KIPS Tr. Software and Data Eng.
Vol.5, No.5 pp.231~242 pISSN: 2287-5905

http://dx.doi.org/10.3745/KTSDE.2016.5.5.231

MRQUTER: A Parallel Qualitative Temporal Reasoner Using MapReduce Framework

Jonghoon Kim† · Incheol Kim††

ABSTRACT

In order to meet rapid changes of Web information, it is necessary to extend the current Web technologies to represent both the valid time and location of each fact and knowledge, and reason their relationships. Until recently, many researches on qualitative temporal reasoning have been conducted in laboratory-scale, dealing with small knowledge bases. However, in this paper, we propose the design and implementation of a parallel qualitative temporal reasoner, MRQUTER, which can make reasoning over Web-scale large knowledge bases. This parallel temporal reasoner was built on a Hadoop cluster system using the MapReduce parallel programming framework. It decomposes the entire qualitative temporal reasoning process into several MapReduce jobs such as the encoding and decoding job, the inverse and equal reasoning job, the transitive reasoning job, the refining job, and applies some optimization techniques into each component reasoning job implemented with a pair of Map and Reduce functions. Through experiments using large benchmarking temporal knowledge bases, MRQUTER shows high reasoning performance and scalability.

Keywords : Qualitative Temporal Reasoning, Disjunctive Relations, Composition Table, MapReduce, Parallel Temporal Reasoner

MRQUTER: MapReduce 프레임워크를 이용한 병렬 정성 시간 추론기

김 종 훈† · 김 인 철††

요 약

빠른 웹 정보의 변화에 잘 대응하기 위해서는, 사실과 지식이 실제로 유효한 시간과 장소들도 함께 표현하고 그들 간의 관계도 추론할 수 있도록 웹 기술의 확장이 필요하다. 본 논문에서는 그동안 소규모 지식 베이스를 이용한 실험실 수준의 정성 시간 추론 연구들에서 벗어나, 웹 스케일의 대규모 지식 베이스를 추론할 수 있는 병렬 정성 시간 추론기인 MRQUTER의 설계와 구현을 소개한다. Hadoop 클러스터 시스템과 MapReduce 병렬 프로그래밍 프레임워크를 이용해 개발된 MRQUTER에서는 정성 시간 추론 과정을 인코딩 및 디코딩 작업, 역 관계 및 동일 관계 추론 작업, 이행 관계 추론 작업, 관계 정제 작업 등 몇 개의 MapReduce 작업으로 나누고, 맵 함수와 리듀스 함수로 구현되는 각각의 단위 추론 작업을 효율화하기 위한 최적화 기술들을 적용하였다. 대규모 벤치마킹 시간 지식 베이스를 이용한 실험을 통해, MRQUTER의 높은 추론 성능과 확장성을 확인하였다.

키워드 : 정성 시간 추론, 이접 관계, 조합 표, 맵 리듀스, 병렬 시간 추론기

1. 서 론

웹(Web)을 이용하는 이용자의 수와 정보의 양이 기하급수적으로 증가함에 따라, 웹 이용자의 요구도 점차 더 다양해지고 복잡해지고 있다. 이에 따라 대규모 비-정형 웹 데이터로부터 이용자의 목적에 부합하는 정보만을 효율적으로 추출해내고, 이용자의 요구에 효과적으로 답할 수 있는, 시맨틱 웹(semantic web) 기술에 관한 연구가 최근 몇 년간 매우 활발히 진행되어 왔다. 시맨틱 웹 기술은 응용 영역에 필요한 기초 개념(concept)들과 그들 간의 관계(relationship), 공유 어휘(common vocabulary), 추론 규칙(inference rule) 등을 체계적으로 정의한 온톨로지 지식(ontological knowledge)을 이용함으로써, 프로그램 스스로 웹 콘텐츠의 의미(semantics)를 올바르게 이해하고 작업을 수행할 수 있도록 해준다. 대표적인 온톨로지 표현 언어로는 RDF(Resource Description

※ 본 연구는 산업통상자원부의 재원으로 기술혁신사업의 지원을 받아 수행한 연구 과제(No. 10060086, 개인 서비스용 로봇을 위한 지능-지식 집약·개방·진화형 로봇지능 소프트웨어 프레임워크 기술 개발)입니다.
※ 이 논문은 2015년도 한국정보처리학회 춘계학술발표대회에서 '맵리듀스 프레임워크를 이용한 대용량 시간 추론기 설계 및 구현'의 제목으로 발표된 논문을 확장한 것임.
† 준 회 원 : 경기대학교 컴퓨터과학과 석사과정
†† 종신회원 : 경기대학교 컴퓨터과학과 교수
Manuscript Received : January 11, 2016
Accepted : February 17, 2016
* Corresponding Author : Incheol Kim(kic@kyonggi.ac.kr)

Framework), OWL(Web Ontology Language)[1] 등이 주로 이용되고 있다.

한편, 최근 공학 기술 분야, 의학 및 신약 기술 분야, 보안 기술 분야 등을 비롯해 다양한 응용 분야들에서 학문적, 기술적 발전 속도가 매우 빨라져, 매일 수많은 새로운 사실들과 발견들이 웹을 통해 발표되고 있다. 따라서 이러한 빠른 웹 정보의 변화에 잘 대응하기 위해서는, 시맨틱 웹 기술이 단순히 단편적인 사실과 지식만을 표현하는데서 벗어나, 해당 사실과 지식이 실제로 유효한 시간(time)과 장소(location)들도 함께 표현하고 그들 간의 관계도 추론할 수 있도록 기술적 확장이 필요하다.

시간 정보 표현과 연관된 선행 연구들은 크게 시간 지식(temporal knowledge) 표현법, 시간 질의(temporal query) 표현법, 시간 순서 관계(temporal ordering relationship) 표현법 등으로 나눌 수 있다. 시간 정보를 표현할 수 있도록 RDF와 OWL 지식 표현법을 확장하고자 하는 대표적인 연구들로는 tRDF[2], tOWL[3], stRDF[4], STOWL5] 등이 있다. 반면에, 표준 질의 언어인 SPARQL을 확장한 시간 질의 표현 연구들로는 T-SPARQL[6], stSPARQL[4] 등이 있다. 기존의 많은 시간 지식 표현법이나 시간 질의 표현법에 관한 연구들에서는 시간 자체는 하나의 시간 점(time point)이나, 혹은 한 쌍의 시작 점(starting point)과 종료 점(ending point)으로 구성되는 하나의 시간 간격(time interval)으로 표현하였다. 그리고 이러한 형태의 시간 지식과 질의를 처리하는 시스템에서는 질의에 대한 답을 얻기 위해, 시간 점과 시간 간격을 나타내는 시간 데이터에 관한 연산(computation of temporal data)에 주로 의존하는 정량적 시간 추론(quantitative temporal reasoning)을 많이 수행하였다.

한편, 기존의 시간 순서 관계 표현법들은 주로 Kautz[7]의 점 대수(point algebra) 이론이나 Allen[8]의 간격 대수(interval algebra) 이론을 토대로 두 시간 점(time point)들, 혹은 시간 간격(time interval)들 사이의 순서 관계를 미리 정의해놓은 관계 서술자(relation property)들을 이용하여 정성적으로 표현하였다. 그뿐만 아니라 새로운 두 이벤트(event) 사이의 시간 순서 관계를 파악하기 위해서는 각 이벤트의 발생 시간 데이터를 이용한 연산에 의존하기 보다는, 이미 알고 있는 이벤트들의 정성적 시간 순서 관계들을 이용하는 정성적 시간 추론(qualitative temporal reasoning)을 주로 수행하였다. 정량적 시간 표현과 추론은 실시간성과 정밀성을 요구하는 생산 자동화, 제어 시스템과 같은 응용 분야에서 더 효과적으로 이용될 수 있는데 반해, 자연어 이해(natural language understanding), 질의-응답 시스템(question-answering system), 의사 결정 지원 시스템(decision support system)과 같은 특정 응용 분야들에서는 이벤트들 간의 정성적 순서 관계를 표현하고 추론하는 정성적 시간 추론이 더 효과적으로 이용될 수 있다.

본 논문에서는 그동안 GQR[9], SOWL[10], CHRONOS[11, 12] 등과 같이 소규모 지식 베이스를 이용한 실험실 수준의 정성 시간 추론 알고리즘 연구들을 확장하여, 웹 스케일의 정성 시간 추론을 수행할 수 있는 병렬 정성 시간 추론기인 MRQUTER의 설계와 구현을 소개한다. 먼저 2장에서는 정성 시간 추론에 관한 선행 연구들을 소개하고, 3장에서는 본 연구의 기초가 되는 Allen의 정성적 시간 순서 관계 표현과 추론 규칙들, 그리고 이들을 구현하기 위한 최적화된 이접 관계 조합 표를 정의한다. 이어서 4장에서는 Hadoop MapReduce[13] 병렬 프로그래밍 프레임워크를 이용한 병렬 정성 시간 추론 알고리즘을 소개하고, 5장에서는 병렬 정성 시간 추론 시스템인 MRQUTER의 구현과 성능 분석 실험 결과에 대해 설명한다. 끝으로 6장에서는 결론과 향후 연구를 소개한다.

