# 커뮤니티 컴퓨팅을 위한 규칙기반 지식 표현 방법*

권혁준° 이건수 김민구
아주대학교 정보통신 전문대학원
{neoera°, lks7256, minkoo}@ajou.ac.kr

# The Knowledge Representation Techniques for Community Computing

Hyouckjun Kwon°, Keonsoo Lee, Minkoo Kim
Graduate school of information and communication, Ajou university

요 약

커뮤니티 컴퓨팅이란 유비쿼터스 컴퓨팅 환경에서 하나의 컴퓨팅 요소를 통해 해결하기 어려운 문제들을 커뮤니티란 메타포를 통해 해결하고자 하는 방법론이다. 커뮤니티는 어떤 목표를 달성하기 위한 컴퓨팅 요소들의 조직으로 기존의 방법론들이 어떤 문제를 해결하기 위해 하나의 조직에 컴퓨팅 요소들이 고정적으로 존재했던 반면 커뮤니티는 컴퓨팅 요소가 동적일 뿐만 아니라 조직 자체도 동적인 특징을 지닌다. 본 논문에서는 기존의 커뮤니티 컴퓨팅에서 필요한 롤(Role)과 정책(Policy)을 기술하는 방법과 구현에서 나타나는 문제점을 지적하고, 대안으로 규칙기반 방법론을 역할과 정책의 기술 방법으로 제시한다. 그리고 이를 실제 구현하기 위한 규칙기반 시스템을 소개하고 기존의 커뮤니티 컴퓨팅의 역할과 정책을 규칙형태로 표현하여 규칙기반 시스템을 이용했을 때의 실현성과 효율성을 검증해 보았다.

## 1. Introduction

유비쿼터스 컴퓨팅은 사람을 포함한 현실 공간에 존재하는 모든 대상물들을 기능적·공간적으로 연결해 사용자에게 필요한 정보나 서비스를 즉시에 제공할 수 있는 기반 기술로서, 어떤 목표(Goal)를 달성하기 위해 서로 상호작용하는 분산 컴퓨팅으로 볼 수 있다[1,2]. 유비쿼터스 컴퓨팅 환경에서 주어지는 목표들 중에는 하나의 개체를 통해 달성하기 어려운 것들이 있다. 기존의 분산 컴퓨팅 또는 유비쿼터스 컴퓨팅을 고려한 에이전트 기반 모델링 기법들은 목표를 달성하기 위한 조직을 동적으로 생성하지 못하거나, 조직 내에서의 각 멤버의 롤(Role)을 기술하는 것을 충분히 제공하지 못한다[3,4,5]. 그러나 커뮤니티 컴퓨팅(Community Computing)에서는 동적으로 조직을 구성하고 조직의 구성요소의 롤을 기술하는 방법을 제공한다.

커뮤니티 컴퓨팅은 멀티에이전트 기반의 유비쿼터스 시스템에서 커뮤니티라는 메타포를 통해 동적인 임무지향 조직(Mission-oriented organization)의 관리와 협력(Cooperation)을 위한 개념모델이다. 커뮤니티 컴퓨팅은 기존의 멀티 에이전트 모델들이 지원하지 못하는 유비쿼터스 컴퓨팅의 동적인 특성을 임무지향의 커뮤니티와, 롤, 목표, 그리고 롤들이나 커뮤니티들 간의 협력과 같은 특징들을 통해 반영하고 있다. 커뮤니티 컴퓨팅에서 커뮤니티는 항상 정적으로 존재하지 않고 목표가 발생하여 이를 달성할 필요가 있을 경우에, 이를 위한 임무지향의 커뮤니티가 구성되고 목표의 달성 이후 사라지는 동적인 존재이다. 또한 멤버 역시 항상 존재하는 것이 아닌 동적으로 배치되는 컴퓨팅 요소로 고려된다. 커뮤니티 컴퓨팅에서는 컴퓨팅 요소들이 존재하는 환경을 하나의 소사이어티로 정의하고 소사이어티의 멤버들이 목표(Goal)를 달성하기 위해 동적으로 커뮤니티(Community)를 형성하고, 커뮤니티를 형성한 멤버들 간의 상호작용을 통해 목표를 달성한다. 커뮤니티 컴퓨팅 모델에는 전체 유비쿼터스 환경을 나타내는 소사이어티와 소사이어티의 멤버들, 소사이어티에서 발생할 수 있는 목표들, 어떤 목표를 달성하기 위한 커뮤니티와 커뮤니티를 구성하기 위한 소사이어티의 멤버들 그리고 커뮤니티 멤버들 간의 상호작용에 대해 기술되어있다. 이를 통해 유비쿼터스 환경의 동적인 컴퓨팅 요소들의 특성을 반영하는 컴퓨팅 환경을 구현할 수 있게 된다.

