
getvalue(Attribute,Value) :- is_fact(Attribute,Value),!.
getvalue(Attribute,Value) :- write("   Input ",Attribute," : "),
     readreal(Value),asserta(is_fact(Attribute,Value)).
out_solvency(solvent) :-
     output("The company","is","solvent"),!.
out_solvency(within1year) :-
     output("The company","could be","bankrupt within one year"),!.
out_solvency(within2to3years) :-
     output("The company","could be","bankrupt within two or three years"),!.
out_solvency(within4to5years) :-
     output("The company","could be","bankrupt within four to five years"),!.
/*************************************************************/
/*  process( 5) : coverage of debt interest and principal  */
/*************************************************************/
coverage(Status) :-
     write("  Earning before interest and taxes to interest ratio is\n\n"),
     write("    a: average or greater\n"),
     write("    b: low\n"),
     readchar(EBIT), verify_ab(EBIT),nl,
     write("  Cash flow to current mature long-term debt ratio is\n\n"),
     write("    a: average or greater\n"),
     write("    b: low\n"),
     readchar(CashFlow), verify_ab(CashFlow),
     act_coverage(EBIT,CashFlow,Status).
act_coverage('a','b',Status) :- Status = "interest",!.
act_coverage('a','a',Status) :- Status = "both",!.
act_coverage('b','a',Status) :- Status = "unknown",!.
act_coverage('b','b',Status) :- Status = "none",!.
out_coverage(interest) :-
     output("Coverage of deb interest","is possible",
            "but principal questionable"),!.
out_coverage(both) :-
     output("Coverage of debt interest and principal","is",
            "acceptable"),!.
out_coverage(none) :-
     output("Coverage of debt intest and principal","is",
            "in jeopardy"),!.
out_coverage(unknown) :-
     output("I can't evaluate coverage of debt interest and principal\n",
            "     because","I have no knowledge for this case"),!.
solution_coverage(interest) :-
     output("Solution","could be sell off",
            "assets to cover principal payment"),
     output("Solution","could be obtain","more equity"),!.
solution_coverage(none) :-
     output("Solution","could be obtain",
            "more equity and short term loan"),!.
solution_coverage(_) :- !.
/*****************************************/
/*  process( 6) : firm's liquidity       */
/*****************************************/
liquidity(Status) :-
     write("  Quick ratio is\n\n"),
     write("    a: equal to or greater than 1\n"),
     write("    b: less than 1\n"),
     readchar(Quick), verify_ab(Quick),nl,
     write("  Current ratio is\n\n"),
     write("    a: average or greater\n"),
     write("    b: less than average\n"),
     readchar(Current), verify_ab(Current),
```

```
     act_liquidity(Quick,Current,Status).
act_liquidity('a','b',Status) :- Status = "minimally",!.
act_liquidity('b','a',Status) :- Status = "unacceptable",!.
act_liquidity('a','a',Status) :- Status = "excellent",!.
act_liquidity('b','b',Status) :- Status = "crisis",!.
out_liquidity(minimally) :-
     output("The firm's liquidity","is","minimally acceptable"),!.
out_liquidity(unacceptable) :-
     output("The firm's liquidity","is","unacceptable"),!.
out_liquidity(excellent) :-
     output("The firm's liquidity","is","excellent"),!.
out_liquidity(crisis) :-
     output("The firm's liquidity","is","near crisis stage"),!.
solution_liquidity(crisis) :-
     output("Solution","could be","immediate infusion of cash"),!.
solution_liquidity(unacceptable) :-
     output("Solution", "could be", "short term loan"),!.
solution_liquidity(_) :- !.
