KBIMCODE LANGUAGE MANUAL

2015.12.11 v0.9

한양대학교 연구팀

건축물 설계품질 혁신을 위한 개방형 BIM기술 환경 구축

한양대학교 연구팀

연구책임자: 이진국

연구원: 이현수, 박서경, 신재영, 김현정, 황금화, 김하얀

목 차

1. 서론

1.1 KBimCode Language란?

건축법규 등에 기술되어있는 자연어 형태의 제규칙·기준 등은 논리규칙 체계화 과정을 통하여 컴퓨터에서 실행 가능한 형태로 변환되어야 한다. 기존의 컴퓨터 프로그래밍 언어를 통한 하드 코딩 방식이나 특정 소프트웨어에 종속적인 룰셋 등의 방식을 탈피하여, 본 과제에서는 개방형 BIM을 지향하는 표준적인 중간언어를 개발하고 이를 KBimCode로 명명 하였다. 모든 대상 건축법규 문장은 KBimCode로 변환되어 데이터베이스에 저장되고, 사용자의 목적에 맞게 재사용 가능하다. KBimCode는 논리규칙 관리 프로그램인 KBimLogic을 통하여 저작 및 관리되며, KBimLogic의 최신 법규 DB와 연동되고, 전문가의 지속적인 관리를 통해 신뢰를 높인다.

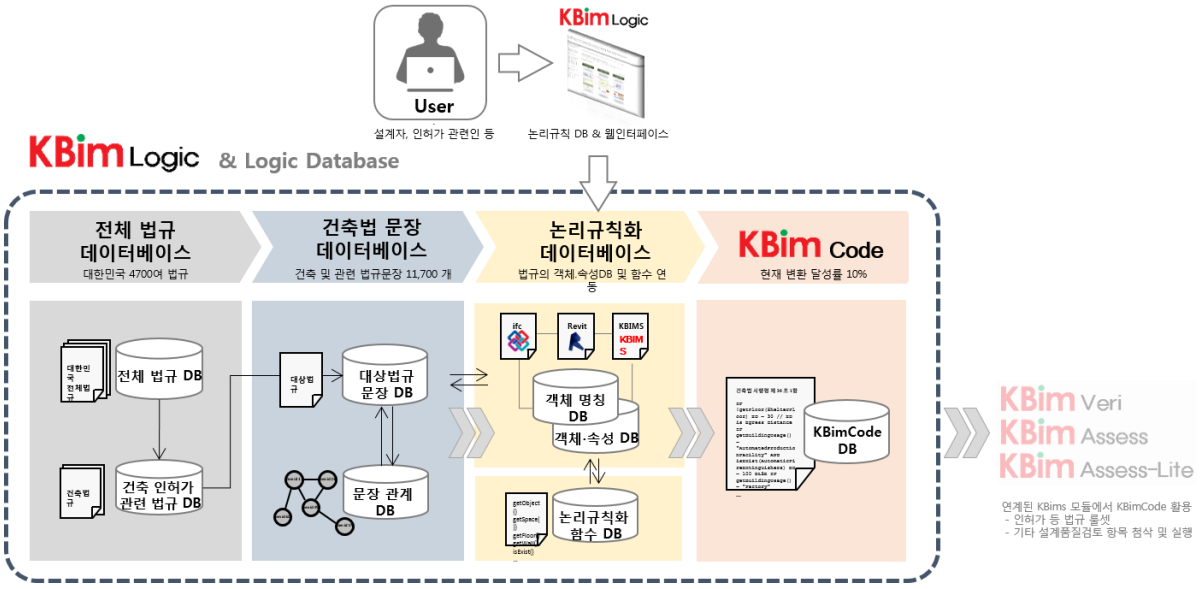


그림 1. KBimCode 개요. 건축법규 문장은 논리규칙 관리 프로그램인 KBimLogic을 통하여 KBimCode로 변환되며, DB로 저장되어 사용자의 목적에 맞게 재사용이 가능하다.

이러한 배경을 바탕으로, KBimCode Language는 다음과 같이 요약될 수 있다.

