KIPS Tr. Software and Data Eng.
Vol.6, No.2 pp.103~116 pISSN: 2287-5905

https://doi.org/10.3745/KTSDE.2017.6.2.103

SSQUSAR: A Large-Scale Qualitative Spatial Reasoner Using Apache Spark SQL

Jonghoon Kim† · Incheol Kim††

ABSTRACT

In this paper, we present the design and implementation of a large-scale qualitative spatial reasoner, which can derive new qualitative spatial knowledge representing both topological and directional relationships between two arbitrary spatial objects in efficient way using Aparch Spark SQL. Apache Spark SQL is well known as a distributed parallel programming environment which provides both efficient join operations and query processing functions over a variety of data in Hadoop cluster computer systems. In our spatial reasoner, the overall reasoning process is divided into 6 jobs such as knowledge encoding, inverse reasoning, equal reasoning, transitive reasoning, relation refining, knowledge decoding, and then the execution order over the reasoning jobs is determined in consideration of both logical causal relationships and computational efficiency. The knowledge encoding job reduces the size of knowledge base to reason over by transforming the input knowledge of XML/RDF form into one of more precise form. Repeat of the transitive reasoning job and the relation refining job usually consumes most of computational time and storage for the overall reasoning process. In order to improve the jobs, our reasoner finds out the minimal disjunctive relations for qualitative spatial reasoning, and then, based upon them, it not only reduces the composition table to be used for the transitive reasoning job, but also optimizes the relation refining job. Through experiments using a large-scale benchmarking spatial knowledge base, the proposed reasoner showed high performance and scalability.

Keywords : Qualitative Spatial Reasoning, Spark SQL, Topological Relation, Minimal Disjunctive Relations, Distributed Parallel Programming

SSQUSAR: Apache Spark SQL을 이용한 대용량 정성 공간 추론기

김 종 훈† · 김 인 철††

요 약

본 논문에서는 Apache Spark SQL을 이용하여 임의의 두 공간 객체들 간의 위상 관계와 방향 관계를 나타내는 새로운 정성 공간 지식을 효율적으로 추론해내는 대용량 정성 공간 추론기의 설계와 구현에 대해 소개한다. Apache Spark SQL은 Hadoop 클러스터 컴퓨터 시스템에서 다양한 데이터들 간의 매우 효율적인 조인 연산과 질의 처리 기능을 제공하는 분산 병렬 프로그래밍 환경이다. 본 공간 추론기에서는 정성 공간 추론의 전체 과정을 지식 인코딩, 역 관계 추론, 동일 관계 추론, 이행 관계 추론, 관계 정제, 지식 디코딩 등 크게 총 6개의 작업들로 나누고, 논리적 인과관계와 계산 효율성을 고려하여 작업들 간의 처리 순서를 결정하였다. 지식 인코딩 작업에서는 추론의 전처리 과정으로서 XML/RDF 형태의 입력 지식을 보다 간략한 내부 형태로 변환함으로써, 추론 대상인 지식 베이스의 크기를 축소시켰다. 일반적으로 이행 관계 추론 작업과 관계 정제 작업의 반복은 정성 공간 추론에 필요한 가장 많은 계산 시간과 기억 공간을 소모한다. 이 작업들을 효율화하기 위해 본 공간 추론기에서는 공간 추론에 필요한 최소한의 이접 관계들을 찾아내고, 이들을 기반으로 이행 관계 추론을 위한 조합표를 큰 폭으로 축소하고 관계 정제 작업도 최적화하였다. 대규모 벤치마킹 공간 지식 베이스를 이용한 실험을 통해, 본 논문에서 제안하는 대용량 정성 공간 추론기의 높은 추론 성능과 확장성을 확인하였다.

키워드 : 정성 공간 추론, Spark SQL, 위상 관계, 최소 이접 관계들, 분산 병렬 프로그래밍

1. 서 론

최근 들어 인터넷을 이용하는 사용자 수와 웹 기술이 발전함에 따라, 링크드 오픈 데이터(linked open data)가 급격히 증가하고 있다. 링크드 오픈 데이터 중에서 좌표나 공간 객체들 간의 공간 관계를 나타내는 공간 정보 데이터들은 첨

※ 본 연구는 미래창조과학부 및 정보통신기술연구진흥센터의 정보통신·방송 연구개발사업의 일환으로 수행하였음(10044494, WiseKB: 빅데이터 이해 기반 자가학습형 지식베이스 및 추론 기술 개발).
† 준 회 원 : 경기대학교 컴퓨터과학과 석사과정
†† 종신회원 : 경기대학교 컴퓨터과학과 교수
Manuscript Received : August 8, 2016
First Revision : September 22, 2016
Accepted : October 4, 2016
* Corresponding Author : Incheol Kim(kic@kyonggi.ac.kr)

단 정보 기술들과 융합되어 다양한 응용 분야에 이용 가능하다. 이를 효과적으로 이용하기 위한 공간 추론 기술과 공간 질의 처리 기술에 대한 연구들도 활발히 진행되고 있다.

공간 추론의 대상이 되는 공간 지식을 표현하는 방법으로는 크게 정량 공간 지식 표현 방법과 정성 공간 지식 표현 방법이 있다. 정량 공간 지식 표현 방법은 일반적으로 개별 공간 객체의 모양이나 위치 정보, 그리고 공간 객체들 간의 거리와 방향 관계 등을 수치적으로 상세히 표현하는 방식이다. 정량 공간 지식이 이용되는 대표적인 사례들로는 자동차 내비게이션 지도, 토목 혹은 건축용 지도, 지리 정보 검색 시스템 등이 될 수 있다. 반면에, 정성 공간 지식 표현 방법은 개별 공간 객체의 상세한 모양 정보나 공간 객체들 간의 수치화 된 거리와 방향 대신, 개별 공간 객체의 모양이나 위치 정보, 그리고 그들 간의 관계들을 추상화(Abstract)하여 개념적 기호(Conceptual Symbol)들로 표현하는 방식을 말한다. 정량 공간 지식 표현 방법은 기계나 정보시스템에 의해 현실 세계에서 구체적인 공간 관련 작업을 수행하기에는 반드시 필요하지만, 공간 객체들 간의 관계를 보다 신속하고 직관적으로 파악할 필요가 있는 의사결정 지원시스템이나 질의 응답 시스템의 경우에는 정성 공간 지식 표현이 더욱 중요하다. 그 뿐만 아니라 정성 공간 지식 표현 방법은 사람이 이해하기 쉽고 분석하기 편하다. 정성 공간 지식 표현법은 자연 언어 이해(Natural Language Understanding), 의사결정 지원 시스템(Decision Support System), 질의 응답시스템(Question Answering System), 지능형 서비스 로봇(Intelligent Service Robot) 등과 같은 다양한 분야에서 폭넓게 이용되고 있다[18].

대표적인 정성 공간 지식 표현법으로는 공간 객체들 간의 위상 관계(topology)를 표현하기 위한 RCC(Region Connection Calculus)-8[1], 방향(direction)을 표현하기 위한 CSD(Cone Shaped Direction Relations)-9[2], 거리를 표현하기 위한 TPCC(Ternary Point Configuration Calculus)[3], 모양을 표현하기 위한 TLT(Tripartite Line Tracks)[4] 이론 등이 있다. 이러한 정성 공간 지식 표현법을 기반으로 그동안 개발되어온 대표적인 정성 공간 추론기들로는 GQR[5], SOWL[6], PelletSpatial[7], CHOROS[8], QUSAR[9] 등이 존재한다. 그러나 이 추론기들은 모두 단일 머신 컴퓨팅 환경에서 동작하기 때문에, 대용량의 공간 지식 베이스를 추론하기에는 하드웨어 성능 면에서 한계가 존재한다.

이러한 한계점을 극복하고자, 최근 대용량 데이터를 분산 처리할 수 있는 오픈 프레임워크를 이용해 공간 추론의 성능을 향상시키고자 하는 연구들이 활발히 시도되고 있다. 그 중에서도 Hadoop 클러스터 컴퓨터 시스템에서 동작하는 분산 병렬 처리 프레임워크인 MapReduce[11]를 이용한 추론 방식이 있다. 대표적인 MapReduce 기반 정성 공간 추론기로는 MRQUSAR[12]가 존재한다. 하지만, 정성 공간 추론 과정은 단위 작업들 간의 순차성과 반복성이 높은 특성을 가지고 있는데 반해, MapReduce 프레임워크는 연속된 작업과 작업 사이에 잦은 파일 입출력을 요구하기 때문에 공간 추론 성능 향상에 한계성을 나타냈다. 한편, Hadoop 클러스터 컴퓨터 시스템에서 수행 가능한 새로운 분산 병렬 처리 프레임워크인 Apache Spark[13]는 작업들 간의 데이터 전달을 위해 RDD(Resilient Distributed Dataset) 데이터를 메모리에 캐싱함으로써 파일 입출력을 최대한 줄일 수 있어 순차 및 반복 작업에 보다 효과적이다. 최근에 개발된 Apache Spark 기반 정성 공간 추론기로는 SPQUSAR[15]가 있다. 하지만 Apache Spark 기반의 정성 공간 추론기인 SPQUSAR의 경우, 메모리 캐싱 기능을 이용한 추론 성능의 향상은 있으나, Apache Spark의 특성상 개발자가 직접 세밀하게 정의해주어야 하는 각 단위 추론 작업의 세부 지식 처리 절차의 최적화가 충분치 못한 한계점이 있다. 한편, Apach Spark 상의 한 DB 응용 개발 도구로 새롭게 소개된 Spark SQL[14]은 Spark 프로그램 개발자가 SQL 질의문을 이용해 데이터 프레임(Data Frame)이라 불리는 구조화된 데이터 집합들에 대한 효과적인 접근과 처리를 가능하게 해준다. 특히 Spark SQL은 조인 연산(join operation) 등 대용량 데이터 연산을 위해 다양한 최적화 기법을 적용하고 있으며, 개발자는 이러한 최적화된 데이터 연산 기능들을 고수준의 비-절차 질의 언어인 SQL문을 통해 이용 가능하다는 잇점이 있다.

본 논문에서는 이러한 특성을 가진 Apache Spark SQL를 이용하여 개발한 대용량 정성 공간 추론기인 SSQUSAR의 설계와 구현에 대해 소개한다. 먼저 2장에서는 본 연구와 연관 있는 기존의 정성 공간 추론 연구들과 분산 추론 환경들에 대해 살펴보고, 3장에서는 본 논문의 SSQUSAR 추론기에서 채택하고 있는 정성 공간 지식 표현법과 추론 규칙들, 그리고 추론을 위한 최적화 기법들을 소개한다. 이어서 4장에서는 Apache Spark SQL 프레임워크를 이용한 정성 공간 추론 작업들의 세부 설계에 대해 설명하고, 5장에서는 정성 공간 추론기 SSQUSAR의 구현과 성능 실험 결과들에 대해 소개한다. 끝으로 6장에서는 결론과 향후 연구를 소개한다.