/*****************************************/
/*  process( 7) : turnover of inventory   */
/*****************************************/
inventory(Status) :-
     write("  Sales to receivable ratio is\n\n"),
     write("    a: high\n"),
     write("    b: average\n"),
     write("    c: low\n"),
     readchar(SalesToReceivable), verify_abc(SalesToReceivable),nl,
     write("  Days inventory is\n\n"),
     write("    a: high\n"),
     write("    b: low\n"),
     readchar(DaysInventory), verify_ab(DaysInventory),
     act_inventory(SalesToReceivable,DaysInventory,Status).
act_inventory('a','b',Status) :- Status = "insufficient",!.
act_inventory('c','a',Status) :- Status = "excessive",!.
act_inventory('b','a',Status) :- Status = "acceptable",!.
act_inventory( _ , _ ,Status) :- Status =  "unknown",!.
out_inventory(insufficient) :-
     output("Turnover of inventory","indicates\n",
            "     probable insufficient inventory stocks"),!.
out_inventory(excessive) :-
     output("Turnover of inventory","indicates\n",
            "     poor sales - excessive production obsolescence"),!.
out_inventory(acceptable) :-
     output("Turnover of inventory","is","average and acceptable"),!.
out_inventory(unknown) :-
     output("I can't evaluate turnover of inventory\n",
            "     because","I have no knowledge for this case"),!.
solution_inventory(insufficient) :-
     output("Solution","could be increase","inventory"),!.
solution_inventory(excessive) :-
     output("Solution","could be", "new products"),
     output("Solution","could be to increase","sales"),!.
solution_inventory(_) :- !.
/*****************************************/
/*  process( 8) : receivables            */
/*****************************************/
receivables(Status) :-
     write("  Sales to receivable ratio is\n\n"),
     write("    a: average or lower\n"),
     write("    b: lower than average\n"),
     readchar(SalesToReceivable), verify_ab(SalesToReceivable),nl,
     write("  Days receivables is\n\n"),
     write("    a: greater than average\n"),
     write("    b: average or lower\n"),
     readchar(DaysReceivalbes), verify_ab(DaysReceivables),
```

```
     act_receivables(SalesToReceivable,DaysReceivables,Status).
act_receivables('b','a',Status) :- Status = "mismanaged",!.
act_receivables('b','b',Status) :- Status = "low",!.
act_receivables('a','a',Status) :- Status = "delinquent",!.
act_receivables('a','b',Status) :- Status =  "acceptable",!.
out_receivables(mismanaged) :-
     output("The receivables","are","probably being mismanaged"),!.
out_receivables(low) :-
     output("The receivables quality","could be","low"),!.
out_receivables(delinquent) :-
     output("The receivables","are","delinquent"),!.
out_receivables(acceptable) :-
     output("The receivables","are","acceptable"),!.
solution_receivables(mismanaged) :-
     output("Solution","could be factor","receivables"),
     output("Solution","could be increase","cash sales"),
     output("Solution","could be limit","credit to better risks"),!.
solution_receivables(delinquent) :-
     output("Solution","could be accelerate", "receivables collections"),!.
solution_receivables(_) :- !.
/*****************************************/
/*  process( 9) : working capital        */
/*****************************************/
workingCapital(Status) :-
     write("  Sales to working capital ratio is\n\n"),
     write("    a: high and positive\n"),
     write("    b: average and positive\n"),
     write("    c: low and positive\n"),
     write("    d: negative\n"), readchar(Response),
     act_workingCapital(Response,Status).
act_workingCapital('a',Status) :- Status = "insufficient",!.
act_workingCapital('b',Status) :- Status = "adequate",!.
act_workingCapital('c',Status) :- Status = "inadequate",!.
act_workingCapital('d',Status) :- Status =  "crunch",!.
act_workingCapital(_,_) :- beep,nl,
     write("   Invalid response. Try again."), readchar(Response),
     act_workingCapital(Response,Status),!,nl.
out_workingCapital(insufficient) :-
     output("Working capital","indicates","insufficient working capital"),!.
out_workingCapital(adequate) :-
     output("Working capital","is","adequate and properly managed"),!.
out_workingCapital(inadequate) :-
     output("Working capital","is","not adequately employed"),!.
out_workingCapital(crunch) :-
     output("Working capital","indicates","cash crunch"),!.
solution_workingCapital(adequate) :- !.
solution_workingCapital(insufficient) :-
     output("Solution","could be obtain","revolving bank credit"),!.
solution_workingCapital(inadequate) :-
     output("Solution", "could be redeploy", "excess capital more profitably"),!.
solution_workingCapital(crunch) :-
     output("Solution", "could be", "immediate injection of cash required"),!.