1) KBimCode Language는 다음과 같이 분류될 수 있다.
   - 도메인 특정 언어 (domain specific language)
   - 프로그래밍 언어 (programming language)
   - 룰 언어 (rule language)

2) KBimCode Language의 주요 기능은 자동화된 설계품질검토를 위하여 자연어로 기술된 룰을 컴퓨터에서 실행 가능한 룰로 변환하는 것이다.

3) KBimCode Language는 고수준(high-level) 언어로서, 프로그래밍에 익숙하지 않은 인허가 관련 인도 룰을 작성할 수 있는 사용자 친화적 특성을 지향한다.

4) KBimCode Language는 대한민국 건축법 중 인허가 관련법을 기초로 개발 되었으며, 추후 다양한 디자인 가이드, RFP 등에 적용 가능하도록 확장될 예정이다.

1.2 목적과 범위

본 문서는 KBimCode의 언어 정의를 설명함에 그 목적이 있다. 이를 통해 다음과 같은 두 가지 목표를 달성하고자 한다.

1) KBimCode 작성 방법을 설명한다.
2) 설계품질검토 소프트웨어와 연동 가능하도록 KBimCode 언어의 디자인 전략을 설명하고 문법을 제공한다.

다음 그림2는 건축법규 자동검토의 프로세스를 간략화하여 보여주는 다이어그램이다. 목표1)은 건축법과 KBimLogic 사이의 프로세스에 관한 것으로서, 자연어 룰을 컴퓨터에서 실행 가능한 룰로 변환하는 작업의 실질적인 규칙을 제공한다. 목표2)는 KBimLogic과 설계품질검토 소프트웨어 사이의 프로세스에 관한 것으로서, KBimCode가 설계품질검토 소프트웨어에서 적용 가능하도록 언어의 문법을 문맥 자유 EBNF (Context-free Extended Backus Naur Form)의 형태로 제공한다.

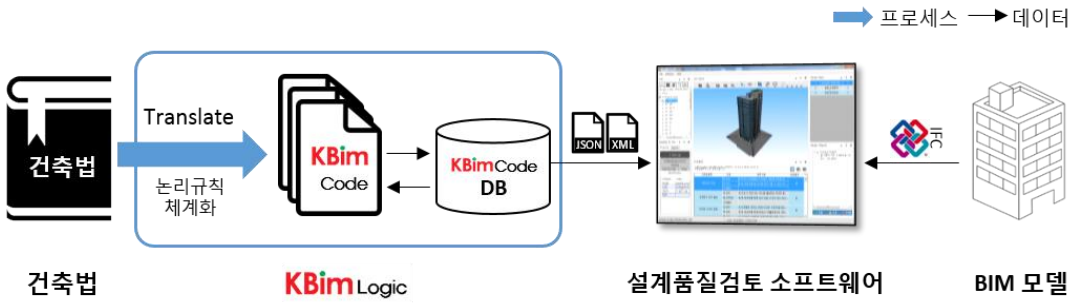


그림2. 건축법규 자동검토 프로세스

본 문서에서 설명하는 KBimCode Language는 대한민국 건축 인허가 관련법 47개(법령 33개, 행정규칙 13개, 자치법규 1개)로부터 추출된113개 조항 1258개 문장에 대한 분석을 기반으로 개발되었다. 대상 법규의 목록은 표1과 같으며, 추출된 조항은 다음의 내용에 대한 검토사항을 다룬다; 1) 건축물의 피난시설 및 용도제한, 2) 지하층, 3) 건축물의 용적률, 4) 승강기, 5) 건축선에 따른 건축제한, 6) 건축물의 내화구조와 방화벽, 7) 방화구획의 설치, 8) 거실 등의 방습, 9) 건축물의 대지가 지역·지구 또는 구역에 걸치는 경우의 조치. 대상 법규문장의 일부를 예시로 들어 KBimCode 언어 정의에 대한 검증을 수행하였으며, 그 내용을 본 문서의 "4.예시"에 설명하였다.