2. 관련 연구

2.1 정성 공간 지식 표현법

RCC(Region Connection Calculi)-8[1] 이론에서는 2차원 공간 위의 임의의 두 지역(region)간의 위상 관계를 DC (disconnect), EC(externally connected), PO(partially overlapping), EQ(equal), TPP(tangential proper part), TPPi(tangential proper part inverse), NTTPP(non-tangential proper part), NTPPi(non-tangential proper part inverse)등 총 8개 위상 관계 중 하나로 표현할 수 있다. 한편, CSD(Cone-Shaped Directional)-9[2] 이론에서는 2차원 공간 위의 임의의 두 지점 (point) 간의 방향 관계를 한 지점을 중심으로 판단할 때 다른 한 지점의 방향을 동(E), 서(W), 남(S), 북(N), 북동(NE), 북서(NW), 남동(SE), 남서(SW), 그리고 동향(Identical) 등 총 9개 방향 관계 중 하나로 표현할 수 있다.

TPCC(Ternary Point Configuration Calculus)[3] 이론에서는 2차원 공간 위의 임의의 두 지점(point) 간의 방향과

거리의 관계를 한 지점을 중심으로 판단할 때 다른 한 지점의 방향과 거리를 앞(front), 뒤(back), 왼쪽(left), 오른쪽(right), 평행(straight), 멀리 있다(distant), 가깝다(close)의 조합으로 만들수 있는 총 27가지 관계 중 하나로 표현할 수 있다. 한편, TLT(Tripartite Line Tracks)[4] 이론에서는 2차원 공간 위의 임의의 한 지역을 15등분으로 나눈 뒤 해당하는 부분을 세 개의 연결된 선으로 구성된 36개의 표본 중에서 가장 유사한 것을 선택한다.

2.2 정성 공간 추론기

대표적인 정성 공간 추론기들로는 GQR[5], SOWL[6], PelletSpatial[7], CHOROS[8], QUSAR[9] 등이 존재한다. GQR은 XML과 텍스트 파일을 통해서 기본지식 베이스를 표현한다. 또한, 입력지식에 대해서 제약 네트워크(constraint network)를 기반으로 지식들을 인접 행렬 형태로 구성하고 추론할 수 있는 정성 추론기이다. 정성 시간 지식 표현과 추론을 위해서 Allen의 이론과 Allen 시간 조합표를, 정성 공간 지식 표현과 추론을 하기 위해서 방향을 표현할 수 있는 CSD-9 이론과 CSD-9 조합표를, 위상 관계를 표현할 수 있는 RCC-8 이론과 RCC-8 조합표를 각각 적용하였다. GQR은 역추적(back tracking)을 이용한 경로 일관성 알고리즘[16]을 통해서 효율적인 정성 추론 및 일관성 검사를 할 수 있다.

SOWL은 대표적인 정성 시공간 추론기로써, 시맨틱 웹 온톨로지 언어인 RDF/OWL으로 표현된 지식 베이스를 4차원 술어(4-D fluent)와 다자 관계(N-ary relation)를 사용했다. 또한 시공간 추론을 위해서 시맨틱 웹 규칙 언어인 SWRL (A Semantic Web Rule Language Combining OWL and RuleML)을 이용하여 시 공간 추론 규칙을 정의하고 시공간 추론을 수행한다. 이 추론기에서는 정성 시간 지식 표현과 추론을 위해서 Allen의 이론과 Allen 시간 조합표를, 정성 공간 지식 표현과 추론을 위해서 방향을 표현할 수 있는 CSD-9 이론과 CSD-9 조합표를, 위상 관계를 표현할 수 있는 RCC-8 이론과 RCC-8 조합표를 각각 적용하였다. 하지만 SWRL을 이용한 시공간 추론 규칙과 일관성 검사를 위한 규칙을 모두 정의해야 하는 구현 방식의 한계가 있어 실용적으로 이용하기 어려운 성능을 보였다.

PelletSpatial[7]은 RDF/OWL 추론기 중 하나인 Pellet을 확장한 정성 추론기이다. RDF/OWL을 통해서 지식 베이스를 표현하고, 공간 지식 표현을 위해서 위상 관계를 표현할 수 있는 RCC-8 이론을 사용하였다. 이 추론기는 2가지 방식으로 RCC-8 공간 추론을 수행할 수 있다. 첫 번째는 RCC-8 관계를 RDF/OWL 관계로 변경한 후 추론하는 방식이고, 두 번째는 효율성 높은 경로 일관성 알고리즘[17]을 채용하여 RCC-8 조합표를 기반으로 추론하는 방식이다. 실험결과 두 번째 방식이 첫 번째 방식보다 성능 면에서 훨씬 나은 모습을 보였다. 그러나 PelletSpatial은 방향 관계에 대한 추론 기능이 없다는 단점이 있다.

CHOROS[8]는 PelletSpatial에 CSD-9 방향 관계 추론도 가능하도록 확장한 정성 공간 추론기로서 RDF/OWL을 통해서 지식 베이스를 표현하고, 공간 지식 표현을 위해서 위상 관계를 표현할 수 있는 RCC-8이론과 방향 관계를 표현할 수 있는 CSD-9을 사용했다. 또한, PelletSpatial에서 채용한 경로 일관성 알고리즘을 사용하여 공간 추론을 수행한다. 공간 추론에 사용되는 RCC-8 조합표와 CSD-9 조합표는 SOWL에서 정의한 조합표들을 사용하였고, 멀티쓰레딩 기법을 이용하여 RCC-8과 CSD-9 추론을 병렬로 수행하였다. CHOROS는 CSD-9 방향 관계와 RCC-8 위상 관계를 모두 다루지만, 방향 관계와 위상 관계를 함께 고려하여 통합적으로 추론하고, 둘 간의 일관성 검사를 할 수 없다.

QUSAR[9]는 PelletSpatial에 CSD-9 추론과 교차 추론도 가능하도록 확장한 정성 공간 추론기이다. 이 추론기는 RDF/OWL을 통해서 지식 베이스를 표현하고 RCC-8 위상 관계와 CSD-9 방향 관계를 각각 추론할뿐만 아니라, 이 둘 간의 상호 교차 일관성 검사 기능도 포함되어 있다. 상호 일관성 검사 기능이란, 위상 관계와 방향관계를 통합적으로 추론하고 일관성 검사를 하는 것으로서 CHOROS 보다 더 많은 공간 지식을 생성하고 강도 높은 공간 지식 베이스의 일관성 문제를 확인할 수 있다.

2.3 분산 컴퓨팅 환경

최근 시맨틱 웹의 발전에 따라 웹에 존재하는 엄청난 양의 정보들을 이용하여 대용량의 지식 베이스를 생성할 수 있게 되었다. 이에 따라 공간 추론에 사용되는 지식 베이스의 크기도 수십, 수백억 개로 확장되면서 단일머신 컴퓨팅 환경에서 추론 작업을 수행하기에는 성능적 한계가 존재한다. 따라서 이와 같은 성능의 한계성을 극복하기 위한 방안으로, 최근 들어 다양한 분산 병렬 컴퓨팅 개발 환경을 이용하고자 하는 노력들이 활발해졌다. MapReduce[11]는 Hadoop 클러스터 컴퓨터 시스템에서 빅데이터 처리를 가능하게 해주는 대표적인 분산 병렬 프로그래밍 프레임워크이다. 하지만 MapReduce 작업들 사이에는 파일 입출력이 자주 발생하고, 모든 작업이 Map 함수와 Reduce 함수로 구현되는 2 단계 처리과정으로 이루어지기 때문에, 작업들 간의 순차성과 반복성이 많은 정성 공간 추론기 구현에는 비-효율적이다. 기존의 Hadoop MapReduce 프레임워크의 단점을 극복하기 위해서, 인-메모리(in-memory) 방식의 분산 병렬 프로그래밍 프레임워크가 등장하게 되었는데, 대표적인 예가 Apache Spark[13]이다. Apache Spark는 자주 사용하는 데이터를 메모리에 캐싱(caching)함으로써 후속 데이터 요청시 빠르게 응답한다. 이에 따라, 계산 지연시간을 축소시키고 순차 및 반복 계산이 많은 작업에 효율적이다. Apache Spark는 RDD (Resilient Distributed Dataset)라는 읽기 전용 데이터 구조를 기반으로 하고 있으며, RDD에서 직접 RDD로의 변환을 연쇄적으로 수행할 수 있기 때문에 기존의 Hadoop MapReduce보다 10배에서 100배 정도 빠른 계산 성능을 보인다. 한편,

Apache Spark SQL[14]은 Spark 프로그램에서 SQL 질의문을 통해 구조화된 데이터 처리를 지원할 수 있도록 구현한 프로그램 개발 환경이다. 따라서 Spark SQL은 Spark 프레임워크를 기반으로 하고 있으며, 빠른 질의 처리를 위해 조인 연산의 효율적인 구현과 다양한 질의 처리 최적화 기술들을 이용하고 있다. Spark SQL에서는 Spark에서 사용되는 데이터 형식인 RDD를 스키마 기반의 데이터프레임(DataFrame) 형식으로 변환하여 사용한다. 데이터프레임은 기존 RDBMS (Relational Database Management System)의 테이블(table)과 유사한 형태로 되어있어 스키마의 이점을 이용할 수 있으며, 최적 알고리즘이 적용되어 클러스터 시스템 내의 여러 노드들에 효과적으로 분산 저장된다.

3. 정성 공간 추론

3.1 공간 지식 표현

정성 공간 추론을 위해서는 먼저 추론에 필요한 공간 지식을 표현하기 위한 공간 지식 표현법(spatial knowledge representation), 즉 공간 온톨로지(spatial ontology)를 정의할 필요가 있다. 본 연구에서 사용되는 공간 지식은 시맨틱 웹 표준 온톨로지 언어인 XML/RDF 기반의 트리플 문장(triple statement)으로 표현하고, 본 연구에서 제안하는 클래스(class)들과 술어(property)들을 사용하여 Fig. 1과 같은 공간 온톨로지를 정의하였다.