2. 관련 연구

간격 대수(interval algebra)라 불리는 Allen의 연구[8]에서는 서로 다른 두 개의 시간 간격들(time intervals) 사이의 가능한 모든 순서 관계들을 표현할 수 있는 총 13 가지의 관계 서술자(relation property)들을 제시하였고, 이들을 기초로 새로운 시간 순서 관계를 정성적으로 추론해내기 위한 추론 규칙(inference rule)들을 하나의 조합 표(composition table) 형태로 정의하였다. 반면에, 점 대수(point algebra) 이론을 담고 있는 Kautz[7]의 연구에서는 서로 다른 두 개의 시간 점들(time points) 사이의 순서 관계를 표현하기 위해 before, equal, after 등 총 3 가지 관계 서술자들과 이들을 이용한 정성 추론 규칙들을 제시하였다. Allen의 간격 대수 이론과 Kautz의 점 대수 이론은 그동안 사실상 정성 시간 추론의 기초를 이루어 왔다. 따라서 이 이론들을 토대로 다양한 정성 시간 추론기들이 개발되었는데, 대표적인 것들로는 GQR[9], SOWL[10], CHRONOS[11, 12] 등이 있다.

GQR[9]은 제약 네트워크(constraint network) 기반의 일관성 검사(path consistency checking)를 통해, 다양한 정성 지식을 추론할 수 있는 범용 정성 추론기이다. C++로 구현된 GQR은 현재 간격 대수(interval algebra) 기반의 시간 추론 기능뿐만 아니라, RCC-5, RCC-8, CSD-9 이론 기반의 공간 추론 기능도 제공하고 있다. GQR은 범용성과 확장성을 위해, 텍스트나 XML문서로 명세한 새로운 정성 추론 규칙들을 쉽게 추론기에 반영할 수 있도록 설계하였다. GQR은 다양한 시간 및 공간 추론 규칙들을 적용할 수 있는 높은 범용성을 제공하고 있으나, 시간 추론 혹은 공간 추론에 특화된 추론기들에 비해 추론 성능이 낮고, RDF와 OWL을 기초로 하는 시맨틱 웹과는 독립적인 지식 표현 체계를 따르고 있다.

한편, SOWL[10]은 시맨틱 웹 온톨로지 언어인 OWL을 기초로 시간과 공간 지식을 표현하고, 추론할 수 있는 정성 시간-공간 추론기이다. 이 추론기에서는 트리플(triple) 형태의 OWL 지식 표현 한계를 벗어나 4차원 서술자(4-D fluent)와 다자 관계(N-ary relation)를 이용하여 시간 정보와 공간 정보를 표현하고자 시도하였다. SOWL에서는 Allen의 간격 대수 이론에 기초한 시간 추론 규칙들과 CSD-9과 RCC-8

이론에 기초한 공간 추론 규칙들을 모두 시맨틱 웹 규칙 언어(semantic web rule language)인 SWRL[14]로 표현하였다. 하지만, SOWL에서는 Allen의 간격 대수에서 조합 표형태로 제시한 시간 추론 규칙들을 모두 구현하지 못하고, 그 중 소수의 결정적 규칙들(deterministic rules)만을 구현함으로써 추론의 완전성을 만족하지 못하고 있다. 그뿐만 아니라, SOWL은 외부에서 개발된 SWRL 추론 엔진을 채용함으로써, 실제 SWRL 규칙들이 수행되는 세부 과정에 대한 최적화가 이루어지기 어려워 추론 성능의 한계를 보인다.

CHRONOS[11, 12]는 OWL 추론과 RCC-8 공간 추론 기능을 제공하는 PelletSpatial[15]을 변경하여, 공간 추론 기능 대신 간격 대수 기반의 시간 추론 기능을 제공할 수 있도록 만든 정성 시간 추론기이다. Java로 구현된 CHRONOS는 SOWL과는 달리, 비-결정적 규칙들(non-deterministic rules)을 포함해 간격 대수 조합 표에서 제시한 모든 시간 추론 규칙들을 구현하였다. 하지만 CHRONOS는 추론의 세부 과정들에 최적화가 충분히 이루어지지 않아, 비효율적인 연산을 많이 내포하고 있다. GQR, SOWL, 그리고 CHRONOS에 이르기까지 현재까지 개발된 거의 모든 정성 시간 추론기들은 단일 컴퓨터 실행 환경을 가정하고 개발되어, 이들은 공통적으로 웹 스케일의 대규모 지식 베이스를 추론하기에는 성능에 상당한 한계가 있다.

3. 시간 지식 표현과 추론 규칙

3.1 시간 지식 표현

정성 시간 추론기를 설계하기 위해서는 먼저 추론의 대상이 되는 시간 지식을 어떻게 표현할 것인지, 즉 시간 지식 표현 체계를 먼저 정의해야 한다. 본 논문에서 가정하는 시간 지식은 시맨틱 웹 표준 온톨로지 언어인 RDF와 OWL을 기초로 주어(subject), 술어(predicate), 목적어(object)들로 구성되는 트리플(triple) 형태로 표현되며, Allen의 간격 대수 이론[8]과 Kautz의 점 대수 이론[7]을 통합하고 확장하여 Fig. 1의 온톨로지와 같은 시간 지식 표현 체계를 가정한다. 먼저 문장(rdfs:Statement) 클래스는 사실(fact)들의 집합을 나타내며, 하나의 사실을 나타내는 각 문장은 RDF와 OWL의 형식에 따라 단순히 주어(subject), 술어(predicate), 목적어(object)들로 구성되는 트리플(triple) 형태로만 표현된다. 하지만, 본 온톨로지에서 각 문장은 해당 사실이 유효한 시간 정보를 부여해주기 위해 occursOn 프로퍼티를 통해 하나의 시간 이벤트(kgu:Time Event)와 연결될 수 있다. 한편, 하나의 시간 이벤트(kgu:Time Event)는 점 대수 이론과 같이 하나의 시간 점(kgu:Time Point)이 될 수도 있고, 하나의 시간 간격(kgu:Time Interval)이 될 수도 있다. 하나의 시간 간격(kgu:Time Interval)은 각각 kgu:tm_begin 프로퍼티와 kgu:tm_end 프로퍼티를 통해 시작 점(starting point)과 종료 점(ending point)에 해당하는 두 개의 시간 점(kgu:Time Point)들과 연결될 수 있다. 그리고 이 온톨로지의 가장 중요한 부분은 바로 두 개의 시간 이벤트(kgu:Time Event)들 사이의 순서 관계를 나타내는 13 가지의 시간 프로퍼티(Temporal Property)들이다. kgu:before, kgu:meets, kgu:overlaps 등과 같은 13 가지 서로 다른 시간 프로퍼티들은 Fig. 2와 같은 시간 이벤트들 사이의 순서 관계를 나타낸다. 예컨대, 시간 순서 관계 서술자 before는 이벤트 a의 종료 시점이 이벤트 b의 시작 시점보다 앞선 경우를 나타내기 위해 사용되며, 이러한 사실은 “a before b”와 같은 문장으로 표현한다. 또 다른 시간 순서 관계 서술자인 meet는 이벤트 a의 종료 시점과 이벤트 b의 시작 시점이 일치하는 경우를 나타내기 위해 사용되며, 이러한 사실은 “a meets b”와 같은 문장으로 표현한다. 본 연구에서는 시간 이벤트들 사이의 가능한 모든 순서 관계는 이와 같이 서로 다른 총 13 가지(before, meets, overlaps, starts, during, finishes, equals, after, met-by, overlapped-by, started-by, contains, finished-by)의 시간 순서 관계 서술자들을 이용하여 정성적으로 표현한다고 가정한다.