표1. 대상법규 목록

No	법규명	유형
1	건설기술진흥법 시행령 (구 건설기술관리법 시행령)	법령
2	건축물의 구조기준 등에 관한 규칙	법령
3	건축물의 설비기준 등에 관한 규칙	법령
4	건축물의 피난·방화구조 등의 기준에 관한 규칙	법령
5	건축법	법령
6	건축법 시행규칙	법령
7	건축법 시행령	법령
10	국토의 계획 및 이용에 관한 법률	법령
11	국토의 계획 및 이용에 관한 법률 시행령	법령
12	다중이용시설 등의 실내공기질관리법	법령
13	다중이용시설 등의 실내공기질관리법 시행규칙	법령
14	다중이용시설 등의 실내공기질관리법 시행령	법령
15	다중이용업소의 안전관리에 관한 특별법	법령
16	다중이용업소의 안전관리에 관한 특별법 시행령	법령
17	도로법	법령
18	사도법	법령
19	소방시설 설치·유지 및 안전관리에 관한 법률 시행규칙	법령
20	소방시설 설치·유지 및 안전관리에 관한 법률 시행령	법령
21	소방시설 설치・유지 및 안전관리에 관한 법률	법령
22	스프링클러설비의 화재안전기준(NFSC 103)	법령
23	장애인·노인·임산부 등의 편의증진보장에 관한 법률	법령
24	장애인·노인·임산부 등의 편의증진보장에 관한 법률 시행규칙	법령
25	장애인·노인·임산부 등의 편의증진보장에 관한 법률 시행령	법령
26	주차장법	법령
27	주차장법 시행규칙	법령
28	주차장법 시행령	법령
29	주택건설 등에 관한 규정	법령
30	주택법 시행령	법령
32	초고층 및 지하연계 복합건축물 재난관리에 관한 특별법	법령
33	초고층 및 지하연계 복합건축물 재난관리에 관한 특별법 시행령	법령
34	서울특별시 건축조례	자치법규
35	간이스프링클러설비의 화재안전기준(NFSC 103A)	행정규칙
36	건축물의 마감재료의 난연성능 및 화재 확산 방지구조 기준	행정규칙
37	건축물의 에너지 절약설계기준	행정규칙
38	비상경보설비의 화재안전기준(NFSC 201)	행정규칙
39	비상콘센트설비의 화재안전기준(NFSC 504)	행정규칙
40	소화기구 및 자동 소화장치의 화재안전기준(NFSC 101)	행정규칙
41	연결살수설비의 화재안전 기준(NFSC 503)	행정규칙
42	연결송수설비의 화재안전 기준(NFSC 502)	행정규칙
43	연소방지설비의 화재안전기준(NFSC 506)	행정규칙

45	옥내소화전설비의 화재안전기준(NFSC 102)	행정규칙
46	유도등 및 유도표지의 화재안전기준(NFSC 303)	행정규칙
47	특별피난계단의 계단실 및 부속실 제연설비의 화재안전기준(NFSC 501A)	행정규칙

2. KBimCode Language Overview

KBimCode Language 는 기본형으로 check 선언을 가지며, 기본형에 statement group 선언을 추가적으로 사용할 수 있다. Check 선언과 statement group 선언은 1) KBimCode 객체 모델 (KOM), 2) 조건문, 그리고 3)Arithmetic Logic Unit(ALU)으로 이루어진 KBimCode Statement 를 포함한다.

2.1 Check 선언

KBimCode Language는 다음과 같은 기본형으로 구성된다.

```
check (arg) {
   Statement;
   …
}
```

- `Check()`는 특정 법규문장에 대한 KBimCode 작성을 선언하는 컨트롤 함수이다.
- `arg`는 검토 대상 법규문장에 대한 식별자이다.
- `Statement`는 법규문장에서 정의하는 룰이다.

예를 들어, 다음의 법규를 KBimCode로 표현하면 1)과 같다.