기존의 커뮤니티 컴퓨팅에서 롤을 기술하는 언어는 어떤 멤버에게 롤의 이름을 미리 주고 단순하게 비교하는 방법으로 기술됐다. 이 경우 멤버들과 커뮤니티들이 다양해지는 경우에 문제가 된다. 또한 커뮤니티나 멤버들 간에 충돌을 해결하기위한 정책에 관한 연구가 있었지만 해당연구에 사용된 Ponder 정책 언어를 위한 수행기 등은 제공 중단으로 인해 이용이 불가능한 상태이다. 따라서 커뮤니티 컴퓨팅에서 롤과 정책을 표현하기 위한 언어와 실제 구현하는 방법론에 대한 연구가 필요하다[6,7,8].

본 논문에서는 커뮤니티 컴퓨팅의 롤과 정책을 표현하기 위해 지식표현의 방법 중 하나인 규칙기반 방법론과 이를 구현하기 위한 규칙기반 시스템에 대해 소개한다. 또한 기존의 커뮤니티 컴퓨팅의 롤과 정책을 규칙기반 방법과 규칙기반 시스템에 적용하여 그 실현성과 효율성을 검증해 보았다.

## 2. Background

*"본 연구는 21세기 프론티어 연구개발사업의 일환으로 추진되고 있는 정보통신부의 유비쿼터스컴퓨팅및네트워크원천기반기술개발사업의 지원에 의한 것임"

### 2.1 커뮤니티 컴퓨팅에서 롤

커뮤니티 컴퓨팅에서 롤은 소사이어티에서 목표 달성을 위해 커뮤니티를 구성할 때 필요한 멤버의 속성을 기술한 것이다. 그리고 롤은 커뮤니티 템플릿에 컴퓨팅 모델 언어로 롤을 수행할 수 있는 멤버에게 필요한 속성과 함께 기술되어있다. 현재의 커뮤니티 컴퓨팅은 커뮤니티 템플릿에 기술된 롤을 수행할 수 있는 멤버를 표현하는 멤버 템플릿에 미리 어떤 롤을 할 수 있는 지를 표현하는 타입을 부여하여 커뮤니티가 생성될 때에 커뮤니티 템플릿의 롤과 타입을 비교하여 적합한 멤버를 커뮤니티의 멤버로 불러온다. 하지만 이러한 방법의 경우 새로운 멤버 템플릿을 기술할 때에는 항상 모든 커뮤니티에서 어떤 롤이 기술되어있는지를 고려하여 새로 만드는 멤버 템플릿의 멤버 타입을 어떻게 설정할 것인지를 고려해야 한다. 또한 마찬가지로 새로운 커뮤니티 템플릿을 추가할 때에 기존의 모든 멤버 템플릿들을 일일이 확인하여 멤버 타입을 추가 또는 변경시켜야하는 단점이 있다. 작은 규모의 소사이어티에서는 새로 커뮤니티나 멤버의 템플릿을 추가하는 것이 문제가 되지 않지만 소사이어티의 규모가 커지게 되는 경우 새로운 커뮤니티나 멤버 템플릿을 추가하게 되는 많은 시간이 걸리게 되어 효율적인 시스템 개발을 어렵게 만든다.

### 2.2 커뮤니티 컴퓨팅에서 정책

커뮤니티 컴퓨팅에서 정책은 커뮤니티 컴퓨팅에서 구성요소들이 실행될 때 상호작용하는 요소로들 간의 충돌을 해결하기 위한 방법으로 소사이어티, 커뮤니티 그리고 멤버 이 세 가지 수준에서의 충돌이 일어날 수 있다고 도출되어 이를 해결하기 위한 정책이 제시되었다[6]. 그러나 현재 커뮤니티 컴퓨팅을 위한 정책을 표현하기 위한 언어가 존재하지 않고, 기존의 커뮤니티 컴퓨팅의 정책에 관한 연구에서 실제 구현을 위한 방법론의 예로 제시된 Ponder 정책 언어는 Ponder를 이용할 수행기가 제공되고 않고 있다[8]. 따라서 Ponder를 위한 수행기를 구현하거나 정책 표현 및 수행을 위한 새로운 방법이 연구되어야할 필요가 있다. [표 1]은 각 수준에서 일어날 수 있는 충돌과 그에 대한 정책 중 대표적인 것을 뽑아본 것이다.