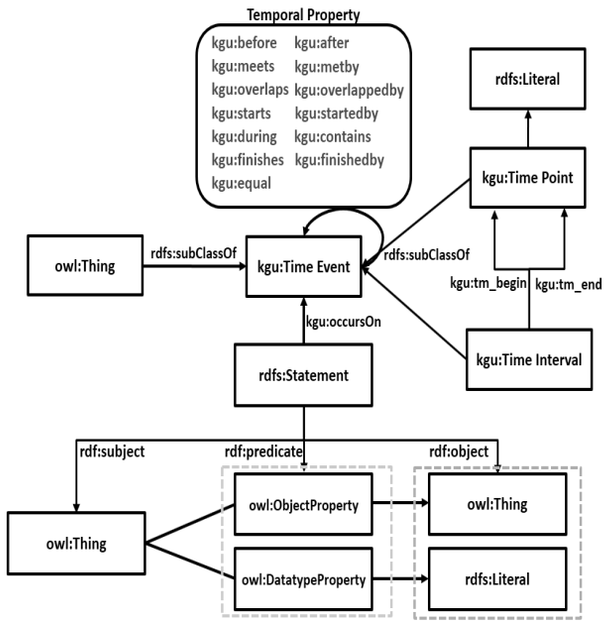

Fig. 1. Temporal Ontology

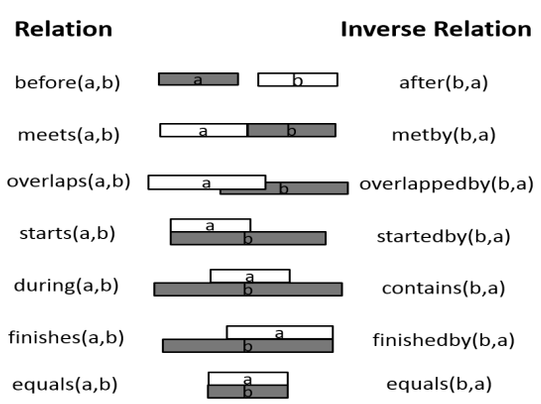

Fig. 2. Illustration of Possible Temporal Ordering Relationships between Two Events

3.2 시간 추론 규칙

앞서 소개한 시간 순서 관계 서술자들을 이용하여 이벤트들 간의 순서 관계들이 정성적으로 정의된 시간 지식 베이스가 존재할 때, 이 지식 베이스에 정의된 시간 순서 관계들 간에 모순(contradiction) 혹은 불일치성(inconsistency)이 존재하는지 검사하고, 나아가 이벤트들 간의 새로운 순서 관계를 발견해내는 것을 정성 시간 추론(qualitative temporal reasoning)이라고 한다. 본 논문에서는 Allen의 간격 대수 이론[8]에서 정의하고 있는 이행 관계 시간 추론 규칙(transitive rule)들 외에, 동일 관계 추론 규칙(equal rule)들, 역 관계 추론 규칙(inverse rule)들, 정제 추론 규칙(refining rule)들을 추가적으로 포함한 시간 추론 규칙들의 집합을 Table 1과 같이 정의한다.

Table 1. Temporal Reasoning Rules

Rules	If	then
Equal	s p o (p is a temporal property)	s equal s
		o equal o
Inverse	s p o (p inverse of q)	o q s
Transitive	x p_1 y y p_2 z	x $[p_1 \mid p_2 \mid \cdots \mid p_n]$ z
Refining	x $[p_1 \mid \cdots \mid p_m]$ y $\cap$ x $[p_1 \mid p_2 \mid \cdots \mid p_n]$ y	x $[p_1 \mid \cdots p_n]$ y

Table 1의 동일 관계 추론 규칙(equal rule)은 지식 베이스에 등장하는 한 이벤트는 자기 자신과 시간적 순서관계가 동일하다(equal)는 것을 의미한다. 따라서 지식베이스에 “s p o”라는 트리플 형태의 한 사실이 포함되어 있으면, 이때 주어(subject)와 목적어(object)를 구성하는 이벤트 s와 o는 각각 자기 자신과 시간적 순서 관계가 동일하므로, “s equal s”, “o equal o”와 같은 새로운 사실들을 유도해낼 수 있다. 한편, Table 1의 역 관계 추론 규칙(inverse rule)은 “s p o”라는 사실이 이미 지식베이스에 포함되어 있고, 서술자 p와 q가 서로 역 관계를 나타낸다면, 주어와 목적어의 위치가 뒤바뀐 “o q s”라는 새로운 사실을 유도해낼 수 있음을 의미한다.

Allen의 간격 대수[8]에서 정의하고 있는 이행 관계 추론 규칙(transitive rule)들은 Table. 2와 같은 하나의 조합 표(composition table)로 요약, 정리할 수 있다. Table. 2의 조합 표는 가로 행과 세로 열이 각각 before(<), after(>), during(d), contains(di), overlaps(o). overlappedby(oi), meets(m), metby(mi), starts(s), startedby(si), finishes(f), finishedby(fi) 등 총 13 가지 시간 순서 관계 서술자들로 구성되어 있다. 조합 표에 포함된 이행 관계 추론 규칙들을 해석하는 방법은 다음과 같다. 즉, 가로 행의 사실과 세로 열의 사실이 동시에 참(true)이라면, 이로부터 해당 행과 열이 교차하는 난에 열거된 새로운 사실들을 유도해낼 수 있음을 의미한다. 예컨대, 가로 행의 “a overlaps(o) b”라는 사실과 세로 열의 “b meets(m) c”라는 사실이 동시에 지식 베이스에 포함되어 있다면, 이것으로부터 해당 행과 열이 교차하는 곳에 before(<)라는 서술자가 열거되어 있으므로 “a before(<) c”라는 새로운 사실을 추론해낼 수 있다. 그리고 “a before c” 같이 두 이벤트 a와 c 사이의 시간적 순서 관계가 명확히 하나의 서술자 before(<)로 정의할 수 있을 때, 이러한 관계를 정의 관계(defined relation)라고 부른다. 이에 반해, “a overlaps(o) b”라는 사실과 “b during(d) c”라는 사

Table 2. Composition Table

B p2 C A p1 B	<	>	d	di	o	oi	m	mi	s	si	f	fi
“before” <	<	*	< o m d s	<	<	< o m d s	<	< o m d s	<	<	< o m d s	<
“after” >	*	>	> oi mi d f	>	> oi mi d f	>	> oi mi d f	>	> oi mi d f	>	>	>
“during” d	<	>	d	*	< o m d s	> oi mi d f	<	>	d	> oi mi d f	d	< o m d s
“contains” di	< o m di fi	> oi di mi si	o oi d di = f fi s si	di	o di fi	oi di si	o di fi	oi di si	di fi o	di	di sioi	di
“overlaps” o	<	> oi di mi si	o d s	< o m di fi	< o m	o oi d di = f fi s si	<	oi di si	o	di fi o	d s o	< o m
“overlapped-by” oi	< o m di fi	>	oi d f	> oi mi di si	o oi d di = f fi s si	> oi mi	o di fi	>	oi d f	oi >mi	oi	oi di si
“meets” m	<	> oi mi di si	o d s	<	<	o d s	<	f fi =	m	m	d s o	<
“met-by” mi	< o m di fi	>	oi d f	>	oi d f	>	s si =	>	d f oi	>	mi	mi
“starts” s	<	>	d	< o m di fi	< o m	oi d f	<	mi	s	s si =	d	< m o
“started-by” si	< o m di fi	>	oi d f	di	o di fi	oi	o di fi	mi	ssi =	si	oi	di
“finishes” f	<	>	d	> oi mi di si	o d s	> oi mi	m	>	d	> oi mi	f	f fi =
“finished-by” fi	<	> oi mi di si	o d s	di	o	oi di si	m	sioi di	o	di	f fi =	fi

Table 3. Set of Disjunctive Relations

Disjunctive Relations(28)
after, metby, meets, during, starts, equals, before, contains, finishes, overlaps, startedby, finishedby, overlappedby, before |meets| overlaps, after| metby| overlappedby, during| overlaps| starts, contains| overlappedby| startedby, equals| startedby| starts, equals| finishes| finishedby, contains| finishedby| overlaps, during| finishes| overlappedby, before| contains| finishedby| meets| overlaps, after| during| finishes| metby| overlappedby, before| during| meets| overlaps| starts, after| contains| metby| overlappedby| startedby, contains| during| equals| finishes| finishedby|overlappedby| overlaps| startedby| starts, after| contains| during| equals| finishes| finishedby| metby| overlappedby| overlaps| startedby| starts, before| contains| during| equals| finishes| finishedby| meets| overlappedby| overlaps| startedby| starts

실이 함께 참일 때, Table 2의 조합 표를 통해서 이행 관계 추론 규칙을 적용하면 두 이벤트 a와 c사이의 시간적 순서 관계는 overlaps(o), during(d), starts(s) 중 하나의 관계를 만족할 수 있다는 것을 알 수 있다. 따라서 새로 유도되는 사실은 "a overlaps(o) ∣ during (d) ∣ starts(s) c"와 같이 다수의 서술자들을 포함한 문장으로 표현되며, 이러한 시간적 순서 관계들을 본 논문에서는 이접 관계라(disjunctive relations) 부른다. 따라서 조합 표로 표현한 Allen의 이행 관계 추론 규칙들은 정의 관계들을 유도하는 결정적 규칙(deterministic rule)들뿐만 아니라, 이접 관계들을 유도하는 비-결정적 규칙(non-deterministic rule)들도 다수 포함하고 있다. 이행 관계 추론이 반복되면 이러한 비-결정적 추론 규칙들로 인해 다수의 이접 관계 사실들이 유도되고, 이들이 다시 또 다른 이접 관계 사실들을 확대 생산하는데 기여를 하는 형태로 추론 과정이 복잡해진다. 따라서 실제로 이행 관계 추론이 수행될 때는 정의 관계들의 가능한 조합들만을 나타내는 Table 2와 같은 정의 관계 조합 표보다는 이접 관계들의 가능한 모든 조합들을 나타내는 이접 관계 조합 표가 있어야 추론을 효율적으로 구현할 수 있다.