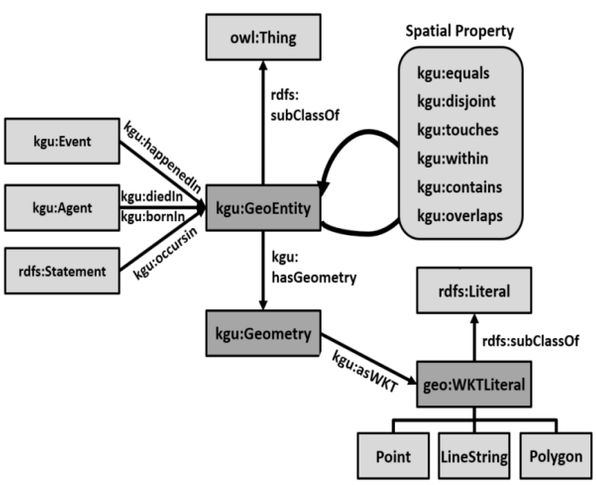


Fig. 1. Spatial Ontology

본 연구에서는 owl:Thing 클래스의 하위 클래스로 kgu:GeoEntity 클래스를 정의하였으며, 이는 실세계에서 도시, 도로, 건물 등과 같은 공간 객체를 표현할 때 사용된다. kgu:GeoEntity 클래스 간의 공간 관계를 표현하기 OGC Simple Feature 기반의 6가지 위상 관계 술어를 통해 표현할 수 있으며, Fig. 2와 같다.

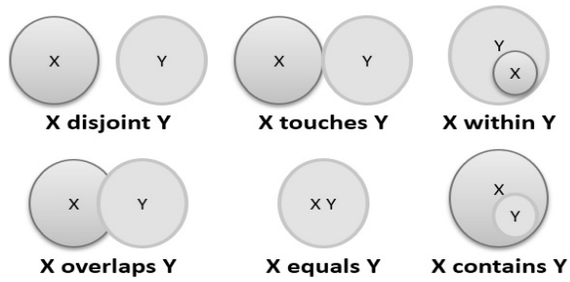


Fig. 2. Topological Properties

위상 관계 공간 술어는 공간 객체 X가 공간 객체 Y와 떨어져 있을 경우, 이를 “X disjoint Y”, X가 Y를 포함하는 경우, 이를 “X contains Y” 등과 같이 총 6가지 disjoint, touches, within, overlaps, equal, contains가 존재한다. 한편, kgu:GeoEntity 클래스는 kgu:Geometry를 하위 클래스로 가지며, 이는 공간 객체를 점(Point), 선(Line String), 면(Polygon)과 같은 기하학적 형태로 정량적으로 표현할 수 있다.

3.2 공간 추론 규칙

공간 객체들 간의 6가지의 위상 관계를 표현하고 있는 공간 지식 베이스로부터 공간 추론 규칙들을 적용함으로써 공간 객체들 간의 새로운 위상 관계를 나타내는 지식을 추가적으로 유도할 수 있다. 본 연구에서 이용하는 위상 관계 공간 추론 규칙들은 Table 1과 같다.

Table 1. Spatial Reasoning Rules

Rules	If	then
Equal	s p o (p is a spatial property)	s equals s
		o equals 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

공간 추론 규칙들은 Table 1에서 보는 것과 같이 동일 관계 추론 규칙(Equal rule), 역 관계 추론규칙(inverse rule), 이행 관계 추론 규칙(transitive rule), 정제 규칙(refining rule) 모두 4가지 종류의 추론 규칙들이 있다. 동일 관계 추론 규칙은 트리플(triple) 형태의 입력 지식 <s p o>에서 주어(subject) s와 목적어(object) o에 해당하는 공간 객체 각각에 대해서 동일 관계(equal)를 가지는 지식 <s equals s>와 <o equals o>를 각각 유도하는 규칙이며, 반면에 역 관계 추론 규칙은 기존 공간 지식 베이스의 지식 <s p o>로부터 역 관계 지식인 <o q s>를 유도하는 규칙이다. 그리고 이행 관계 추론 규칙은 공간 객체 x와 y 간의 공간 관계 지식 <x p1 y>와 공간 객체 y와 z 간의 공간 관계 지식

Table 2. Composition Table for Transitive Reasoning

	disjoint	touches	overlaps	within	contains	equals
disjoint	*	[disjoint | touches | overlaps | within]	[disjoint | touches | overlaps | within]	[disjoint | touches | overlaps | within]	disjoint	disjoint
touches	[disjoint | touches | overlaps | contains]	*	[disjoint | touches | overlaps | within]	[touches | overlaps | within]	[disjoint | touches]	touches
overlaps	[disjoint | touches | overlaps | contains]	[disjoint | touches | overlaps | contains]	*	[overlaps | within]	[disjoint | touches | overlaps | contains]	overlaps
within	disjoint	[disjoint | touches]	[disjoint | touches | overlaps | within]	within	*	within
contains	[disjoint | touches | overlaps | contains]	[touches | overlaps | contains]	[overlaps | contains]	[overlaps | within | contains | equals]	contains	contains
equals	disjoint	touches	overlaps	within	contains	equals

<y p2 z>로부터, 공간 객체 x와 z 간의 새로운 공간 관계 지식 <x [p1|p2|...|pn] z>를 유도하는 규칙이다.

이행 관계 추론 규칙들은 Table 2와 같이 하나의 조합표(composition Table)로 나타낼 수 있다. Table 2의 조합표는 가로 행과 세로 열이 각각 disjoint, touches, overlap, within, contains, equals로 총 5가지의 공간 관계 술어들로 구성되어 있다. 조합표에 포함된 이행 관계 추론 규칙들을 해석하는 방법은 다음과 같다. 가로 행의 지식과 세로 열의 지식이 동시에 참이라면, 해당 행과 열이 교차하는 난에 열거된 새로운 지식들을 조합해 낼 수 있다. 예를 들어, <A overlaps B>와 <B within C>로부터 이행 관계 추론을 통해 <A [overlaps | within] C>를 지식을 유도해낸다. 마지막으로, 관계 정제 추론 규칙은 공간 추론 과정동안 동일한 두 객체 x와 y에 관해 <x [p1|p2|...|pm] y>, .., <x [p1|p2|...|pn] y> 등과 같이 여러 관계 지식들이 유도될 때, 이들 간의 포함 관계를 고려하여 간결한 하나의 지식 <x [p1|p2|...|pk] y>으로 정제하는 규칙을 나타낸다.

3.3 추론 효율화

효율적인 공간 추론을 수행하기 위해 본 연구에서는 두 가지 효율화 방법을 제안한다. 첫째로 공간 추론에 필요한 최소 이접 관계들(minimal disjunctive relations)를 찾아내고, 이들을 토대로 이행 관계 추론을 위한 조합표의 축소(reduction of the composition table)를 제안한다.

Table 2의 조합표에서 보듯이, 이행 관계 추론을 통해 두 공간 객체 사이에 만족 가능한 새로운 공간 관계들이 두 개 이상 존재할 수 있다. 이러한 경우, 복수의 공간 술어들로 새로운 공간 관계를 나타내게 되는데, 이때 공간 관계를 이접 관계(disjunctive relations)라고 부른다. 따라서 정성 공간 추론기는 단일 술어로 표현되는 단일 공간 관계 지식들뿐만 아니라 이접 관계 지식들에 대해서도 이행 관계 추론이 가능해야 한다. 예를 들어 Fig. 3과 같이 기존의 지식 <B within A>와 <A touches E>로부터 <B [disjoint | touches] E>라는 새로운 이접 관계 지식이 추론 되고, <C within B>와 <D overlaps C>로부터 <D [overlaps | within] B>라는 새로운 이접 관계가 추론될 뿐만 아니라, 다시 이 두 이접 관계 지식들로부터 공간 객체 D와 E 사이의 새로운 이접 관계 지식이 추론될 수 있어야 한다.

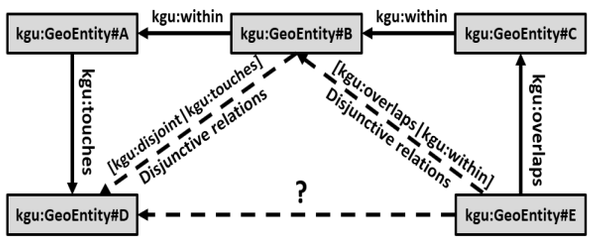


Fig. 3. Transitive Reasoning Over Disjunctive Relations

$$
\begin{aligned}
&< E\ [\ overlaps \mid within\ ]\ B > \circ\ < B\ [\ disjoint \mid touches\ ]\ D > \\
&\Rightarrow E\ [\ overlaps \circ disjoint\ ]\ D \cup E\ [\ overlaps \circ touches\ ]\ D \cup \\
&\quad E\ [\ within \circ disjoint\ ]\ D \cup E\ [\ within \circ touches\ ]\ D \\
&\Rightarrow E\ [\ disjoint \mid touches \mid overlaps \mid contains]\ D \cup E\ [\ disjoint \mid touches \mid overlaps \mid contains]\ D \\
&\quad \cup\ E\ disjoint\ D \cup E\ [\ disjoint \mid touches]\ D \\
&\Rightarrow E\ disjoint\ D
\end{aligned}
$$

Fig. 4. Transitive Reasoning Over Two Disjunctive Relations

따라서 이행 관계 추론 규칙들은 단일 공간 관계 지식들을 유도하는 결정적 규칙(deterministic rule)들 뿐만 아니라, 복수의 이접 관계 지식들을 유도하는 비-결정적 규칙(non-deterministic rule)들도 다수 포함하고 있다. 이행 관계 추론이 반복되면 이러한 비-결정적 추론 규칙들로 인해 다수의 이접 관계 지식들이 유도되고, Fig. 4와 같이 이접 관계 지식들로부터 또 다른 이접 관계 지식들이 확대 재생산되기 때문에 추론 과정이 매우 복잡해진다. 따라서 공간 추론기 내부에서 이접 관계 지식들에 대한 이행 관계 추론 작업을 실제로 구현하기 위해서는 Table 2의 단일 관계 조합표 대신 이접 관계들의 가능한 모든 조합들을 나타내는 이접 관계 조합표가 있어야 한다. 하지만 위상 관계를 나타내는 모든 이접 관계 집합의 개수는 단일 관계를 나타내는 공간 술어 집합의 가능한 모든 부분 집합의 개수에 해당하는 2^6=64개와 같다. 이를 바탕으로 확장된 이행 관계 조합표를 구성할 경우 표의 크기가 $2^6 \times 2^6$=4096이므로, 조합표의 복잡성도 높아지고 조합표를 저장하기 위한 메모리 사용량도 커져, 공간 추론 성능에 한계를 나타낸다. 앞서 언급한 문제점을

해결하기 위해서 본 연구에서는 시뮬레이션 추론과정을 거쳐 공간 추론에 실제로 사용되는 최소 이접 관계들의 집합을 구하였다. 그 결과 기존의 $2^6=64$ 개의 이접 관계 집합에서 Table 3과 같이 17개의 이접 관계들만이 추론과정에서 실제로 발생하며 나머지 이접 관계들은 생성되지 않는다는 사실을 발견하였다. 이와 같은 발견을 토대로, 본 연구에서는 $2^6*2^6=4096$ 크기의 이접 관계 조합표를 $17*17=289$로 축소 정의 하였다. 이를 통해서, 이행 관계 추론을 위한 조합표의 복잡성을 감소시켜 추론 성능을 향상시켰다.