[표 1] 커뮤니티 컴퓨팅에서 충돌과 충돌에 대한 정책

| 수 준 | 구분 | 내 용 |
|---|---|---|
| 소사이어티 | 충 돌 | 서로 공존해서는 안되는 커뮤니티의 생성에 따른 커뮤니티 간 충돌 |
| | 정 책 | 멤버으로부터 커뮤니티 생성 요청시 현재 존재하는 커뮤니티와 공존할 수 없는 커뮤니티의 경우 생성을 보류한다. |
| 커뮤니티 | 충 돌 | 다른 커뮤니티에 속한 멤버의 캐스팅으로 인한 롤 충돌 |
| | 정 책 | 다른 커뮤니티에 속해있는 멤버의 경우 캐스팅 대상에서 제외시킨다. |
| 멤버 | 충 돌 | 멤버의 기본 행동과 소사이어티 매니저로부터 전달 받은 계획의 상충 |
| | 정 책 | 기본 행동과 전달 받은 계획이 같이 수행될 수 있는 지의 여부를 판단하여 같이 수행될 수 없다면 기본행동을 중지하고 계획을 수행한다. |

### 2.3 규칙기반 방법과 추론

규칙은 복잡하고 빠르게 변화하는 환경에 의해 움직여지는 과정들을 묘사하는 자연스러운 방법이다. 일련의 규칙들은 제어의 흐름에 대한 상세한 깊은 지식을 요구하지 않고도 프로그램이 변화하는 자료에 어떻게 반응해야 하는지를 상술할 수 있다[9]. 또한 기존의 일반 프로그램에서는 제어의 흐름과 자료의 사용은 프로그램 코드에 의해 미리 결정지워지는 것에 반해 규칙기반 시스템은 실행시간에 규칙들을 추가할 수 있다. 이와 같은 특징을 지닌 규칙기반 방법론은 유비쿼터스 환경의 동적인 요소들을 반영하는 커뮤니티 컴퓨팅에서 사용되기에 적합한 방법론이라 할 수 있다.

규칙을 이용한 추론은 크게 전향추론과 후향추론이 있다. 전향추론은 이용 가능한 정보로부터 출발하여 적절한 결론을 얻는 방법으로 주어진 상황에 해당하는 사실에 의하여 조건부가 만족되는 규칙을 찾아 결론부를 수행하는 것을 한 사이클로 하여 모든 만족되는 규칙을 찾을 때까지 반복한다. 전향추론은 주어진 사실에서 유도되는 모든 것을 찾기 위한 문제에 적합하다. 후향추론은 어떤 목표를 하나 정한 후 이 목표가 성립하기 위한 조건들을 하나씩 찾는 방법으로 목표의 수가 많지 않은 경우 목표를 하나씩 선택해 가며 선택된 목표가 성립되기 위한 규칙이나 사실과 같은 여러 가지 조건들이 만족 되는가 조사할 때 효율적이다. 따라서 다양한 곳에서 일어날 수 있는 충돌 조건을 알기 위해서는 전향추론이, 커뮤니티에서 어떤 롤을 소사이어티의 멤버가 할 수 있는지를 알기 위해서는 후향추론이 적합하다.

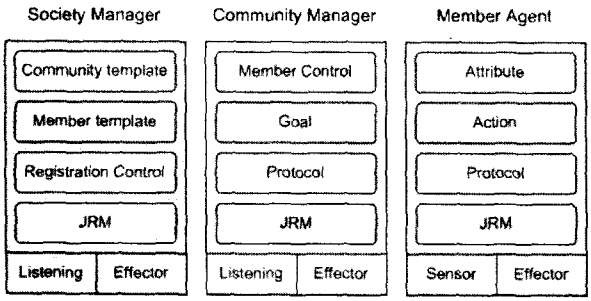
## 3. 커뮤니티 컴퓨팅에서의 Java Rule Master (JRM)