/*********************************************************************/
/*  process(10) : depreciation amortization and depletion expenses  */
/*********************************************************************/
depreciation(Status) :-
     write("  Percent depreciation amortization\n",
           "  and depletion expenses to net sales is \n\n"),
     write("    a: below average\n"),
     write("    b: average or less\n"),
     readchar(Response),
     act_depreciation(Response,Status).
act_depreciation('a',Status) :- Status = "below",!.
act_depreciation('b',Status) :- Status = "average",!.
act_depreciation(_,_) :- beep,nl,
```

```
     write("   Invalid response. Try again."), readchar(Response),
     act_depreciation(Response,Status),!,nl.
out_depreciation(below) :-
     output("The depreciation amortization and depletion expenses\n",
            "     are","excessive"),!.
out_depreciation(average) :-
     output("The depreciation amortization and depletion expenses\n",
            "     are","acceptable"),!.
solution_depreciation(average) :- !.
solution_depreciation(below) :-
     output("Solution","could be increase", "amortization time"),!.
/*****************************************/
/*  process(11) : net worth              */
/*****************************************/
netWorth(Status) :-
     write("  Total debt to tangible net worth is\n\n"),
     write("    a: higher than average\n"),
     write("    b: average or lower\n"),
     write("    c: negative\n"),
     readchar(TotalDebt), verify_abc(TotalDebt),nl,
     write("  Fixed assets to tangible net worth is\n\n"),
     write("    a: higher than average\n"),
     write("    b: positive and average or lower\n"),
     write("    c: negative\n"),
     readchar(FixedAsset), verify_abc(FixedAsset),
     act_netWorth(TotalDebt,FixedAsset,Status).
act_netWorth('a', _ ,Status) :- Status = "undercapitalization",!.
act_netWorth( _ ,'a',Status) :- Status = "undercapitalization",!.
act_netWorth('b','b',Status) :- Status = "safetycushion",!.
act_netWorth('c','c',Status) :- Status = "zero",!.
act_netWorth( _ , _ ,Status) :- Status = "unknown",!.
out_netWorth(safetycushion) :-
     output("The net worth of tangible assets\n","     indicates",
          "financial safety cushion for creditors"),!.
out_netWorth(zero) :-
     output("The net worth of tangible assets", "is", "zero"),!.
out_netWorth(undercapitalization) :-
     output("The net worth of tangible assets", "indicates\n",
            "     lenders are less safe due to undercapitalization"),!.
out_netWorth(unknown) :-
     output("I can't evaluate net worth of tangible assets\n",
            "     because","I have no knowledge for this case"),!.
solution_netWorth(zero) :-
     output("Solution","could be increase","equity investment"),!.
solution_netWorth(undercapitalization) :-
     output("Solution","could be increase","equity investment"),!.
solution_netWorth(_) :- !.
/*********************************************/
/*  process(12) : lease and rental expenses  */
/*********************************************/
lease(Status) :-
     write("  Percent lease and rental expenses to net sales is \n\n"),
     write("    a: average or less\n"),
     write("    b: greater than average\n"),
     readchar(Response),
     act_lease(Response,Status).
act_lease('a',Status) :- Status = "normal",!.
act_lease('b',Status) :- Status = "excessive",!.
act_lease(_,_) :- beep,nl,
     write("   Invalid response. Try again."), readchar(Response),
     act_lease(Response,Status),!,nl.
out_lease(normal) :-
     output("The lease and rental expenses","are","normal"),!.
out_lease(excessive) :-
     output("The lease and rental expenses","are","excessive"),!.
```

```
solution_lease(normal) :- !.
solution_lease(excessive) :-
     output("Solution","could be find","cheaper rent"),
     output("Solution","could be renegotiate","lease to lower payments"),!.