Table 3. Minimal Disjunctive Relations

Disjunctive Relations(17)
equals, contains, within, overlaps, touches, disjoint, overlaps | within, contains | overlaps, disjoint | touches, overlaps | touches, disjoint | overlaps | touches, contains | overlaps | touches, overlaps |within | touches, contains | overlaps | within | equals, disjoint | overlaps | within | touches, disjoint | contains | overlaps | touches, contains | overlaps | within | touches | equals.

본 연구에서 제안하는 공간 추론의 두 번째 효율화 방안은 관계 정제를 위한 최적화 기법이다. 공간 추론 과정동안 동일한 두 공간 객체 x와 y에 대해, 서로 다른 여러 공간 관계 지식들이 유도될 수 있다. 이때 이 지식들 간에 서로 모순이 없는지 일관성 검사(consistency check)를 수행하고, 모순이 없는 경우에는 이 지식들 간의 포함관계를 고려하여 하나의 지식으로 간결하게 통합 정리하는 작업이 관계 정제이다. 이 관계 정제는 지식들 간의 모순을 발견함으로써 공간 추론의 조기 종료 시점을 알아낼 수 있을 뿐만 아니라, 두 공간 객체사이에 새로 발견해내는 공간 관계 지식들을 수시로 통합 정리해주는 역할을 수행하기 때문에 공간 추론기의 성능에 매우 중요한 영향을 미치는 처리 과정이 된다. 본 연구에서는 앞서 발견한 최소 이접 관계들을 이용하여 아래와 같이 관계 정제 최적화 방법을 제시한다.

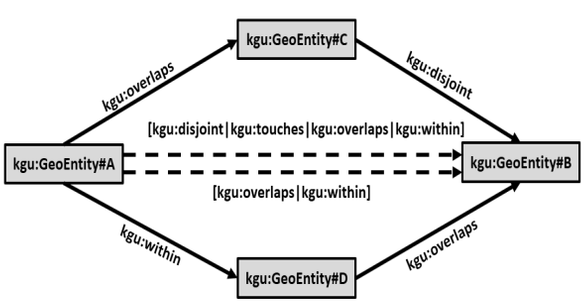


Fig. 5. Illustration of Relation Refining

예를 들어, Fig. 5와 같이 기존의 지식 <A overlaps C>와 <C disjoint B>로부터 <A [disjoint | touches | overlaps | within] B>라는 이접 관계가 추론 되고, <A within D>와 <D overlaps B>로부터 <A [overlaps | within] B>라는 이접 관계가 추론 될 수 있다. 새롭게 추론된 두 지식은 주어와 목적어 관점은 같지만, 공간 술어가 서로 다르므로 교집합 연산을 통해 최종적으로 <A [overlaps | within] B>라는 정제된 지식을 얻을 수 있다. 이와 같은 정제 작업은 분산 환경에서 매우 큰 비용이 요구된다.

본 연구에서는 분산 환경에서 효율적인 관계 정제를 하기 위해서 미리 정제 가능 여부를 미리 계산했다. 하지만 공간 추론을 통해 얻는 지식 중에는 주어와 목적 관점이 같은 수 있는 지식에 대하여 관계 정제의 대상이 되는 모든 경우의 수가 2^{64}개다. 예를 들어 Fig. 5와 같이 두 공간 객체 'kgu:GeoEntity#A'와 'kgu:GEntity#B' 사이의 관계인 '[kgu:disjoint|kgu:touches|kgu:overlaps|kgu:within]'와 '[kgu:overlaps|kgu:within]'은 관계 정제 대상이 되는 2^{64}개 경우 중 하나이다. 하지만 2^{64}개의 정제 가능한 경우에 대하여 정제 가능여부를 미리 계산 하는 것은 메모리를 매우 많이 요구한다. 본 연구에서는 효율적으로 정제 가능 여부를 계산하기 위해 앞서 이접관계를 17개로 축소시킨 것을 바탕으로 정제 가능한 경우의 수를 2^{64}개 에서 2^{17}개로 축소시켰다. 그리고 2^{17}개의 정제 가능한 경우의 수 중에서 정제 가능 여부를 미리 모두 계산하여 함수화 한다.

4. 대용량 정성 공간 추론기의 설계

본 연구에서는 분산 컴퓨팅 환경에서 메모리를 사용하여 대용량 공간 지식 베이스의 추론을 신속하게 가능케 하는 Apache Spark SQL[14] 프레임워크를 기반으로 설계했다. Apache Spark SQL은 Spark의 메모리 기반으로 생성되는 RDD(Resilient Distributed Datasets)의 독특한 데이터 구조를 바탕으로 RDBMS(Relational Database Management System)와 비슷한 테이블 형식의 데이터프레임(DataFrame)을 사용한다. 이를 통해서 특정 데이터를 처리할 때 키(key) 범위의 일부 데이터만 추출할 수 있고, 결과를 갖고 올 때 아주 적은 데이터만 읽어서 처리할 수 있다. SQL 질의연산을 통해 대용량 데이터 인덱싱(indexing) 및 특정 칼럼 집합에 대한 검색이나 조인(join) 연산이 효과적이다. 또한, Spark SQL은 사용자가 분산 컴퓨팅 환경에서의 데이터 분석과 코드를 직접 작성할 필요 없이 프레임워크 자체로 질의문을 분석하여, 최적화된 전략을 자동으로 수립하고 수행하기 때문에 대용량 데이터를 관리하는 응용 시스템 구현이 쉽다. 본 연구에서는 분산 컴퓨터 메모리를 효율적으로 사용하여 대용량 공간 지식 베이스를 추론하기 위한 과정에서 일어나는 공간 추론 작업들의 실행 순서와 각 추론 작업의 세부적인 처리 방법에 대해서 소개한다.

4.1 공간 추론 작업 순서

정성 공간 추론을 위한 전체 과정을 어떤 Apache Spark SQL 작업 단위들로 나누고, 이들 간의 실행 순서를 어떻게 결정하느냐는 추론기의 성능에 영향을 미치는 가장 중요한 설계 요소 중 하나이다. 본 공간 추론기에서는 정성 공간 추론의 전

체 과정을 지식 인코딩(knowledge encoding), 역 관계 추론(inverse reasoning), 동일 관계 추론(equal reasoning), 이행 관계 추론(transitive reasoning), 관계 정제(refining), 지식 디코딩(knowledge decoding) 등 크게 총 6개의 작업들로 나누고, 논리적 인과관계와 계산 효율성을 고려하여 Fig. 6과 같이 작업들 간의 수행 순서를 결정하였다.

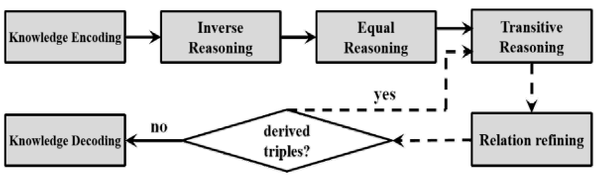


Fig. 6. Job Flow for Qualitative Spatial Reasoning

추론을 위해 가장 먼저 수행하는 작업은 추론의 전처리 과정(preprocessing)인 지식 인코딩(knowledge encoding) 작업이다. 이 작업을 통해서 입력 지식 베이스의 용량을 대폭 감소시켜서 추론의 효율성을 높인다. 다음으로, 역 관계 추론 작업을 한 뒤 동일 관계 작업 추론 작업을 한다. 동일 관계 작업을 먼저 하게 된다면 주어와 목적어에 대하여 모두 동일 관계 추론을 해야 하지만, 역 관계 작업을 먼저 하게 된다면 역 관계 추론이 끝난 확장된 지식 베이스에 대해서 주어와 목적어 둘 중 하나에 대해서만 동일 관계 추론 작업을 하면 된다는 이점이 있다. 이어서 이행 관계 추론 작업에서는 이행 관계를 통해 새로운 지식을 추론한다. 이 작업은 술어 이행 관계를 사용하기 때문에, 역 관계 추론, 동일 관계 추론 작업 이후에 수행한다. 이어서 관계 정제 작업에서는 이행 관계 추론을 통해 얻어진 지식들 간의 모순이 있는지 검사하고, 모순이 없다면 이 지식들을 보다 간결한 하나의 지식으로 통합 정리하는 일을 수행한다. 마지막으로 이와 같은 추론 작업들을 통해 신규로 얻어진 지식이 있는지 판단하고, 추론된 신규 지식이 없다면 공간 추론을 종료하고, 만약 추론된 신규 지식이 있다면 다시 이행 관계 추론 작업과 관계 정제 작업을 반복한다.

4.2 인코딩과 디코딩 작업

메모리 기반 대용량 추론 환경에서는 메모리 크기에 비해서 데이터의 크기가 크면 추론 성능에 한계가 있다. 따라서 이를 극복하기 위해 추론에 앞서 인코딩 작업(knowledge encoding)을 통해서 방대한 양의 공간 지식 베이스의 크기를 줄였다. 이 작업은 XML/RDF 형식의 문자열로 표현된 공간 지식을 지식의 각 구성 요소인 주어(subject), 술어(property), 목적어(object) 각각에 대응하는 식별자 코드로 대신 변환하여 지식의 크기를 줄이는 작업이다.