13개의 시간 순서관계 서술자들로부터 만들 수 있는 가능한 모든 이접 관계들은 이 서술자 집합의 가능한 모든 부분집합(subset)들을 구하는 것과 동일하므로, 총 2^{13}개의 서로 다른 이접 관계들이 생성될 수 있다. 따라서 등장 가능한 모든 이접 관계들의 조합 표를 구성하면, 이 조합 표는 $(2^{13}\times 2^{13})$의 크기를 갖기 때문에, 이 조합 표를 저장하고 추론에 참조하기 위해서는 매우 큰 기억 공간과 많은 계산 시간을 요구하게 된다. 이러한 문제점을 극복하기 위해 본 연구에서는 시뮬레이션 추론과정을 거쳐 이행 관계 추론 과정 중에 실제 등장 가능한 이접 관계들의 집합을 구해보았다. 그 결과 2^{13} 개의 이접 관계들 중 Table 3에 열거한 총 28 개의 이접 관계들만 추론과정에 발생하고, 나머지 많은 이접 관계들은 실제로는 생성되지 않는다는 사실을 발견하였다. 따라서 본 논문에서는 이행 관계 추론 규칙을 $(2^{13}\times 2^{13})$ 크기의 낭비적인 이접 관계 조합 표 대신 Table 3에 열거한 28 개의 이접 관계들의 조합만을 표현한 (28×28) 크기의 이접 관계 조합 표로 축소, 정의하였다. 이와 같은 최적화된 이접 관계 집합의 발견과 조합 표의 축소는 정성 시간 추론 전반에 걸쳐 성능 향상에 큰 도움을 준다.

마지막으로 Table 1의 정제 추론 규칙(refining rule)은 추론 과정을 통해 두 이벤트 x와 y의 시간 순서 관계에 관한 서로 다른 여러 이접 관계 사실들이 유도되면, 이들 간에 모순(contradiction)이 없는 지 검사하고, 모순이 없다면 이들을 좀 더 간결한 하나의 사실로 통합, 정제해주는 역할을 수행한다. 예컨대, 기존의 지식 베이스에 두 이벤트 a와 b의 시간 순서 관계를 나타내는 "a before(<) b"라는 사실이 이미 있고, 추론을 통해 "a before(<) ∣ overlaps(o) ∣ meets(m) b"라는 새로운 사실이 유도되었다면, 이 두 사실들은 before를 공통 관계로 포함하고 있기 때문에 서로 모순을 나타내지 않으며, 나아가 이 두 사실은 "a before b"라는 보다 간결한 하나의 사실로 통합, 정제될 수 있다.

4. 병렬 시간 추론기의 설계

4.1 추론 작업 구성과 흐름

본 논문에서는 3.2 절에서 정의한 시간 지식 표현과 추론 규칙들을 기초로, 웹 스케일의 대규모 시간 지식 베이스를 추론할 수 있는 Hadoop MapReduce[13] 프레임워크 기반의 병렬 정성 시간 추론 알고리즘을 설계하였다. MapReduce 프레임워크는 Hadoop 클러스터 시스템에서 분산 병렬 응용 프로그램을 개발할 수 있는 대표적인 프로그래밍 환경이다. 다수의 컴퓨터 노드들에서 병렬로 수행되는 하나의 MapReduce 작업(job)은 한 쌍의 Map 함수와 Reduce 함수로 구성된다. MapReduce 프레임워크를 이용한 효율적인 병렬 정성 시간 추론기를 개발하기 위해서는, 먼저 앞서 정의한 정성 시간 추론 규칙들을 몇 개의 MapReduce 작업들로 묶고, 작업들 간의 의존성을 고려하여 작업 순서를 결정하는 일이 중요하다. 그런 다음, 다시 각 추론 작업별로 불필요한 자원 소모를 최소화하고 연산의 병렬성을 극대화할 수 있도록 Map 함수와 Reduce 함수를 설계하는 일이 필요하다. 본 논문에서 제시하는 MapReduce 프레임워크 기반의 병렬 정성 시간 추론기는 Fig. 3과 같은 작업 구성과 흐름으로 설계하였다.

Fig. 3에서 보듯이 본 연구의 병렬 정성 시간 추론기에서 가장 먼저 수행되는 MapReduce 작업은 사전 인코딩(Dictionary

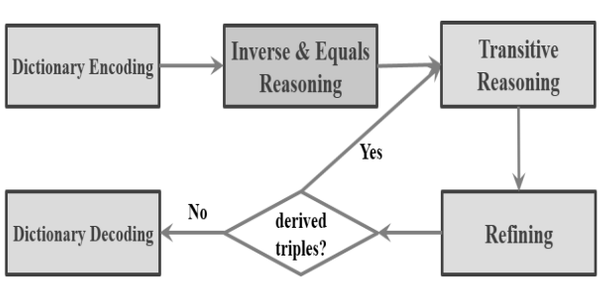

Fig. 3. Job Flow of the Temporal Reasoner

Encoding) 작업이다. 사전 인코딩 작업은 지식 베이스에 포함된 문자열 형태의 각 트리플 문장(triple statement)들을 숫자 형태의 주어 식별자(subject ID), 술어 식별자(predicate ID), 목적어 식별자(object ID)들의 조합으로 변환함으로써, 추론 작업 이전에 미리 지식 베이스의 크기를 효과적으로 축소하는 작업이다. 이러한 사전 인코딩 작업은 추론 작업 전반의 효율성을 높이는데 큰 도움을 준다. 인코딩 작업이 끝난 지식 베이스에 대해, 다음은 역 관계 및 동일 관계 추론(Inverse & Equal Reasoning) 작업이 수행된다. 이 작업에서는 인코딩된 지식 베이스에 대해 역 관계와 동일 관계를 동시에 추론한다. 이어서 더 이상 새로운 사실들이 유도되지 않을 때까지, 이행 관계 추론(Transitive Reasoning) 작업과 관계 정제(Refining) 작업이 차례대로 반복 수행된다. 이행 관계 추론 작업에서는 앞서 정의한 이행 관계 추론 규칙들을 적용하여 새로운 사실들을 유도하며, 관계 정제 작업에서는 추론된 사실들 간의 모순을 체크하고 이접 관계 사실들을 간결하게 정제하는 일을 수행한다. 끝으로 사전 디코딩(Dictionary Decoding) 작업은 추론이 완료된 최종 결과 지식 베이스를 다시 원래 모습인 텍스트 형태로 변환하는 일을 수행한다. 이어지는 후속 절들에서는 5 가지 MapReduce 작업 각각의 상세 설계에 대해 설명한다.

4.2 인코딩과 디코딩 작업

앞서 설명한대로 인코딩(Dictionary Encoding) 작업은 텍스트 형태의 각 트리플 문장(triple statement)들을 숫자 형태의 주어 식별자(subject ID), 술어 식별자(predicate ID), 목적어 식별자(object ID)들의 조합으로 변환함으로써, 추론 작업 이전에 미리 지식 베이스의 크기를 효과적으로 축소하는 작업이다.

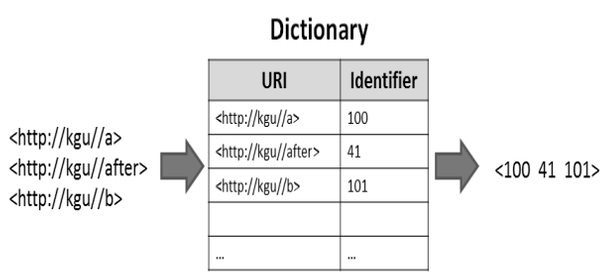

Fig. 4. Example of Encoding

Fig. 4는 인코딩 작업의 한 예를 보여주고 있다. 이 작업에서는 "<http://kgu//a> <http://kgu//after> <http://kgu//b>"와 같이 URI들을 포함하는 긴 문자열 형태의 트리플 문장들이 Fig. 4의 가운데 위치한 사전(dictionary)을 이용해 "100 41 101"와 같이 숫자로 된 식별자(ID)들의 조합으로 변환된다. 사전에는 원래 주어, 술어, 목적어의 URI들과 그들에 대응되는 각각 식별자들이 함께 저장된다. 이와 같은 인코딩 작업은 Fig. 5와 같이 다시 2 개의 순차적인 세부 MapReduce 작업들로 구현된다. 첫 번째 작업에서는 지식 베이스를 구성하는 각 트리플 문장을 주어, 술어, 목적어 등 구성 요소별로 분해하고, 각 구성 요소의 URI에 고유한 식별자(ID)를 부여하여 사전에 등록한다. 그와 동시에 각 구성 요소의 URI를 사전에 등록된 고유 식별자로 변환한다. 두 번째 작업에서는 변환된 구성 요소들의 식별자들을 결합하여, 식별자들로만 구성된 간결한 트리플 문장을 재구성한다.