/*****************************************/
/*  process(13) : officer's compensation  */
/*****************************************/
compensation(Status) :-
     write("  Percent officer's compensation to net sales is\n\n"),
     write("    a: higher than average\n"),
     write("    b: average\n"),
     write("    c: lower than average\n"),
     readchar(Response),
     act_compensation(Response,Status).
act_compensation('a',Status) :- Status = "higher",!.
act_compensation('b',Status) :- Status = "average",!.
act_compensation('c',Status) :- Status = "lower",!.
act_compensation(_,_) :- beep,nl,
     write("   Invalid response. Try again."), readchar(Response),
     act_compensation(Response,Status),!,nl.
out_compensation(higher) :-
     output("The officer's compensation","is","above normal"),!.
out_compensation(average) :-
     output("The officer's compensation","is","normal"),!.
out_compensation(lower) :-
     output("The officer's compensation","is","below normal"),!.
solution_compensation(higher) :-
     output("Soulution","could be reduce","officer's compensation"),!.
solution_compensation(_) :- !.
/*****************************************/
/*  process(14) : overall performance  */
/*****************************************/
netProfit(NetProfit) :-
     write("  Net profit as percent of sales is:\n\n"),
     write("    a: acceptable\n"),
     write("    b: unacceptable\n"),
     readchar(Response),verify_ab(Response),
     Response = 'a',NetProfit = "acceptable",!.
act_overall(effective,undercapitalization,acceptable,acceptable,compatible) :- !.
act_overall(effective,acceptable,acceptable,acceptable,compatible) :- !.
act_overall(ineffective, _ ,unacceptable, _ ,incompatible) :- !.
act_overall( _ ,deficit, _ , _ ,incompatible) :- !.
act_overall( _ ,underemployment, _ , _ ,incompatible) :- !.
act_overall( _, _, _, _, unknown) :- !.
out_overall(compatible) :-
     output("Overall performance","is","comparable to the competition"),!.
out_overall(incompaitble) :-
     output("Overall performance","is","not comparable to the competition"),!.
out_overall(unknown) :-
     output("I can't evaluate overall performance\n",
            "     because","I have no knowledge fot this case"),!.
/*****************************************/
/*    Toolbox for Expert Loan Officer    */
/*****************************************/
output(S,V,Cor0) :- nl, sound(5,165), sound(5,220), sound(5,294), sound(5,392),
       write("   -> ", S," ",V," ",Cor0,".\n").

pauser :-
   nl,
   write("    -- Hit any key to continue. --\n\n"),
   readchar(_).
verify_ab('a') :- !.
verify_ab('b') :- !.
verify_ab(_)  :- beep, nl,
    write("    Invalid response. Try again."), readchar(Response),
```

```
    verify_ab(Response),!,nl.
verify_abc(Response) :- Response >= 'a', Response <= 'c',!.
verify_abc(_)  :- beep, nl,
    write("    Invalid response. Try again."), readchar(Response),
    verify_abc(Response),!,nl.
max(C1,C2,C2) :- C2 >= C1,!.
max(C1,C2,C1) :- C2 <  C1,!.
min(C1,C2,C2) :- C2 <= C1,!.
min(C1,C2,C1) :- C2 >  C1,!.
```

각 주

1) 본 시나리오와 이후에 논의되는 룰(rule)은Turban의 「Decision Support and Expert Systems - Managerial Perspectives」에 나오는 시나리오를 우리의 실정에 맞게 정한 것임을 밝혀둔다

2) 본 프로그램은 Turbo-Prolog Toolbox에 있는 Menu.pro를 참고하였음을 밝혀둔다

참고 문헌

박상범 譯, 제 5세대 컴퓨터언어 PROLOG 입문 - 지식정보처리의 서곡 -, 기전연구사 1985

이동만,서창교,이영숙, Turbo-Prolog를 이용한 전문가시스템 쉘의 원형개발, 경상집, 21권 3호, 1993.

Bonnet,A., Haton,J-P. and Truong-Ngoc,J-M., Expert Systems, Prentice Hall, 1988

Mockler,R.J., Knowledge-Based Systems for Management Decisions, Prentice Hall,1989

Robinson,P.R., Using Turbo Prolog, Osborne McGraw-Hill, 1987

Turban,E.,Decision Support and Expert Systems -Managerial Perspectives- Macmillan Publishing Company, 1988

Tyran,C.K., and George,J.F., "The Implementation of Expert Systems:A Survey of Successful Implementations", DATA BASE , Vol. 24 No. 1, 1993 Winter,pp. 5-15

OWaterman,D.A., A Guide to Expert Systems, Addison-Wesley Publishing Company, 1986