Fig. 7은 지식 인코딩 작업을 예시하고 있다. 이 작업에서는 “<http://ailab.kyonggi.ac.kr/A http://ailab.kyonggi.ac.kr/B http://ailab.kyonggi.ac.kr/C>”와 같이 URI들을 포함하는 긴 문자열 형태의 트리플을 사전 테이블(dictionary table)을 참조하여 “<1 16 2>”와 같이 숫자로 된 식별자(ID)들의 조합

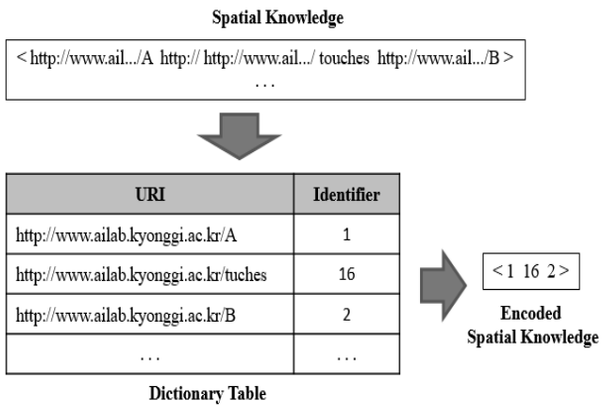


Fig. 7. Illustration of Knowledge Encoding

으로 변환한다. 지식 인코딩 작업은 원래 지식으로부터 인코딩 테이블을 생성하는 세부 작업과 이 인코딩 테이블을 이용하여 원래 지식을 식별자 번호들로 변환하는 세부 작업들로 구성된다.

사전 테이블(dictionary table)에는 주어, 술어, 목적어에 해당하는 자원들의 URI와 그들에 대응되는 각각의 식별자들이 함께 저장된다. 다양한 공간 객체를 나타낼 수 있는 주어와 목적어와는 달리, 위상 및 방향 관계를 나타내는 술어의 종류는 미리 정의할 수 있기 때문에 인코딩 작업 이전에 Table 4와 같은 술어 사전 테이블(property dictionary table, P_Dictionary)을 추론기 내부에 저장하고 있다가 인코딩 작업을 수행할 때 이용한다. 본 연구에서는 Table 4의 단일 관계 술어들뿐만 아니라, 이들을 결합한 이접 관계 술어들도 각기 다른 고유한 식별자 번호를 하나씩 부여하는 방식을 사용한다.

Table 4. Example of Property Dictionary Table

URI	Identifier
http://www.ailab.kyonggi.ac.kr/equals	1
http://www.ailab.kyonggi.ac.kr/contains	2
http://www.ailab.kyonggi.ac.kr/within	4
http://www.ailab.kyonggi.ac.kr/overlaps	8
http://www.ailab.kyonggi.ac.kr/tuches	16
http://www.ailab.kyonggi.ac.kr/disjoint	32

앞서 설명한대로 지식 인코딩을 위한 첫 번째 작업은 원래 지식의 주어와 목적어가 나타내는 공간 객체들에 대한 주어-목적어 사전 테이블(subject-object dictionary table, SO_Dictionary)을 만드는 작업이며, 두 번째 작업은 술어 사전 테이블과 주어 술어 사전 테이블을 참조하여 원래 지식을 식별자 번호들로 변환하는 작업이다. 이 각각의 작업은 다수의 노드들로 구성된 Hadoop 클러스터 컴퓨터 시스템에서 Spark SQL 질의문을 통해 구현되며, 테이블 형식의 데이터프레임(DataFrame) 자료구조를 사용한다. 이후 본 논문

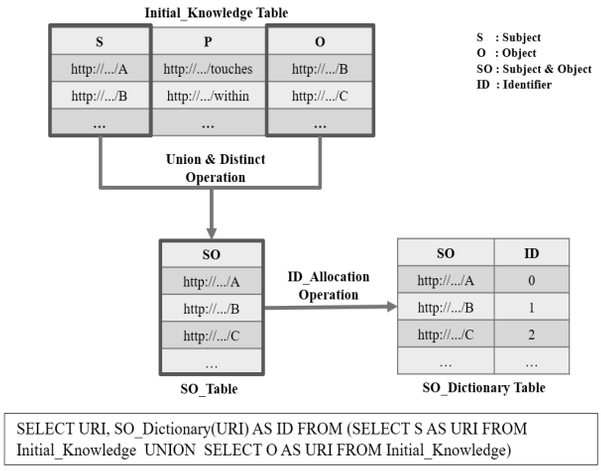


Fig. 8. Dictionary Building Job

의 Spark SQL 질의문들에 등장하는 테이블들은 모두 데이터프레임 자료구조임을 가정한다. 주어, 목적어에 대한 리소스 인코딩 테이블을 만드는 과정과 이를 수행하는 Spark SQL 질의문은 Fig. 8과 같다.

먼저, 트리플 형태의 공간 지식을 포함하고 있는 대용량 XML/RDF 파일로부터, 주어(subject), 술어(property), 목적어(object)를 각각의 컬럼으로 가지는 초기 지식 테이블(Initial_Knowledge Table)을 생성한다. 그리고 이 초기 트리플 테이블로부터 각각 주어와 목적어 부분을 추출해내고 이들에 대해 합집합(union) 연산과 중복제거(distinct) 연산을 수행하여 주어-목적어(SO_Table)을 만든다. 그런 다음, 주어와 목적어에 고유한 식별자(ID)를 부여하여 주어-목적어 사전 테이블(SO_Dictionary Table)을 생성한다. 이 때, 해쉬맵(hash map) 형태인 주어-목적어 사전은 키(key)로 URI, 값(value)으로 식별자를 가진다. URI를 조회하여 만약 그에 해당하는 식별자가 없다면 새로운 식별자를 부여한다. 술어 사전 테이블(P_Dictionary Table)과 주어-목적어 사전 테이블(SO_Dictionary Table)을 참조하여 원래 지식을 식별자 번호들로 변환하는 작업은 주어 인코딩(subject encoding), 목적어 인코딩(object encoding), 술어 인코딩(property encoding) 순으로 진행된다.

Fig. 9는 주어 인코딩 작업에 대한 예시와 이 작업을 구현하기 위한 Spark SQL 질의문을 나타내고 있다. 이 작업을 위해서는 먼저 초기 지식 테이블(Intial_Knowledge Table)의 주어(subject) 필드와 주어-목적어 사전 테이블(SO_ Dictionary Table)의 URI 필드를 매개로 두 테이블들에 대한 조인 연산을 수행한다. 조인 결과 테이블로부터 필요한 컬럼들만을 선택함으로써 주어가 인코딩된 지식 테이블(S_Encoded_Knowledge Table)을 생성한다.

Fig. 10은 목적어 인코딩 작업에 대한 예시와 작업 구현을 위한 Spark SQL 질의문을 나타낸다. Fig. 9의 주어 인코딩 작업과 유사한 처리를 통해 주어뿐만 아니라 목적어까지 인코딩된 지식 테이블(SO_Encoded_Knowledge Table)을 생성한다.

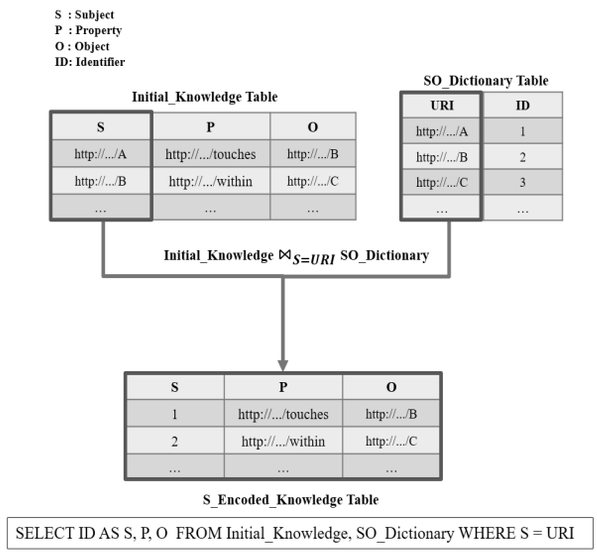


Fig. 9. Subject Encoding Job

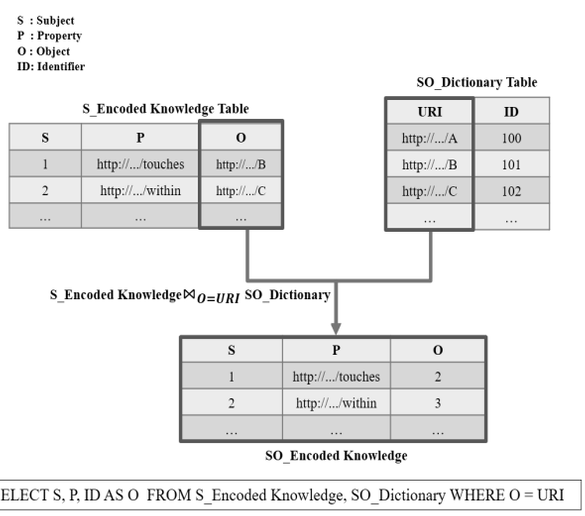


Fig. 10. Object Encoding Job

한편, Fig. 11은 술어 인코딩 작업에 대한 예시와 이 작업을 구현하기 위한 Spark SQL 질의문을 나타낸다. 술어 인코딩 작업에서는 주어, 목적어 인코딩 작업에 이용되었던 주어-목적어 사전 테이블(SO-Dictionary Table) 대신, 술어 사전 테이블(P_Dictionary Table)을 이용한다는 점 외에는 앞선 두 인코딩 작업과 유사한 연산들을 수행하여 주어, 목적어, 술어가 모두 인코딩 된 지식 테이블(Fully_Encoded_Knowledge Table)을 얻게 된다. 한편, 인코딩된 공간 지식 베이스를 이용한 정성 공간 추론이 완전히 완료된 후에는 추론된 모든 공간 지식들을 다시 원래의 XML/RDF 형태로 환원하기 위한 지식 디코딩 작업(knowledge decoding job)이 수행된다. 지식 디코딩 작업에서는 리소스 인코딩 테이블과 술어 인코딩 테이블을 함께 참조하여, 각각의 지식을 표현하는 주어, 목적어, 술어의 식별자 번호를 다시 원래의 URL로 변환하는 일을 수행한다.