[건축법 64조 1항]
"*6층 이상으로서 연면적이 2천제곱미터 이상인 건축물을 건축하려면 승강기를 설치하여야 한다.*"

1)
```
check (BA_64_1) {
              IF (getBuildingStoriesCount() >= 6
                 AND getFloorArea() >= 2000)
                  THEN isExist(Elevator) = TRUE;
}
```

위 예시에서 `check(BA_64_1)`은 해당 KBimCode가 건축법(BA) 제64조 제1항에 대한 것임을 나타낸다. (법규를 표현하는 방법은 부록3을 참고하시오) 중괄호 내부의 statement는 조건문으로, 구체적인 룰의 검토 조건과 내용을 나타낸다. `Statement`의 유형에는 1) KBimCode 객체 모델 (KOM), 2) 조건문, 3) 산술논리단위 (Arithmetic Logic Unit: ALU)가 있다 (각 유형에 대한 상세한 설명은 2.3 KBimCode Statement를 참고하라). 하나 이상의 statement를 묶어 객체로 표현할 수 있으며, 이를 Statement Group이라고 지칭한다.

2.2 Statement Group 선언

Statement Group은 하나 이상의 KBimCode Statement에 대한 집합이다. 기본형은 다음과 같다.

```
var {
   Statement;
   …
}
```

- `Var`은 statement group의 변수명이다.
- `Statement`는 법규문장에서 정의하는 룰이다.

2.1의 예시 1)을 statement Group을 사용하여 표현하면 다음과 같다.

```
2) check (BA_64_1) {
          IF (CS) THEN KS
}
3) CS {
          getBuildingStoriesCount() >= 6
          AND getFloorArea() >= 2000;
}
4) KS {
          isExist(Elevator) = TRUE;
}
```

모든 KBimCode는 check 선언만으로 표현가능 하다. 그러나 statement group 선언을 활용하면 다수의 복잡한 룰을 포함하고 있는 문장을 명료하게 표현할 수 있다는 이점이 있다. Statement Group의 변수명은 해당 KBimCode의 check 선언 및 외부 KBimCode에 사용 될 수 있다.

2.3 KBimCode Statement

2.3.1 KBimCode 객체 모델 (KBimCode Object Model: KOM)

KBimCode 객체모델은 KBimCode객체 모델의 핵심적인 특징 중 하나이다. 이는 법규에 명시된 건축 객체를 사람이 건물을 이해하는 방식으로 추상화 한 것으로, BERA Language의 BERA 객체 모델(BERA Object Model)의 개념을 차용하였다 [Jin-Kook Lee: 2011, Building Emvironment Rule and Analysis (BERA) Language, *Ph.D. Dissertation*, Georgia Institute of Technology]. 사용자는 KOM을 통해다양한 제약조건을 가지는 가상의 객체를 정의할 수 있다. KOM에는 두 가지 종류가 있다. 첫번째는 static KOM으로, 주어진 건물 모델이 메모리에 로드되는 순간 생성되는 데이터 셋이 이에 해당한다. 두번째는 dynamic KOM으로 사용자가 정의한 조건에 의해 static KOM의 일부가 재구성된 것이다. 즉, dynamic KOM은 사용자가 정의한 제약조건(룰)에 따라 동적으로 인스턴스화 된 객체이다. KOM을 사용함으로써, 사용자는 건물 모델에서 원하는 정보를 추출할 수 있다.

KOM의 기본형은 다음과 같다.

```
ObjectType var {
   Statement
   …
}
```

- `ObjectType`은 KOM의 객체 유형이다.
- `var`은 KOM의 변수명이다.
- `Statement` 는 KOM의 제약조건을 나타낸다.

KOM의 예시는 다음과 같다.