### 3.1 커뮤니티 컴퓨팅에서의 JRM의 위치

커뮤니티 컴퓨팅의 초기화와 커뮤니티 생성과 수행 그리고 커뮤니티 소멸의 단계를 거치며 진행된다. 소사이어티 매니저와 멤버들이 생성되고, 목표의 발생을 인지한 멤버들이 소사이어티 매니저에게 커뮤니티 생성을 요청한다. 소사이어티 매니저는 커뮤니티 매니저를 생성하고 목표를 전달한다. 커뮤니티 매니저는 자신이 가진 필요한 롤에 적합한 멤버들을 캐스팅하고 멤버들에게 수행할 계획을 전달하여 목표를 달성한다. 자신에게 주어진 모든 계획의 수행이 끝났다는 것을 인지한 멤버는 소사이어티 매니저에게 커뮤니티 소멸을 요청한다. 커뮤니티 소멸 후 각 멤버들은 초기상태로 돌아간다.

커뮤니티 컴퓨팅에서 JRM은 소사이어티를 관리하는 소사이어티 매니저와 목표를 달성하기 위해 생성된 커뮤니티를 관리하는 커뮤니티 매니저 그리고 소사이어티의 멤버에 각각 위치하여 필요에 따라 규칙기반 추론을 수행한다. [그림 1]은 커뮤니티 컴퓨팅에서 각 구성요소들의 구조를 나타낸 그림이다. 커뮤니티 컴퓨팅에서 JRM의 롤은 각 구성요소에 위치하여 규칙기반으로 기술된 롤과 정책을 추론을 통해 수행하는 것이다.

롤은 롤에 적합한지를 판단하기 위한 규칙의 형태로 커뮤니티 매니저만 가지고 있으며 커뮤니티 매니저가 소사




[그림 1] 커뮤니티 컴퓨팅 구성요소 구조도

이어티의 멤버을 캐스팅하기 전에 규칙으로 표현된 롤을 멤버로 전달한다. 멤버는 전달받은 규칙과 자신의 정보를 가지고 추론하여 적합한 경우 캐스팅을 수행한다. 다음은 교통사고를 다루는 커뮤니티의 예에서 환자와 구급차 그리고 구급대원의 롤을 JRM-RL로 표현한 예이다[표 2].

[표 2] JRM-RL로 표현된 커뮤니티 컴퓨팅 롤의 예

| 환자의 롤 |
|---|
| (defrule ROLES_PATIENT<br>(people (ID ?x) (Bodycondition ABNORMAL) (STATUS CLASHED))<br>=> (doCasting (Member (ID ?x))) |
| 구급차의 롤 |
| (defrule ROLES_AMBULANCE<br>(ambulance (ID ?x) (OperationMode IDLE))<br>=> (doCasting (ambulance (ID ?x))) |
| 구급 대원의 롤 |
| (defrule ROLES_PARAMEDIC<br>(person (ID ?x) (ParamedicLicense ACQUIRED))<br>=> (doCasting (person (ID ?x))) |

[표 3] JRM-RL로 표현된 커뮤니티 컴퓨팅 정책의 예

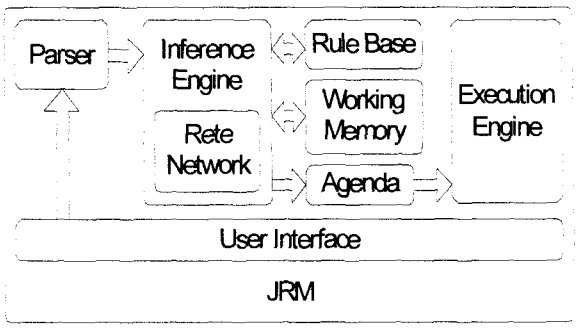
| 커뮤니티 생성 : 멤버로부터 커뮤니티 생성 요청시 현재 존재하는 커뮤니티와 공존할 수 없는 커뮤니티의 경우 생성을 보류한다. |
|---|
| (COMMUNITY_CREATION<br>(RequestedCommunity (Goal ?x) (ID ?y))<br>(RegisteredCommunity (Goal ?z))<br>(OppositeGoal (Goal ?x) (Goal ?z))<br>=> remove (RequestedCommunity (ID ?y))<br>assert (SuspendedCommunity (ID ?y) (Goal ?x))) |
| 커뮤니티에서 멤버 캐스팅 : 다른 커뮤니티에 속해있지 않는 구성원의 경우 캐스팅을 수행한다. |
| (MEMBER_CASTING<br>(Member (ID ?x) (CommunityID ?y))<br>(?y = NULL)<br>=> (doCasting (Member (ID ?y))) |
| 커뮤니티와 소사이어티 멤버 사이의 통신오류 : 캐스팅 된 멤버와의 통신 문제 발생시 3회시도 후 멤버에 대한 캐스팅 작업을 다시 수행한다. |
| (COMMUNICATION_ERROR<br>(Member (ID ?x) (CommunicationErrorCount 3))<br>=> assert (rejectMember (ID ?x))) |