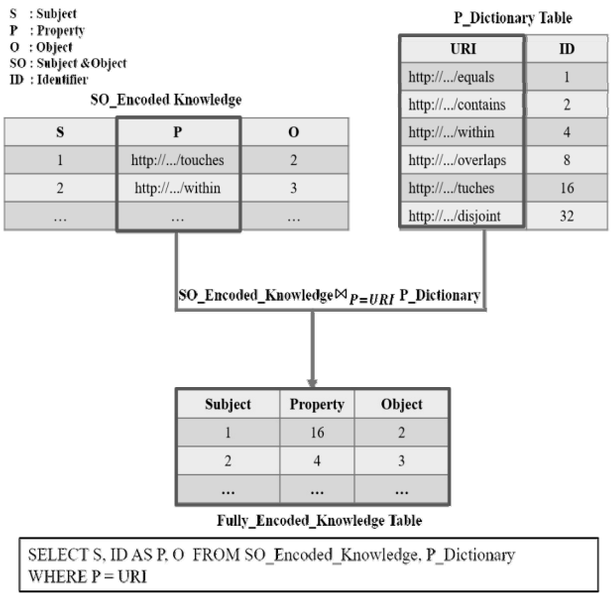


Fig. 11. Property Encoding Job

4.3 역 관계 및 동일 관계 추론 작업

앞서 서술한 공간 추론 작업 순서에 따라 각 추론 규칙과 Spark SQL 질의 연산을 통한 공간 추론을 수행한다. 또한 공간 추론 작업은 각 단계마다 추론된 지식을 기존 지식에 확장하여 다음 단계로 넘어간다. 먼저, 인코딩된 지식 테이블(Fully_Encoded_Knowledge Table)로부터 역 관계 추론 작업을 한다. 이 작업은 입력으로 들어온 모든 공간 지식에 대해 역 관계들을 생성한다. 역 관계 추론 과정과 이를 수행하는 Spark SQL 질의문은 Fig. 12와 같다. 인코딩 된 지식 테이블(Fully_Encoded_Knowledge Table)의 <주어, 술어, 목적어>로 구성된 스키마로부터 셀렉트(select) 연산을 사용하여 <목적어, 역 술어, 주어> 형태로 변환된 역 지식 테이블(Inverse_Knowledge Table)을 생성한 뒤, 인코딩된 지식 테이블과 합집합(union) 연산을 하여 확장된 지식 테이블(Expanded_Knowledge Table)을 생성한다.

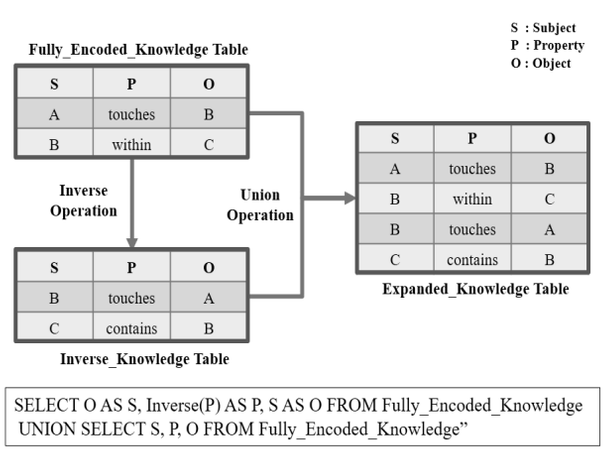


Fig. 12. Inverse Reasoning Job

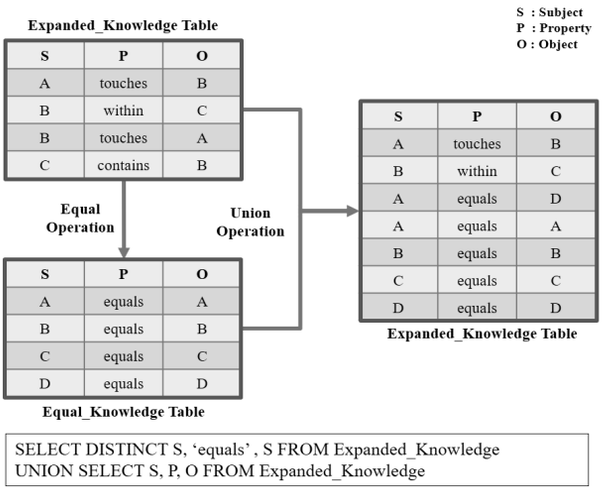


Fig. 13. Equal Reasoning Job

역 관계 추론 작업이 완료 된 후에는 동일 관계 추론 작업을 수행한다. 동일 관계 추론 작업 수행과정은 Fig. 13과 같다.

역 관계 추론 작업으로부터 생성된 확장된 지식 테이블(Expanded_Knowledge Table)의 <주어, 술어, 목적어>로 구성된 스키마로부터 선택(select) 연산을 사용하여 <주어, equals, 주어> 형태로 변환된 동일 지식 테이블(Equal_ Knowledge Table)을 생성한다. 확장된 트리플 테이블과 합집합(union)연산을 통해 동일 관계가 추가된 확장된 지식 테이블(Expanded_ Knowledge Table)을 생성한다.

4.4 이행 관계 추론 작업

이행 관계 추론 작업을 수행하는 과정과 이를 위한 Spark SQL 질의문은 Fig. 14와 같다. 이전 작업으로 확장된 지식 테이블(Expanded_Knowledge Table)에서 이행 관계 추론의 조건인 특정 지식의 목적어와 또 다른 지식의 주어가 일치하는 두 개의 지식을 효율적으로 찾기 위해, 확장된 지식 테이블에 특

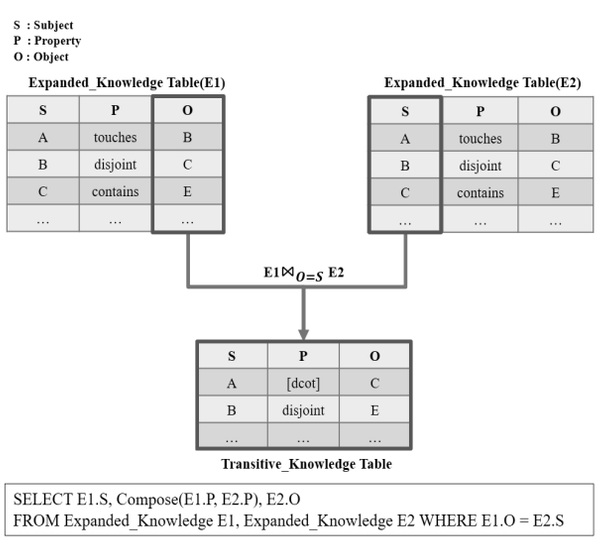


Fig. 14. Transitive Reasoning Job

정 지식의 목적어와 또 다른 지식의 주어가 같은 두 개의 지식을 대상으로 동일-조인(equi-join) 연산을 수행한다. 이행 관계 추론을 통해 새로운 지식을 추론하여 이행 추론된 지식 테이블(Transitive_Knowledge Table)을 생성한다.

4.5 관계 정제 작업

본 연구에서는 분산 환경에서 지식들 간의 관계 정제 가능한 여부를 빨리 판별하고 정제 결과를 효과적으로 계산하기 위한 효율적인 방법을 제시한다. 관계 정제 대상이 되는 술어 집합에 대해서 효율적으로 관계 정제를 하기 위해 술어 집합에 대한 정제된 값을 3.3장에서 언급했듯이 해쉬 맵(hash map) 형태의 함수로 만들었다. 키(key)는 정제의 대상이 되는 술어 집합, 그리고 값(value)은 정제 가능 여부와 정제된 결과로 하였다. 하지만 키 값인 정제 대상이 되는 술어 집합의 개수는 미리 알 수 없기 때문에, Table 4 처럼 술어에 인덱스를 부여하고, 이에 대한 합을 키로 사용했다. 이때 중복되는 경우를 배제하기 위해서 술어의 인덱스를 1부터 2의 배수 형태로 부여하여 모든 경우의 술어 집합 합의 값, 즉 키가 서로 다른 값을 가지도록 했다. 관계 정제 수행 과정과 Spark SQL 질의문은 Fig. 15와 같다.

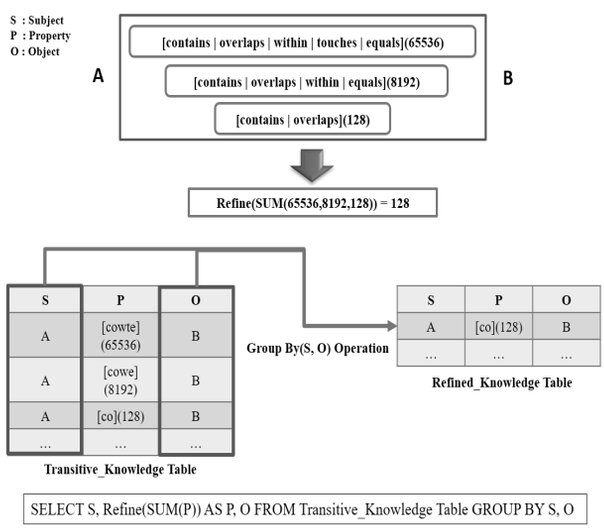


Fig. 15. Relation Refining Job

이전 작업으로 이행 관계 추론된 테이블(Transitive_Knowledge Table)에서 관계 정제 추론의 대상이 되는 주어와 술어가 일치하는 지식들을 효율적으로 찾기 위해, 이행 관계 추론된 테이블 특정 지식의 술어가 같은 지식들을 대상으로 그룹화(group by) 연산을 수행한다. 그룹화 결과를 이용해 그룹화 조건은 주어, 목적어이고, 술어 집합에 대하여 합(sum) 연산을 한 뒤, 합에 해당하는 정제된 값을 정제 함수(refine function)를 통해 얻은 결과를 술어로 가지는 정제된 지식 테이블(Refined_Knowledge Table)을 생성한다. 그리고 더 이상 신규 지식들이 생성되지 않을 때까지, 이행 관계 추론 작업과 정제 추론 작업을 반복 수행한다.

5. 구현 및 실험

5.1 대용량 공간 추론기의 구현

본 연구에서는 순차 및 반복 작업을 요구하는 정상 공간 추론에 적합한 인-메모리 기반의 Apache Spark SQL[14]을 이용하여 대용량 정성 공간 추론기인 SSQUSAR를 구현했다. 구현 환경은 Java 1.7 버전과 Apache Spark 1.4.0 버전을 사용하였으며, 추론기 구조는 Fig. 16과 같다. Apache Spark 클러스터 관리자(Spark Cluster Manager)는 하둡 분산 파일 시스템(HDFS-Hadoop Distributed File System)에 있는 공간 지식 파일을 분산 클러스터를 구성하는 각 노드에 데이터 프레임 형식으로 변환하여 적재한 후, 카탈리스트 최적기(Catalyst Optimizer)를 통해서 공간 추론 작업을 생성한다. 그리고 각각의 작업자 노드(Worker Node)가 어떤 작업을 수행할 것인지 작업 스케줄(schedule)을 정한다. 그 후, 공간 추론 작업을 수행하기 위한 데이터와 작업 태스크(task)를 가져온 후 작업을 수행하며, 수행 결과로 얻은 추론 지식들은 HDFS(Hadoop Distributed File System)에 저장한다.