[건축법 시행령 35조 (피난계단의 설치) 1항]
*"법 제49조제1항에 따라 **5층 이상 또는 지하 2층 이하인 층**에 설치하는 직통계단은 국토교통부령으로 정하는 기준에 따라 피난계단 또는 특별피난계단으로 설치하여야 한다..."*

```
5) Floor myFloor {
                myFloor.number >5
                OR myFloor.number <=-2;
                }
```

2.3.2 조건문

대한민국 건축법규 문장은 1)검토의 조건을 나타내는 부분과 2) 검토의 내용을 나타내는 부분으로 구성된다. 조건문은 검토의 조건을 나타내는 서술문으로, IF-THEN-ELSEIF-ELSE 논리와 (ELSEIF와 ELSE는 선택적이다) 산술논리단위를 포함한다. 조건문의 예시는 다음과 같다.

```
6) IF(BuildingStoriesCount()>=6) THEN isExist(Elevator)=TRUE
```

```
7) IF (CS) THEN KS
```

2.3.3 산술논리단위(ALU)

산술논리단위는 룰 체킹의 최소단위에 해당하며 참 또는 거짓의 결과를 반환한다. 산술논리단위의 기본 구조는 다음과 같다.

```
operand operator operand
```

산술논리단위의 참, 거짓 판별은 좌항과 우항을 비교함으로써 가능하다. 좌항은 사전 정의된 논리 규칙화 함수 및 컨트롤 함수(부록 2를 참고하시오) 및 KOM의 속성을 나타내기 위한 dot notation 표기 등이 사용될 수 있으며, 우항은 numeric, string, boolean 등의 명료한 값이 사용된다.

연산자는 데이터 유형에 종속적이며 다음과 같다.

- Operators for string: =,!=, ==
- Operators for numeric: >, >=, =, <, <=,!=

산술논리단위는 AND, OR 등의 접속사와 함께 사용되어 다수의 제약조건을 표현할 수 있으며, 조건문에도 사용된다.

산술논리단위의 예시는 다음과 같다.

```
1) getSpaceDistance(LivingRoom, Stairs, MRP) <= EgressDistance;
2) Rail.Space.type = Balcony AND Rail.Floor.number >= 2;
3) getResult(BA.49.1) = TURE;
```

3. 예시

본 장은 인허가 관련 법규를 예시로 작성한 KBimCode를 바탕으로 KBimCode Language의 syntax를 검증한다. 예시의 구성은 다음과 같다.

예시 1: Check 선언

법규 원문

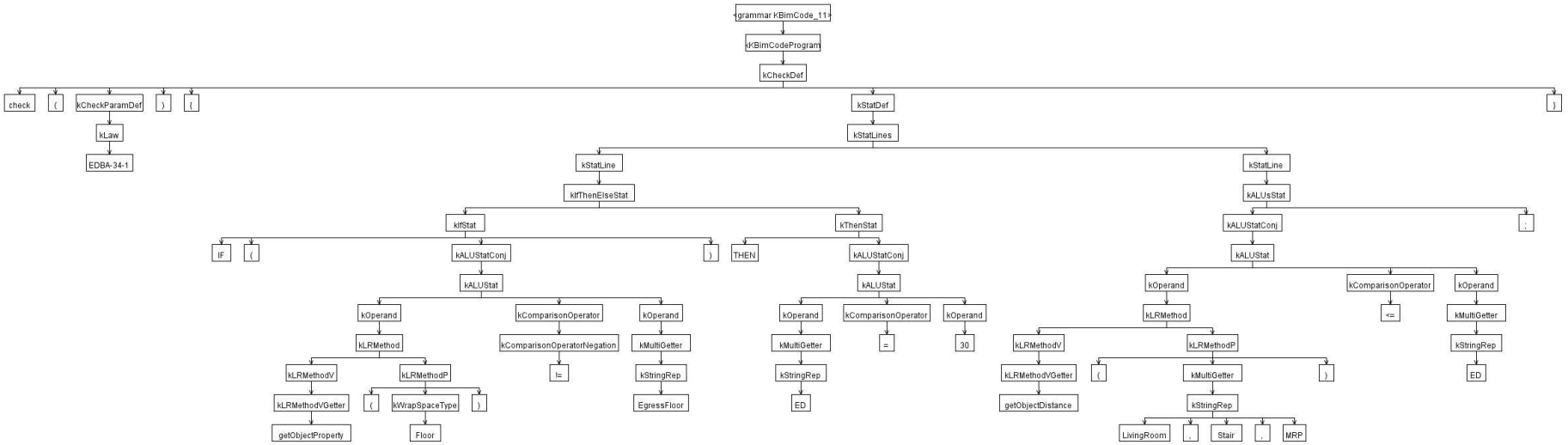
[건축법 시행령 34 조 (직통계단의 설치) 1 항]