[표 3]는 기존의 커뮤니티 컴퓨팅을 위한 정책을 JRM에 적용하기 위하여 기존에 연구에서 도출된 정책 가운데 커뮤니티의 생성, 커뮤니티에서 멤버 캐스팅, 커뮤니티와 소사이어티 멤버 사이의 통신오류에 대한 정책을 JRM-RL을 이용해 기술해 본 예이다[6].

롤은 커뮤니티 매니저만 가지는 반면 정책은 각 구성요소들이 자신에 해당하는 정책을 규칙의 형태로 가진다. 이와같이 기술된 정책을 각 구성요소가 자신의 Rule Base에 가지고 있다가, 각 멤버가 각자의 계획을 따라 통신을 통해 수행하면서 자신의 정보를 전달하면 해당정보가 정책규칙에 적용되어 정책에 따라 계획의 수행을 진행한다.

### 3.3 JRM의 시스템 구조

JRM은 Rete 알고리즘에 기반한 양방향 추론엔진으로 UI를 통해 규칙이나 사실을 JRM-RL(Rule Language)를 이용해 규칙이나 사실을 작성할 수 있도록 하고 있으며, 미리 만들어둔 파일로부터 규칙이나 사실을 가져올 수 있다[10]. [그림 2]는 JRM의 시스템 구조도이다.




[그림 2] JRM 시스템 구조도

구성 요소는 다음과 같다.

- Parser : 사용자의 입력을 문법에 따라 규칙이나 사실로 변환한다.
- 추론엔진(Inference Engine) : 전체적인 시스템을 운영한다.
- Rule Base : 규칙을 관리한다.
- Working Memory(WM) : 사실을 관리한다.
- Rete Network : 규칙과 사실의 매칭상태를 저장한다.
- Agenda : 추론엔진을 통해 선택된 최종적으로 수행할 규칙을 가진다.
- Excution Engine : Agenda로부터 규칙의 결과부를 얻어와 수행한다.

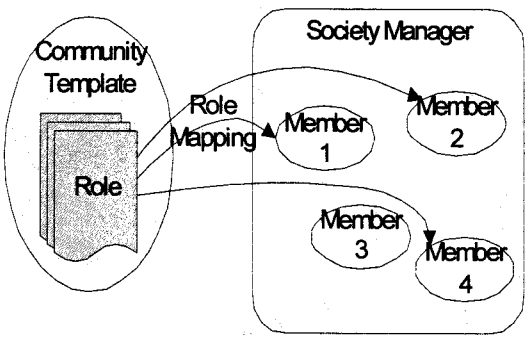
JRM의 수행은 UI를 통해 입력된 내용은 파서를 통해 Rule Base나 WM에 저장 또는 수정되며, 그 내용은 Rete Network에 반영되어 Network이 수정되거나 어떤 규칙들이 만족된 경우 Agenda에서 한 규칙을 선택하여 Execution Engine에 넘겨준다.

## 4. 시뮬레이션

앞서 제안한 방법을 실험하기 위해 시뮬레이션을 선택하였고, 시뮬레이션을 위한 시나리오는 다음과 같다.

한적한 시골길에서 자동차가 논두렁에 빠져 견인을 필요로 한다. 주변에 있는 사물들에게 롤을 수행할 수 있는지를 요청하고 주변의 트럭이 견인차의 롤을 수행하여 문제를 해결한다.