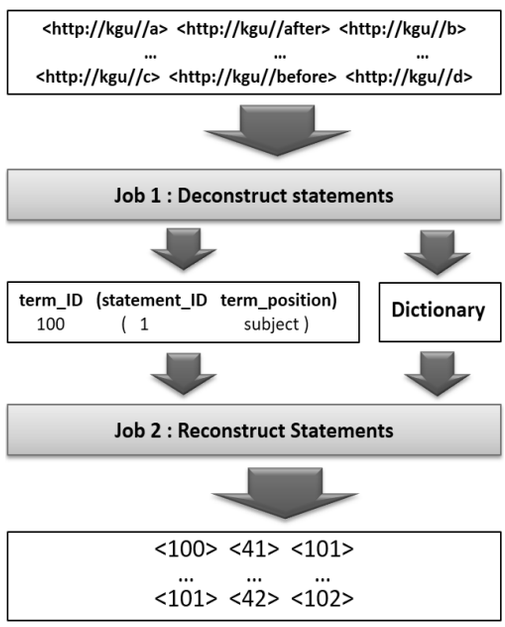

Fig. 5. Encoding Process

시간 추론 작업들이 모두 완료된 후에는 최종 결과 지식 베이스를 다시 본래의 모습인 문자열 형태로 재 변환하는 디코딩(Dictionary Decoding) 작업이 수행된다. 디코딩 작업에서는 인코딩 작업 때 만들어 둔 사전(dictionary)을 이용해, 각 문장에 포함된 식별자(ID)들을 다시 원래의 URL로 변환하는 일을 수행한다.

4.3 역 관계 및 동일 관계 추론 작업

인코딩 작업이 완료된 후에는, 인코딩된 지식 베이스에 대해 역 관계 및 동일 관계 추론 작업을 가장 먼저 수행한다. 이 작업에서는, 예컨대, "a before b"라는 두 이벤트 a와 b의 순서 관계를 나타내는 하나의 사실에 대해, 역 관계를 나타내는 "b after a"와 동일 관계를 나타내는 "a equal a", "b equal b"들을 각각 유도한다.

Fig. 6은 하나의 MapReduce 작업으로 구현되는 역 관계 및 동일 관계 추론의 한 예를 보여준다. 예컨대, 하나의 트리플 문장인 "a after b"에 대해, 맵 단계에서는 역 관계를 나타내는 "b before a"를 키(key)로, 널(null)을 값(value)로 갖는 (키, 값) 쌍을 생성한다. 이어서 리듀스 단계에서는 하나의 키 "b before a"에 대해, 동일 관계들인 "a equal a"와 "b equal b"들을 추가적으로 생성하여 그 결과들을 출력한다. 역 관계 및 동일 관계 추론 작업의 알고리즘은 Fig. 7의 의사코드(pseudo code)와 같다.

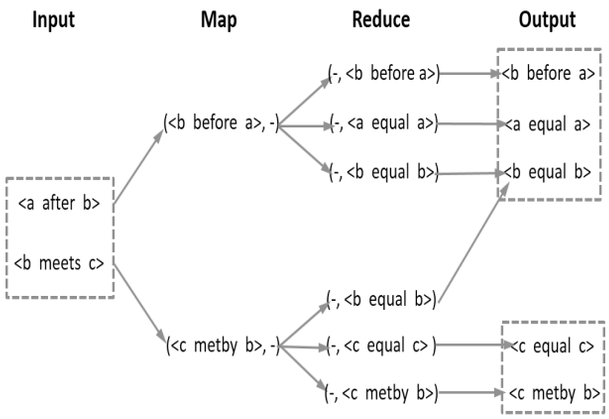

Fig. 6. Example of Inverse & Equal Reasoning

```
map (key, value) :
    // key : irrelevant, value : triple
    if value_sub != value_obj then
        inversePred ← inverse( value_pred)
        write(triple(value_obj, inversePred, value_sub), null)
    end if
    else
        if value_pred != EQ then   exit // Inconsistency
        end if
reduce (key, iterator values) :
        write(null, triple(key_subj, EQ, key_subj))
        write(null, triple(key_obj, EQ, key_obj))
        write(null, triple(key_subj, key_pred, key_obj))
```

Fig. 7. Inverse & Equal Reasoning Algorithm

Fig. 7의 맵 함수에서는 먼저 트리플 문장의 주어와 목적어의 일치성을 검사한다. 만약 주어와 목적어가 일치하지 않으면, 곧바로 역 관계 문장을 키로, 널을 값으로 갖는 (키, 값) 쌍을 생성한다. 하지만 만약 주어와 목적어가 일치하는 경우라면, 술어가 equal(EQ)인지 추가 검사하여 equal(EQ)이 아니라면 이 경우는 지식 베이스의 일관성 오류로 판정한다. Fig. 7의 리듀스 함수에서는 키로 주어진 문장으로부터 주어와 목적어를 추출한 다음, equal(EQ) 서술자를 삽입하여 주어와 주어, 목적어와 목적어 간의 동일 관계를 나타내는 트리플 문장들을 생성한다.

4.4 이행 관계 추론 작업

이행 관계 추론 작업(Transitive Reasoning)은 앞서 정의한 이접 관계 조합 표를 이용하여, 트리플 문장들로 표현된 2 개의 사실들부터 새로운 이접 관계 사실을 유도해내는 작업이다. 예컨대, “a contains(di) b” 와 “b metby(mi) c”라는 2 개의 기존 사실들로부터 “a overlappedby(oi) | contains(di) | startedby(si) c”라는 새로운 이접 관계(disjunctive relation) 사실을 유도한다.

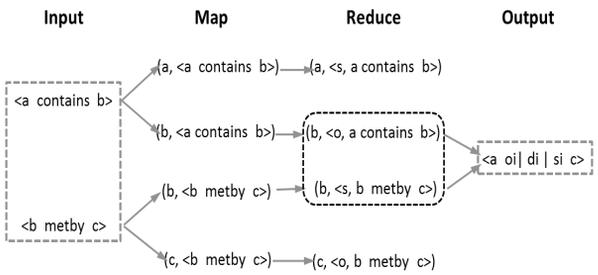

Fig. 8. Example of Transitive Reasoning

Fig. 8은 하나의 MapReduce 작업으로 구현되는 이행 관계 추론의 한 예를 보여준다. 그림과 같이 입력으로 “a contains(di) b”, “b metby(mi) c”와 같은 사실들이 주어지면, 맵 단계에서는 각 문장에서 주어와 목적어를 추출하여, 주어와 목적어를 키(key)로, 원래 문장에서 키의 역할을 나타내는 플래그(flag)와 원래 문장을 결합하여 값(value)으로 하는 (키, 값) 쌍들을 만든다. 키의 역할이 원래 문장에서 주어(subject)이면, 플래그는 s로 표기하고, 목적어(object)이면, 플래그는 o로 표기한다. 즉, “a contains b”로부터는 (a, <s, a contains b>), (b, <o, a contains b>)와 같은 2 개의 (키, 값) 쌍들이 만들어진다.

이어서 리듀스 단계에서는 키가 동일한 (b, <o, a contains b>)와 (b, <s, b metby c>)에 대해, 역할 플래그가 목적어(o)를 나타내면 이접 관계 조합 표(composition table)의 contains(di) 행(row)과 대응시키고, 플래그가 주어(s)를 나타내면 조합 표의 metby(mi) 열(column)과 대응시켜 서로 교차하는 난(entry)에서 지정하는 방식대로 새로운 이접 관계 사실인 “a | overlappedby(oi) | contains(di) | startedby(si) | c”를 유도해낸다. 이행 관계 추론 작업의 알고리즘은 Fig. 9의 의사코드(pseudo code)와 같다.