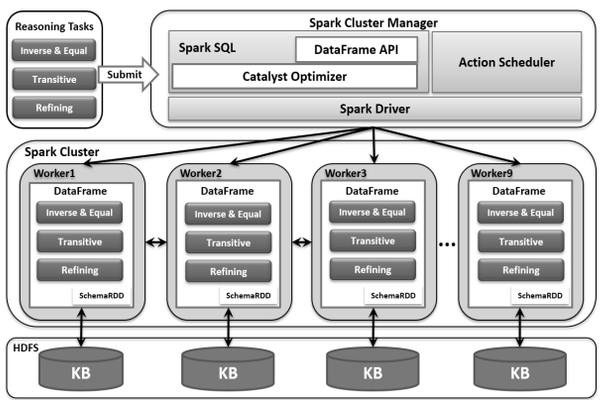


Fig. 16. Architecture of the Qualitative Spatial Reasoner

5.2 성능 실험

본 연구에서 제안하는 기법의 대용량 정성 공간 추론기의 성능을 평가하기 위해서 Table 5와 같이 인공적으로 제작한 대용량의 가상공간 지식 베이스들(Artificial Spatial Knowledge Base, ASK)를 이용한 실험들을 수행하였다. 실험에 이용된 벤치마크용 공간 지식 베이스는 포함된 트리플(triple) 개수에 따라 ASK1000부터 ASK50000K까지 이름을 부여하였다. 실험을 위한 클러스트 컴퓨터 시스템은 1개의 마스터 노드와 9개의 슬레이브 노드들로 구성하였으며, 각 노드는 3.5GHz, 4 Core CPU와 8GB 메인 메모리, 1TB 하드 디스크로 구성하였다.

첫 번째 실험에서는 신규 추론된 지식의 양적(amount of derived knowledge)인 면에서 본 연구에서 제안하는 공간 추론기의 성능을 분석해보았으며, 실험 결과는 Fig. 17과 같다. Fig. 17은 초기 입력 지식 베이스의 크기 증가에 따른 신규 추론된 지식 베이스의 크기를 비교해주고 있다. 그림에서 초기 입력 지식 베이스의 크기가 증가할수록 비례해서

Table 5. Knowledge Bases used for Experiments

Knowledge Base	Number of Spatial Objects	Amount of Spatial Knowledge	Size of Files
ASK1000K	1,000,000	1,000,000	174MB
ASK3000K	3,000,000	3,000,000	527MB
ASK5000K	5,000,000	5,000,000	879MB
ASK7000K	7,000,000	7,000,000	1.2GB
ASK10000K	10,000,000	10,000,000	1.7GB
ASK20000K	20,000,000	20,000,000	3.42GB
ASK30000K	30,000,000	30,000,000	5.15GB
ASK40000K	40,000,000	40,000,000	6.87GB
ASK50000K	50,000,000	50,000,000	8.60GB

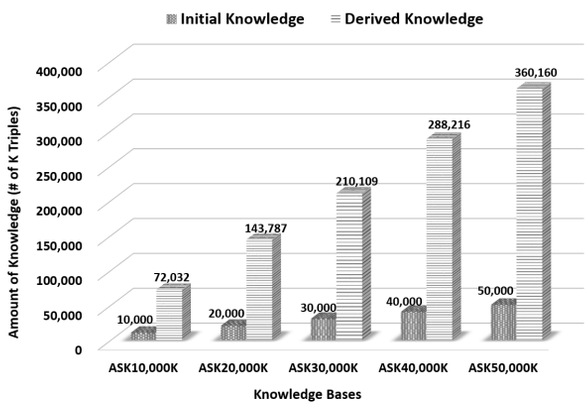


Fig. 17. Amount of Derived Knowledge

더 많은 신규 지식을 추론하였음을 알 수 있을 뿐만 아니라, 약 5,000만개의 트리플(triple)들로 구성된 대용량의 초기 지식 베이스 ASK5000K로부터 본 논문에서 제안한 정성 공간 추론기 SSQUSAR은 약 3억 6,000만개가 넘는 신규 지식들을 유도해내었다는 사실을 알 수 있다. 이와 같은 실험 결과를 통해서, 본 연구에서 제안한 정성 공간 추론기의 높은 신규 지식 추론 성능을 확인할 수 있다.

두 번째 실험에서는 추론 시간(reasoning time) 면에서 본 연구에서 제안하는 공간 추론기의 성능을 분석해보았다. 또한 이와 동시에, 본 연구에서 제안한 지식 인코딩 작업의 효과를 확인해보기 위해, 인코딩 하지 않은 공간 추론(Non-Encoding)의 경우와 인코팅을 수행한 공간 추론(Encoding)의 경우를 추론 시간 면에서 비교해 보았다.

실험 결과는 Fig. 18과 같다. 실험 결과들 중 대표적으로, 약 1,000만개의 지식으로 구성된 대용량 공간 지식 베이스 ASK10000K에 대해 본 연구에서 제안한 방식대로 인코딩 후 공간 추론을 수행한 경우(Encoding)에는 6분 내외의 시간에 추론을 완료한 것을 알 수 있다. 반면에, 인코딩을 하지 않은 공간 추론을 수행한 경우(Non-Encoding)는 약 23분의 추론 시간을 소모하였다. 이러한 실험 결과를 통해서, 본 연구에서 제안하는 인코딩 기법의 긍정적 효과를 확인할 수 있을 뿐 아니라, 본 연구의 공간 추론기가 추론 시간 면에서 높은 성능을 보인다는 것을 확인할 수 있었다.

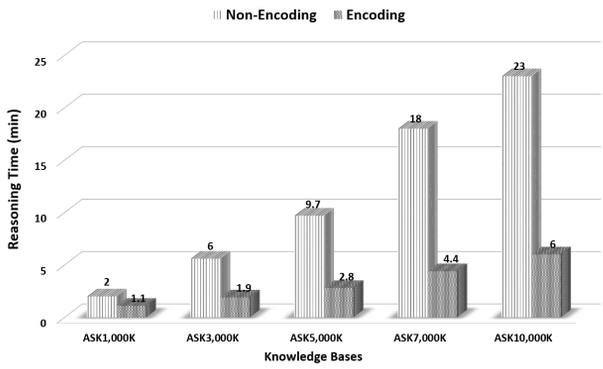


Fig. 18. Total Amount of Reasoning Time

세 번째 실험에서는 본 연구에서 제안한 추론 효율화 기법들이 공간 추론기의 성능에 미치는 효과를 분석해보았다. 본 연구에서는 축소된 이접 관계들과 조합표를 이용하는 이행 관계 추론 방식RCT(Reduced Composition Table)과 질의연산을 이용한 효율적인 관계 정제 추론 기법 IRT (Improved Refining Technique)을 제안하였다. 각 기법의 효과를 분석하기 위해, 실험에서는 RCT 기법과 IRT 기법을 사용하지 않은 기본적인 공간 추론의 경우와 이 기법들을 사용한 추론의 경우를 조합하여 총 4 가지 경우들에 - RCT (x) IRT (x), RCT (o) IRT (x), RCT (x) IRT (o), RCT (o) IRT (o) - 대해 추론 시간을 비교하였으며, 실험 결과는 Fig. 19와 같다. 실험 결과를 살펴보면, RCT 기법과 IRT 기법 중 어느 하나만 적용한 추론의 경우에도 추론 시간이 단축되는 긍정적인 효과를 확인할 수 있으며, 이 중에서 RCT 기법만 적용한 경우에 비해서는 IRT 기법만을 적용한 추론의 경우에 성능 향상 효과가 상대적으로 더 크다는 것을 알 수 있다. 전체적으로는 본 연구에서 제안하는 추론 방식대로 이 2 가지 효율화 기법들을 모두 적용한 추론의 경우가 가장 높은 성능을 보였다. 예컨대, 약 5000만개의 공간 지식을 포함하고 있는 공간 지식 베이스 ASK50,000에 대해 추론을 수행하였을 때, RCT 기법과 IRT 기법을 모두 적용한 추론의 경우가 이들을 전혀 적용하지 않은 기본적인 공간 추론의 경우에 비해 약 25분 정도의 추론 시간이 감소하는 효과를 보였다. 이러한 실험 결과를 통해, 본 연구에서

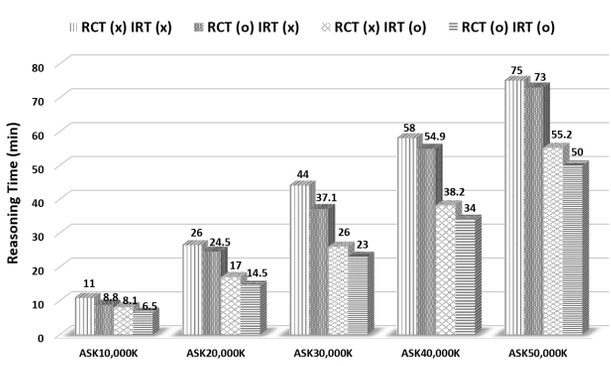


Fig. 19. Comparison of Different Reasoning Methods in Terms of Reasoning Time

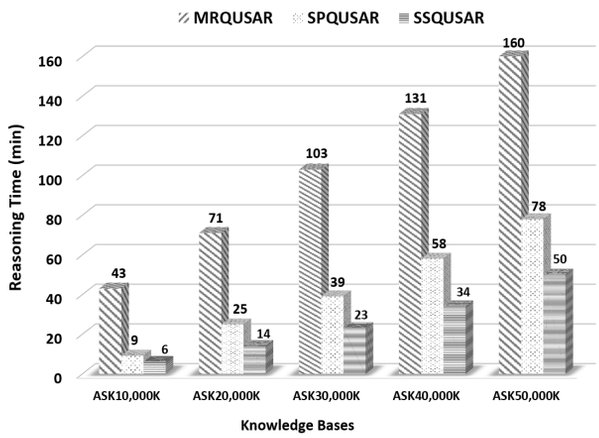


Fig. 20. Comparison of Different Spatial Reasoners in Terms of Reasoning Time

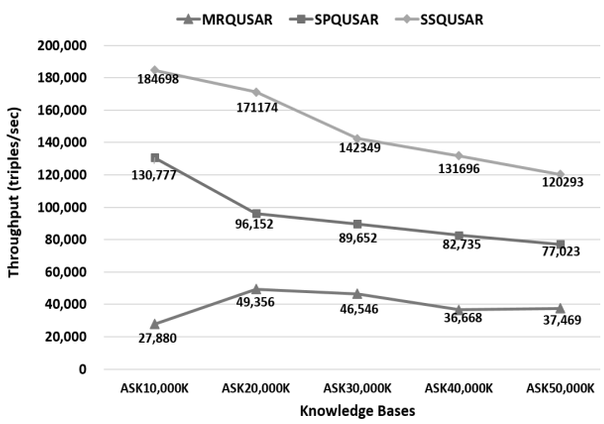


Fig. 21. Comparison of Different Spatial Reasoners in Terms of Throughput

제안한 추론의 효율화 기법들이 공간 추론기의 성능 향상에 긍정적인 효과를 갖는다는 것을 확인할 수 있었다.