이와 같은 시뮬레이션에서 커뮤니티를 구성하기 위해 커뮤니티 템플릿에 정의된 롤과 소사이어티의 멤버를 매핑을 하는 과정을 규칙을 이용한 추론을 통해 수행하였고 이 중 피견인차와 견인차를 매핑하는 규칙은 [표 4] 와에 나와 있다. 시뮬레이션이 진행하는 동안 필요한 정책과 롤들은 JRM-RL을 통해 기술하였고, 다음은 커뮤니티에서 멤버를 캐스팅하는 모습을 나타내는 개념도와 시뮬레이션을 위한 규칙들과 사실들의 일부이다[그림 3, 표 4].




[그림 3] 커뮤니티 컴퓨팅 롤 매핑 개념도

[표 4] 시뮬레이션을 위해 JRM-RL로 표현한 롤들과 멤버들의 정보

| | |
|---|---|
| 구조 커뮤니티를 위한 롤 | 피견인차<br>(defrule WreckedCar<br>(Car (ID ?x) (STATUS ABNORMAL))<br>=><br>(doCasting (Car (ID ?x))) |
| | 견인차<br>(defrule WreckerCar<br>(Car (ID ?x) (STATUS IDLE) (> HP 110))<br>=><br>(doCasting (Car (ID ?x))) |
| 주변 사물들의 정보 | 경운기<br>(Car (ID T001) (STATUS IDLE) (HP 55)) |
| | 트럭<br>(Car (ID T002) (STATUS IDLE) (HP 120)) |

JRM을 통한 커뮤니티 구성 시나리오 시뮬레이션을 통해 우리는 다음과 같은 이점을 확인 수 있었다. 기존의 커뮤니티 컴퓨팅에서 롤의 개념을 이용하기 위해서 멤버 템플릿에 고정적으로 롤을 기술해야 했던 것과는 달리, 규칙기반으로 롤을 기술한 경우 멤버 템플릿에 관계없이 다양한 컴퓨팅 요소들이 롤을 수행할 수 있었다.

## 5. 결론

커뮤니티 컴퓨팅 모델은 유비쿼터스 컴퓨팅 환경에서 문제를 해결하는 과정에서 동적으로 커뮤니티 구성을 통해 유비쿼터스 컴퓨팅 환경의 특성을 반영하고 있다. 커뮤니티를 구성하는 과정을 규칙기반 방법을 이용하는 경우 기존의 커뮤니티 롤과 컴퓨팅 요소들을 매핑하는 과정에서 보다 유연하고 동적인 매핑을 가능하게 함으로서 커뮤니티 컴퓨팅에서 목표로 하는 동적인 유비쿼터스 컴퓨팅 환경의 특성을 반영을 보다 나아지게 한다고 할 수 있다.

## 6. 참고문헌

[1] 우운택. "지능적 통합 공간의 탄생, 유비쿼터스 혁명" 마이크로소프트웨어 vol. 3, 2003, 410-413.

[2] Yuna Jung, Jungtae Lee, Minkoo Kim. "Multi-agent based Community Computing System Development with the Model Driven Architecture" AAMAS, 2006.

[3] Michael Wooldridge, Nicholas R.K, David K. "The Gaia Methodology for Agent-oriented Analysis and Design" *Autonomous Agents and Multi-Agent Systems*, 3, 2000, 285-312

[4] Franco Zambonelli, Hicholas R. Jennings, and Nichael Wooldridge. "Developing Multiagent Systems: The Gaia Methodology" ACM Transactions on Software Engineering and Methodology, Vol. 12, No.3, July 2003, 317-370

[5] Mohan Kumar, Behrooz A. Shirazi, Sajal K. Das, Byung Y. Sun, et. al. "PICO: A Middleware Framework for Pervasive Computing" *Pervasive Computing*, 1536-1268, 2003, 72-79

[6] 조용석, 정유나, 김민구. "커뮤니티 컴퓨팅을 위한 정책기반 충돌해결 방안연구" KCC 2006, 130-132

[7] Damianou, Nicodemos, Dulay, Naranker, Lupu, Emil, Sloman, Morris. "The Ponder Policy Specification Language" In Proc. Policy 2001 workshop on policies for distributed systems and networks, Bristol, UK, 29-31 Jan.2001, Springer-Verlag LNCS 1995.

[8] Site : Ponder - Policy Research Group, DoC, Imperial College http://www-dse.doc.ic.ac.uk/Research/policies/ponder.shtml

[9] Donald A. Waterman. "A guide to expert systems" Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1985

[10] Charles L. Forgy. "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem" Artificial Intelligence, 1982, pp.17-37.