Fig. 9의 맵 함수에서는 한 트리플 문장의 주어($value_{subj}$)와 목적어($value_{obj}$)를 각각 키(key)로, 원래 문장 앞에 해당 키의 역할 플래그 s 또는 o를 각각 붙인 것을 값(value)으로 삼아, 2 개의 (키, 값) 쌍인 ($value_{subj}$, ‘s’+value)와 ($value_{obj}$, ‘o’+value)을 만든다. Fig.9의 리듀스 함수에서는 먼저 맵 함수에서 붙여준 역할 플래그를 검사하여, 플래그가 o이면 이접 관계 조합 표의 행에 대응될 트리플들의 집합(rows)으로, 플래그가 s이면 조합 표의 열에 대응될 트리플들의 집합(columns)으로 분류한다. 그런 다음, 조합 표의 행 $triple_row_{pred}$과 열 $triple_col_{pred}$의 조합을 통해, 새로운 이접 관계 derivedPred를 유도해낸다. 새로 유도된 이접 관계 derivedPred와 이것의 역 관계 inversePred를 각각 술어로 갖는 2 개의 새로운 트리플 문장들을 만들어 낸다.

본 논문에서 설계한 이행 관계 추론 작업에서 한 가지 주목할 점은, 이행 관계 추론으로 얻어지는 새로운 사실들에 대해 그들의 역 관계 사실들도 이 작업을 통해 한 번에 유도하도록 설계하였다는 점이다. 이렇게 함으로써 정성 시간 추론 과정 전반에 필요한 MapReduce 작업의 수를 줄일 수 있어 성능 향상에 도움을 줄 수 있게 된다. 또 다른 효율성

을 고려한 설계 사항은 이행 관계 추론이 가능한 사실들 중에서 “a after b”, “b before a”와 같이 앞 문장의 주어 a와 뒤 문장의 목적어 a가 서로 일치하는 경우에는 선행 연구들[4, 5]과는 달리 본 연구에서는 이행 관계 추론의 대상에서 제외시켰다. 그 이유는 이 경우에 이행 관계 추론의 결과로 얻어지는 사실은 “a equal a” 뿐인데, 이것은 이미 앞선 역관계 및 동일 관계 추론 작업을 통해 얻어진 사실들 중 하나이기 때문이다. 특히 일회성으로 수행되는 역 관계 및 동일 관계 추론 작업과는 달리 이행 관계 작업은 여러 번 반복 수행되어야 하는 작업이기 때문에 이와 같이 이행 관계 추론 작업의 불필요한 추론 연산을 최대한 줄이는 설계는 추론 전반의 성능에 큰 도움을 줄 수 있다.

```
map(key, value) :
    // key : irrelevant, value : triple
    write(value_subj, 's' + value)
    write(value_obj, 'o' + value)
reduce(key, iterator values) :
    for each value in values do
        if value[0] == 'o' then add value to rows
                    else add value to columns
    for each triple_row in rows do
        for each triple_col in columns do
            if triple_row_obj == triple_col_subj then
                derivedPred ←
                    composition(triple_row_pred, triple_col_pred)
                inversePred ← inverse(derivedPred)
                write(null, triple(triple_row_subj, derivedPred, triple_col_obj))
                write(null, triple(triple_col_obj, inversePred, triple_row_subj))
            end if
        end for
    end for
```

Fig. 9. Transitive Reasoning Algorithm

4.5 관계 정제 작업

앞서 설명한 것과 같이 관계 정제(Refining) 작업에서는 추론된 사실들 간의 불일치성을 체크하고 이접 관계 사실들을 간결하게 정제하는 일을 수행한다. 예컨대, 추론을 진행하는 동안에 “a overlappedby(oi) b”와 “a contains(di) | startedby(si) b”라는 두 사실이 유도되었다고 하자. 이 경우 동일한 두 이벤트 a와 b에 대해, 이 두 사실은 서로 동시에 만족할 수 있는 순서 관계를 전혀 포함하고 있지 않기 때문에 서로 불일치성 혹은 모순을 보인다고 판단할 수 있다. 반면에, “a overlappedby(oi) b”와 “a overlappedby(oi) | contains(di) | startedby(si) b”라는 두 사실들의 경우, 두 이벤트 a와 b에 대해 순서 관계 overlappedby(oi)를 공통으로 포함하고 있으므로 서로 모순을 갖지 않는다고 판단하며, 서로 다른 두 사실은 공통 순서 관계들만을 포함한 하나의 사실 “a overlappedby b”로 대체함으로써 지식을 보다 간결하게 정제할 수 있다.

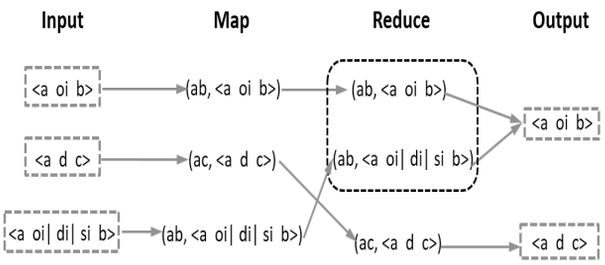

Fig. 10. Example of Refining

Fig. 10은 하나의 MapReduce 작업으로 구현되는 관계 정제 작업의 한 예를 보여준다. “a overlappedby(oi) b”, “a during(d) b”, “a overlappedby(oi) | contains(di) | startedby(si) b” 등과 같은 문장들이 입력으로 주어졌을 때, 맵 단계에서는 각 문장의 주어와 목적어를 결합하여 이것을 키(key)로, 원래 문장을 값(vlaue)으로 하는 (ab, <a oi b>), (ac, <a d c>), (ab, <a oi | di | si b>) 와 같은 (키, 값) 쌍들을 만든다. 이와 같은 과정을 통해 같은 주어와 목적어를 가진 문장들끼리 그룹화(grouping)가 이루어진다. 리듀스 단계에서는 같은 키를 가진, 즉 같은 주어와 목적어를 가진 문장들에 대해 이접 관계들의 교집합(intersection)을 구한 다음, 원래 문장들을 대신하는 보다 정제된 하나의 문장 “a overlappedby(oi) b”를 결과로 출력한다. 관계 정제 작업의 알고리즘은 Fig. 11의 의사코드와 같다.

```
map(key, value) :
    // key : irrelevant, value : triple
    write(value_subj + value_obj, value)
reduce(key, iterator values) :
    for each value in values do
        refinedTriple ← triple(value_subj, refinedTriple_pred ∩ value_pred, value_obj)
    end for
    if refinedTriple_pred == null then exit // Inconsistency
    end if
    write(null, refinedTriple)
```

Fig. 11. Refining Algorithm

Fig. 11의 맵 함수에서는 각 트리플 문장의 주어와 목적어를 결합한 키($value_{subj}$ + $value_{obj}$), 그리고 원래 문장을 값(value)로 하는 (키, 값) 쌍을 만든다. 리듀스 함수에서는 동일한 키를 갖는, 즉 같은 주어와 목적어를 갖는 문장들의 이접 관계 교집합 $refinedTriple_{pred}$을 구해, 그것이 공집합(null)이면 지식들 간에 불일치성이 발견되었음을 통보하고 작업을 종료한다. 그렇지 않으면, 이접 관계 교집합을 이용해 정제된 하나의 문장 refinedTriple을 만들어 낸다.

5. 구현 및 실험

5.1 병렬 시간 추론기의 구현

본 논문에서는 앞서 소개한 MapReduce 기반의 정성 시간 추론 알고리즘을 기초로, 대규모 정성 시간 추론을 수행할 수 있는 병렬 시간 추론기인 MRQUTER를 구현하였다. 구현 환경은 자바 1.7 버전과 Hadoop 2.5.0 버전을 사용하였고, 실험 환경은 10개의 테스크 노드(task node)로 구성된 Hadoop 완전 분산 모드 클러스터를 사용하였다. 각 슬래이브 노드(slave node)는 3.5GHz, 4 Core CPU와 8GB 메인 메모리, 1TB 하드 디스크로 구성되어 있다.

Fig. 12는 Hadoop MapReduce[13] 프레임워크를 기반으로 구현된 병렬 시간 추론기인 MRQUTER의 작업 수행 과정을 간략히 보여주고 있다. 먼저 시간 추론 작업의 대상이 되는 대용량의 지식 베이스를 Hadoop 클러스터 시스템의 분산 파일 시스템인 HDFS(Hadoop Distributed File System)로부터 읽는다. 그 후, 스케쥴러(Schedulr)와 노드 매니저(Node Manger)를 통해서 테스크(task)를 클러스터를 구성하는 각 노드(Node)로 전송한다. 각 노드들은 작업 스케쥴에 따라 해당 테스크를 가져온 후, 역 관계 추론 작업, 이행 관계 추론 작업, 관계 정제 작업 등 추론 작업 파이프라인(pipeline)을 차례대로 병행 수행하며, 추론의 결과로 얻어지는 확장된 지식 베이스는 다시 분산 파일 시스템인 HDFS에 저장한다.