네 번째 실험에서는 추론 시간(reasoning time)과 시간당 작업 처리량(throughput) 면에서 MapReduce 기반의 공간 추론기인 MRQUSAR, Apache Spark 기반의 공간 추론기인 SPQUSAR, 그리고 본 연구에서 제안하는 Spark SQL 기반의 공간 추론기인 SSQUSAR 등 총 3 개의 정성 공간 추론기들의 성능을 비교해보았으며, 실험 결과는 각각 Fig. 20, Fig. 21과 같다.

실험 결과를 통해 전체적으로 Spark 기반의 SPQUSAR와 연구에서 제안하는 Spark SQL 기반의 SSQUSAR 가 추론 시간과 시간당 작업 처리량 면에서 모두 MapReduce 기반의 MRQUSAR 보다 추론 성능이 뛰어 났으며, 특히 SSQUSAR가 SPQUSAR보다 최소 2배에서 최대 7배 높은 성능을 보였다. 이 결과를 통하여, 대용량의 고 성능 공간 추론기를 구현하는데, MapReduce나 Spark 보다 Spark SQL을 이용하는 것이 더 유리하다는 것을 알 수 있었다. 하지만 본 연구에서 제안하는 Spark SQL 기반 SSQUSAR는 ASK20,000을 기점으로 데이터의 양이 많아질수록 시간당 작업 처리량도 감소하는 것을 볼 수 있다. 따라서 인-메모리 방식의 프레임워크인 Spark SQL 기반의 공간 추론기의 성능을 최대한 이끌어내기 위해서는 처리하고자 하는 지식의 양에 비례하여 충분한 메모리 확보가 필요 하다는 것도 알 수 있었다.

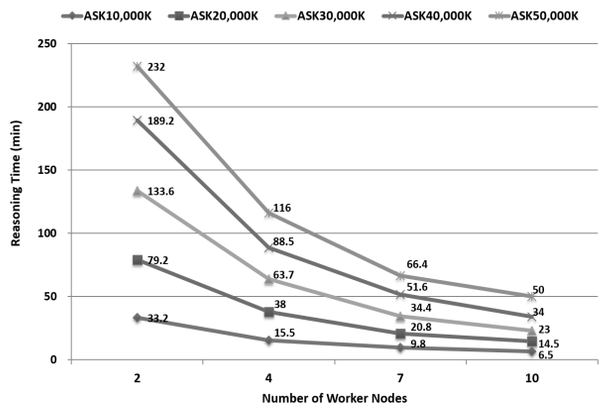


Fig. 22. Analysis of Scalability

다섯 번째 실험에서는 본 연구에서는 제안하는 정성 공간 추론기인 SSQUSAR의 확장성(scalability)을 분석해보았다. 확장성 분석을 위해, 추론에 사용되는 클러스터 컴퓨터 시스템의 작업 노드(worker node)의 수를 증가시켜 가면서 추론 시간의 변화를 관찰해보았다. 이때, 초기 입력 지식 베이스의 크기도 함께 변화시켜 보았으며, 실험 결과는 Fig 22와 같다.

Fig. 22의 실험 결과에서 보듯이 전체적으로 작업 노드의 수가 2 개에서 10 개로 증가함에 따라, 추론 시간은 비례해서 감소하는 결과를 확인할 수 있다. 하지만, 본 실험에서는 작업 노드의 수가 충분히 증가한 이후에는 추론 시간의 감소 폭도 점차 줄어드는 결과를 보였다. 한편, 추론의 대상이 되는 초기 입력 지식 베이스의 크기가 증가할수록 작업 노드의 증가에 따른 추론 시간 감소는 더 뚜렷하게 나타났다. 이것은 대용량의 지식 베이스를 추론할 때 작업 노드의 증가로 더 큰 추론 성능 향상을 얻을 수 있다는 것을 의미한다. 이와 같은 실험 결과를 통해, 본 연구에서 제안한 공간 추론 알고리즘과 공간 추론기가 높은 확장성을 갖는다는 것을 알 수 있다.

마지막 실험에서는 실세계 공간 객체들의 공간 관계 지식을 담고 있는 XB(ExoBrain) 지식 베이스를 이용하여, 본 공간 추론기 SSQUSAR가 올바른 지식들을 추론해낼 수 있는지 정성적 검증 작업을 수행해보았다.

Fig. 23은 XB 공간 지식을 이용한 본 공간 추론기의 추론 결과 중 일부를 보여주고 있다. 정성 공간 추론을 통해 “뮌헨은 독일에 속해 있다.” <Munich within Germany> 라는 지식과 “말리가는 스페인에 속해 있다.” <Malaga within Spain>라는 지식을 신규로 추론해내었는데, 실제 지도상에서 이 지식들이 나타내는 공간 관계들을 확인해본 결과 이들은 모두 참(true)임을 확인할 수 있다. 추론된 지식들에 대한 이와 같은 정성적 검증을 통해, 본 연구의 공간 추론기가 올바른 공간 관계 지식들을 추론해낼 수 있음을 확인할 수 있었다.

Fig. 23. Qualitative Evaluation of Derived Knowledge

6. 결 론

본 논문에서는 Hadoop 클러스터 컴퓨터 시스템 상의 Apache Spark SQL 프레임워크를 사용하여, 공간 객체들 간의 위상 관계를 추론하는 대용량의 정성 공간 추론기의 설계와 구현에 대해 설명하였다. 본 공간 추론기에서는 시맨틱 웹 표준 온토로지언어인 RDF/OWL로 공간 지식을 표현하며, OGC GeoSPARQL 표준에서 정의한 Simple Feature 위상 관계 술어들을 사용하여 공간 객체들 간의 관계를 나타냈다. 효율적인 공간 추론을 위해서는 지식 인코딩을 통해 추론의 대상이 되는 지식 베이스의 크기를 줄이고, 공간 추론에 필요한 최소한의 이접 관계 술어 집합을 찾아서, 이를 기반으로 이행 관계 추론에 필요한 조합표와 관계 정제 기법을 최적화 하였다. 대용량의 가상공간 지식 베이스 ASK들과 실세계 공간 지식 베이스 XB를 이용한 성능 분석 실험을 통해, 본 논문에서 제안한 정성 공간 추론 효율화 기법들의 긍정적인 효과와 공간 추론기 SSQUSAR의 높은 추론 성능을 확인할 수 있었다. 향후에는 본 연구에서 제안한 공간 객체들 간의 2차원 위상 관계뿐만 아니라 3차원 상에서의 방향 관계, 거리 관계 등을 추론할 수 있도록 공간 지식 표현과 추론 알고리즘의 확장에 관해 연구를 계속해나갈 예정이다.

References

[1] J. Renz, "Maximal Tractable Fragments of the Region Connection Calculus: A Complete Analysis," in *Proceedings of IJCAI*, 1999.

[2] D. J. Peuquet and C. X. Zang, "An Algorithm to Determine the Directional Relationship between Arbirtrarily-Shaped Polygons in the Plane," *Pattern Recognition*, Vol.20, No.1, pp.65-74, 1987.

[3] R. Moratz and M. Ragni, "Qualitative Spatial Reasoning about Relative Point Position," *Journal of Visual Languages and Computing*, Vol.19, No.1, pp.75-98, 2008.

[4] B. Gottfried, "Tripartite Line Tracks Qualitative Curvature Information," in *International Conference on Spatial Information Theory. Springer Berlin Heidelberg,* pp.101-117, 2003.

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

[6] S. Batsakis and E. G. Petrakis, "SOWL: A Framework for Handling Spatio-Temporal Information in OWL 2.0," in *International Workshop on Rules and Rule Markup Languages for the Semantic Web. Springer Berlin Heidelberg,* pp.242-249, 2014.

[7] M. Stocker and E. Sirin, "PelletSpatial: A Hybrid RCC-8 and RDF/OWL Reasoning and Query Engine," in *Proceedings of the 6th International Conference on OWL: Experiences and Directions,* Vol.529, pp.39-48, 2009.

[8] G. Christodoulou, "CHOROS: A Reasoning and Query Engine for Qualitative Spatial Information," Dissertion Thesis, Technical University of Crete, Greece, 2011.

[9] S. Nam and I. Kim, "A Qualitative Spatial Reasoner Supporting Cross-Consistency Checks between Directional and Topological Relations," *Journal of KIISE : Computing Practices and Letters,* Vol.20, No.4, pp.248-252, 2014.

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

[11] J. Dean, S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Communications of the ACM,* 2008.

[12] S. Nam and I. Kim, "Design and Implementation of Large-Scale Spatial Reasoner Using MapReduce Framework," *Transactions on KIPS : Software and Data Engineering,* Vol.3, No.10, pp.397-406, 2014.

[13] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, I. Shoica, "Spark: Cluster Computing with Working Sets," in *Proceedings of HotCloud 2010,* Jun., 2010.

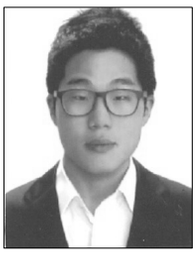
[14] M. Armbrust, et al, "Spark Sql: Relational Data Processing in Spark," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data,* pp. 1383-1394, 2015.

[15] J. Kim and I. Kim, "SPQUSAR: A Large-Scale Qualitative Spatial Reasoner Using Apache Spark," *KIISE Transactions on Computing Practices*, Vol.21, No.12, pp.774-779, 2015.

[16] R. Battle and D. Kolas, "Enabling the Geospatial Semantic Web with Parliament and GeoSPARQL," *Semantic Web Journal,* Vol.3, No.4, pp.355-370, 2012.

[17] P. van Beek and D. W. Manchak, "The Design and Experimental Analysis of Algorithms for Temporal Reasoning," *Journal of Artificial Inteligence Research,* Vol.4, No.1, pp.1-8, 1996.

[18] J. Renz and B. Nebel, "Efficient Methods for Qualitative Spatial Reasoning," in *Proceedings of the 13th European Conference on Artificial Inteligence,* 1998.



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