5.2 성능 실험

본 논문에서 제안하는 병렬 정성 시간 추론기인 MRQUTER의 성능을 평가하기 위한 실험을 수행하였다. 실험을 위해 Table 4에 열거된 벤치마킹 시간 지식 베이스(TG)들을 사용하였다. 각 지식 베이스에 포함된 이벤트(event) 객체들의 수에 따라 TG250K, TG1000K, TG1750K, TG2500K, TG3250K 등으로 부르며, Table 4에는 각 지식 베이스에 포함된 이벤트들의 시간 순서 관계를 나타내는 사실들(facts)의 개수와 총 파일 크기를 함께 소개하고 있다.

Table 4. Knowledge Bases

Knowledge Bases	Number of Events	Number of Facts	Size of Files (MB)
TG250K	250,000	426,701	66
TG1000K	1,000,000	1,701,701	258
TG1750K	1,750,000	2,976,701	455
TG2500K	2,500,000	4,001,601	610
TG3250K	3,250,000	5,526,701	840

첫 번째 실험에서는 병렬 시간 추론기인 MRQUTER의 추론을 통한 새로운 지식 생성 능력을 평가해보았다. 실험 결과는 Fig. 13과 같다. Fig. 13의 그래프에서 가로축은 실험에 사용된 지식 베이스들을 나타내고, 세로축은 트리플(triple) 개수로 표시한 지식의 양을 나타낸다. 그래프에서 검은 색 막대는 추론의 입력으로 사용된 초기 지식의 양을 나타내며, 빗금 친 막대는 추론을 통해 새로 유도된 지식의 양을 나타내고 있다.

그래프를 통해, 우리는 초기 지식의 양이 증가함에 따라 새로 유도되는 지식의 양도 비례해서 증가한 결과를 확인할 수 있다. 그뿐만 아니라 TG3250K 지식 베이스의 경우와 같이, 약 550만개의 트리플로 구성된 대용량의 입력 시간 지식 베이스에 대해 약 7300 만개가 넘는 매우 많은 새로운 지식들이 유도되었다는 것을 알 수 있다. 따라서 이와 같은 실험 결과를 통해, 병렬 시간 지식 추론기인 MRQUTER의

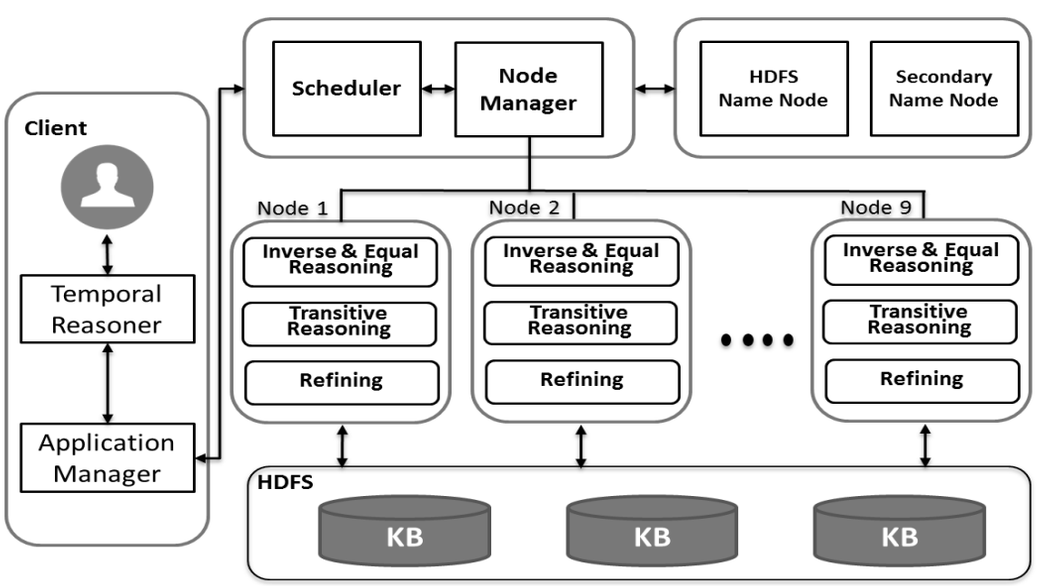

Fig. 12. Organization of MRQUTER

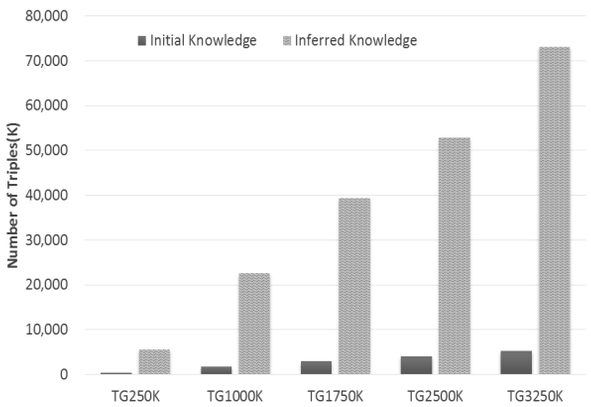

Fig. 13. Number of inferred facts

뛰어난 지식 베이스 확장 능력을 확인할 수 있었다.

두 번째 실험에서는 병렬 시간 추론기인 MRQUTER의 단위 작업 별 추론 시간을 측정해보았다. 측정 대상 작업들은 역 관계 및 동일 관계 추론 작업(Inverse & Equal), 이행 관계 추론 작업(Transitive), 관계 정제 작업(Refining) 등이다. 지식 베이스 TG1000K와 TG1750K 각각에 대해 수행한 실험 결과는 Fig. 14와 같다.

Fig. 14의 그래프에서 보듯이, MRQUTER의 추론 작업들 중에서 역 관계 및 동일 관계 추론 작업 시간은 이행 관계 추론 작업과 관계 정제 작업 시간에 비해 매우 짧고, 나머지 두 개의 추론 작업 시간이 전체 추론 작업 시간의 대부분을 차지한다는 것을 알 수 있다. 그 이유는 동일 및 역 관계 추론 작업은 다른 작업들과 달리 조인(join) 연산이 없고 알고리즘도 비교적 단순하며, 전체 추론 과정 중 단 일회만 수행되기 때문인 것으로 판단된다. 한편, 나머지 이행 관계 추론 작업과 관계 정제 작업은 대부분 조인 연산을 포함하고 있으며, 더 이상 새로운 지식이 추론되지 않을 때까지 반복 수행되어야 하는 작업들이기 때문에 많은 계산 시간을 소모한 것으로 판단한다. 따라서 이 두 개의 단위 추론 작업들의 효율성을 향상시킬 수 있다면, 역 관계 및 동일 관계 추론 작업에 비해 훨씬 더 시간 추론 시스템의 성능 개선에 큰 도움을 줄 수 있을 것으로 판단한다.

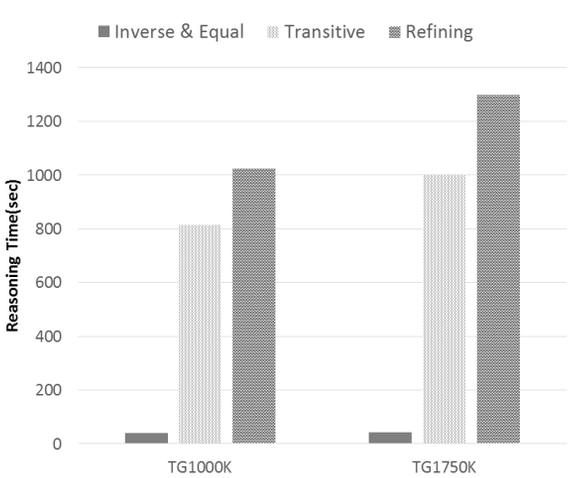

Fig. 14. Reasoning Time for Individual Reasoning Jobs

Table 5. Comparison between Before and After Optimization

Test Cases			Before Optimization	After Optimization
Knowledge Bases	Number of Facts	Number of Inferred Facts	Reasoning Time (Min)	Reasoning Time (Min)
TG250K	426,701	5,547,120	28	18
TG1000K	1,701,701	22,466,340	46	32
TG1750K	2,976,701	39,318,776	68	45
TG2500K	4,001,601	52,793,560	78	58
TG3250K	5,526,701	73,014,753	97	72

세 번째 실험에서는 본 논문에서 제안한 이접 관계 조합표 축소 등 이행 관계 추론과 관계 정제 알고리즘의 최적화 기술들이 MRQUTER 시스템의 성능에 미친 영향을 분석하기 위한 실험을 수행하였다. 이를 위해 최적화 기술들이 적용되기 이전(Before Optimization)과 적용 이후(After Optimization)의 총 추론 시간(reasoning time)을 측정하여 비교해보았다. 실험 결과는 Table 5와 같다.

Table 5에서 보듯이 최적화 이전과 최적화 이후 추론된 지식의 양에는 변화가 없었다. 예컨대, TG3250K 지식 베이스의 경우, 최적화 이후에도 약 550만 개의 입력 초기 지식으로부터 최적화 이전과 동일한 약 7300만 개의 신규 지식이 새롭게 유도되었다는 것을 알 수 있다. 하지만, 최적화 기술은 추론에 소모한 계산 시간, 즉 추론 시간 면에서는 뚜렷한 개선이 있었다는 것을 확인할 수 있다. 최적화 이전에 비해 적게는 약 10분에서 많게는 약 23분 가량의 추론 시간 감소 현상이 나타났다. 이를 통해, 본 논문에서 제안한 추론의 최적화 기술들이 MRQUTER의 추론 성능 향상에 긍정적인 효과를 주었음을 확인할 수 있었다.

끝으로 마지막 실험에서는 MRQUTER 시스템의 확장성(scalability)을 평가하기 위한 실험을 수행하였다. 이를 위해, 입력 지식 베이스의 크기와 Hadoop 클러스터 시스템의 노드(node) 수를 확장하면서 MRQUTER의 추론 시간 변화를 분석해 보았다. 실험결과는 Fig. 15와 같다.

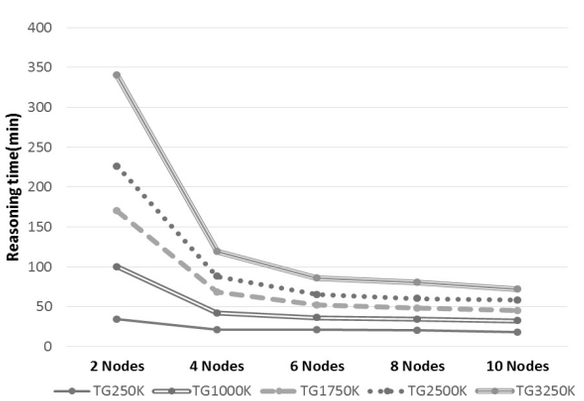

Fig. 15. Node Scalability

Fig. 15의 그래프에서 가로축은 클러스터 시스템을 구성하는 노드들의 수를 나타내고, 세로축은 소모된 추론 시간을 나타내며, 서로 다른 유형의 그래프 선들은 크기가 다른 입력 지식 베이스들을 각각 나타낸다. 전체적으로 클러스터 시스템의 노드 수가 증가할수록 MRQUTER의 총 추론 시간은 뚜렷한 감소 추세를 보인 것을 알 수 있다. 또한, 이러한 추론 시간 감소 현상은 입력 지식 베이스의 규모가 커질수록 더욱 뚜렷하게 나타나는 것도 확인 할 수 있다. 예컨대, 소규모 지식 베이스인 TG250K에 비해 대규모 지식베이스인 TG3250K의 경우 더 큰 폭으로 추론 시간이 감소하였다. 따라서 이러한 실험 결과를 토대로, 클러스터 시스템의 노드 수 확장만으로도 MRQUTER의 성능을 어느 정도 지속적으로 향상시킬 수 있을 것으로 기대할 수 있다.

6. 결론 및 향후 연구

본 논문에서는 그동안 소규모 지식 베이스를 이용한 실험실 수준의 정성 시간 추론 연구들에서 벗어나, 웹 스케일의 대규모 지식 베이스를 추론할 수 있는 병렬 정성 시간 추론기인 MRQUTER의 설계와 구현을 제시하였다. MRQUTER는 Hadoop 클러스터 시스템과 MapReduce 병렬 프로그래밍 프레임워크를 이용해 개발되었으며, 전체 정성 시간 추론 과정을 추론 규칙들 간의 논리적 의존성에 따라 인코딩 및 디코딩 작업, 역 관계 및 동일 관계 추론 작업, 이행 관계 추론 작업, 관계 정제 작업 등 몇 개의 MapReduce 작업으로 나누고, 맵 함수와 리듀스 함수로 구현되는 각각의 단위 추론 작업을 효율화하기 위한 최적화 기술들을 적용하였다. 대규모 테스트베드 시간 지식 베이스를 이용한 실험을 통해, MRQUTER의 높은 추론 성능과 확장성을 확인하였다.

역 관계 및 동일 관계 추론 작업, 이행 관계 추론 작업, 관계 정제 작업 등 단위 추론 작업들의 순차(sequence) 및 반복(iteration)으로 수행되는 정성 시간 추론의 전체 과정은 작업들 간의 데이터 전이를 위한 Hadoop 분산 파일 시스템(HDFS) 상의 많은 파일 입출력을 요구한다. 이것은 병렬 정성 시간 추론기의 성능 향상에 큰 걸림돌로 작용한다. 따라서 작업 간 인-메모리 데이터 전이가 가능한 새로운 병렬 프로그래밍 프레임워크인 Apache Spark[16]을 이용해 병렬 정성 시간 추론기를 구현한다면 더 개선된 성능을 얻을 수 잇을 것으로 기대한다. 향후 연구에서는 현재 MRQUTER 시스템의 추가적인 성능 최적화뿐만 아니라, Apache Spark의 인-메모리 기능을 이용한 병렬 시간 추론 알고리즘의 설계도 진행해볼 계획이다.

References

[1] W3C Recommendation, "OWL Web Ontology Language Semantics and Abstract Syntax," http://www.w3org/TR/owl-ref/, 2004.

[2] C. Gutierrez, C. Hurtado, and A. Vaisman, "Temporal RDF," *Proceedings of European Semantic Web Conference*, 2005.

[3] V. Milea, F. Frasincar, and U. Kaymak, "tOWL: A Temporal Web Ontology Language," *IEEE Transactions on Systems, Man, Cybernetics*, Vol.42, No.1, pp.268-281, 2011.

[4] K. Kyzirakos, "The Data Model stRDF and the Query language stSPARQL," *Proceedings of OGC/W3C Spatial Data on the Web WG*, Barcelona, March 11, 2015.

[5] A. Salguero, C. Delgado, and F. Araque, "STOWL: An OWL Extension for Facilitating the Definition of Taxonomies in Spatio-temporal Ontologies," *Lecture Notes in Computer Science*, Vol.5736, pp.336-345, 2009.

[6] F. Grandi, "T-SPARQL: A TSQL2-like Teporal Query Language for RDF," *Proceedings of the International Workshop on Querying Graph Structured Data*, pp.21-30, 2010.

[7] M. Vilain, H. Kautz, and P. Van Beek, "Constraint Propagation Algorithm for Temporal Reasoning," *Proceedings of the 5th. National Conference on Artificial Intelligence,* 1986.

[8] J. F. Allen, "Maintaining Knowledge about Temporal Intervals," *Communications of the ACM*, Vol.26, pp.832-843, 1983.

[9] Z. Gantner, M. Westphal, and S. Wolfl, "GQR-A Fast Reasoner for Binary Qualitative Constraint Calculi," *Proceedings of AAAI*. Vol.8, 2008.

[10] S. Batsakis, and E. G. M. Petrakis, "SOWL: A Framework for Handling Spatio-Temporal Information in Owl 2.0," *Proceedings of International Symposium on RuleMl*, Vol. 6826, pp.242-249, 2011.

[11] E. Anagnostopoulos, E. G. M. Petrakis, and S. Bastsakis "CHRONOS: Improving the Performance of Qualitative Temporal Reasoning in OWL," *Proceedings of IEEE International Conference on Tools with Artificial intelligence*, pp.309-345, 2014.

[12] S. Batsakis, K. Stravoskoufos, and E.G.M. Petrakis, "Temporal Reasoning for Supporting Temporal Queries in OWL 2.0," *Proceedings of International Conference on KES*, pp.558-567, 2011.

[13] T. Gunarathne, "*Hadoop MapReduce v2 Cookbook*," Packt Publishing, 2015.

[14] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, "SWRL: A Semantic Web Rule Language Combining OWL and RuleML," W3C Member submission, 2004.

[15] M. Stocker and E. Sirin, "PelletSpatial: A Hybrid RCC-8 and RDF/OWL Reasoning and Query Engine," *OWLED*, Vol. 529, 2009.

[16] H. Karau, A. Konwinski, P. Wendell, and M. Zaharia, "*Learning Spark: Lightning-Fast Big Data Analysis*," O'Reilly Media, 2015.

김 종 훈

e-mail : skybrownss@naver.com
2015년 경기대학교 컴퓨터과학과(학사)
2015년~현 재 경기대학교 컴퓨터과학과 석사과정
관심분야 : 인공지능, 시공간 추론, 지능로봇

김 인 철

e-mail : kic@kyonggi.ac.kr
1985년 서울대학교 수학과(학사)
1987년 서울대학교 전산과학과(이학석사)
1995년 서울대학교 전산과학과(이학박사)
1996년~현 재 경기대학교 컴퓨터과학과 교수
관심분야 : 인공지능, 지식표현 및 추론, 기계학습, 지능로봇