"건축물의 피난층(직접 지상으로 통하는 출입구가 있는 층 및 제 3 항과 제 4 항에 따른 피난안전구역을 말한다. 이하 같다) 외의 층에서는 피난층 또는 지상으로 통하는 직통계단(경사로를 포함한다. 이하 같다)을 거실의 각 부분으로부터 계단(거실로부터 가장 가까운 거리에 있는 계단을 말한다)에 이르는 보행거리가 30 미터 이하가 되도록 설치하여야 한다..."

KBimCode

```
check(EDBA_34_1) {
        IF (getObjectProperty(Floor) != EgressFloor)
                THEN ED = 30
        getObjectDistance(LivingRoom, Stair, MRP) <= ED;
}
```

파싱트리



예시 2: Statement Group 선언

법규 원문

[건축법 64 조 (승강기) 1 항]

"건축주는 6 층 이상으로서 연면적이 2 천제곱미터 이상인 건축물(대통령령으로 정하는 건축물은 제외한다)을 건축하려면 승강기를 설치하여야 한다. 이 경우 승강기의 규모 및 구조는 국토교통부령으로 정한다."

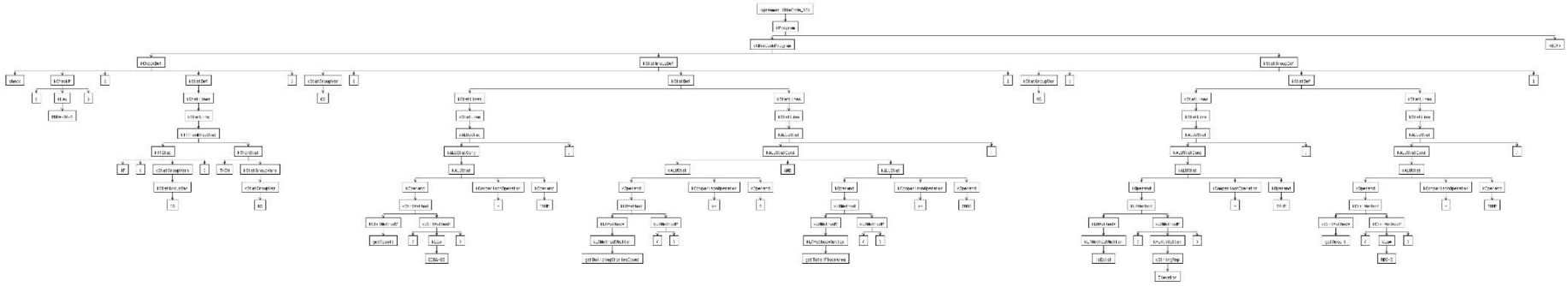
KBimCode

```
check(EDBA_34_1) {
        IF (CS) THEN KS
}

CS {
        getResult(EDBA-89) = TRUE;
        getBuildingStoriesCount() >= 6 AND getTotalFloorArea() >=2000;
}

KS {
        isExist(Elevator) = TRUE;
        getResult(RBS_5) = TRUE;
}
```

파싱트리



예시 4: 복수 문장에 대한 KBimCode

법규 원문

[건축법 시행령 35 조 (피난계단의 설치) 1 항, 1 호, 2 호]

"법 제 49 조제 1 항에 따라 5 층 이상 또는 지하 2 층 이하인 층에 설치하는 직통계단은 국토교통부령으로 정하는 기준에 따라 피난계단 또는 특별피난계단으로 설치하여야 한다. 다만, 건축물의 주요구조부가 내화구조 또는 불연재료로 되어 있는 경우로서 다음 각 호의 어느 하나에 해당하는 경우에는 그러하지 아니하다.

1. 5 층 이상인 층의 바닥면적의 합계가 200 제곱미터 이하인 경우
2. 5 층 이상인 층의 바닥면적 200 제곱미터 이내마다 방화구획이 되어 있는 경우."

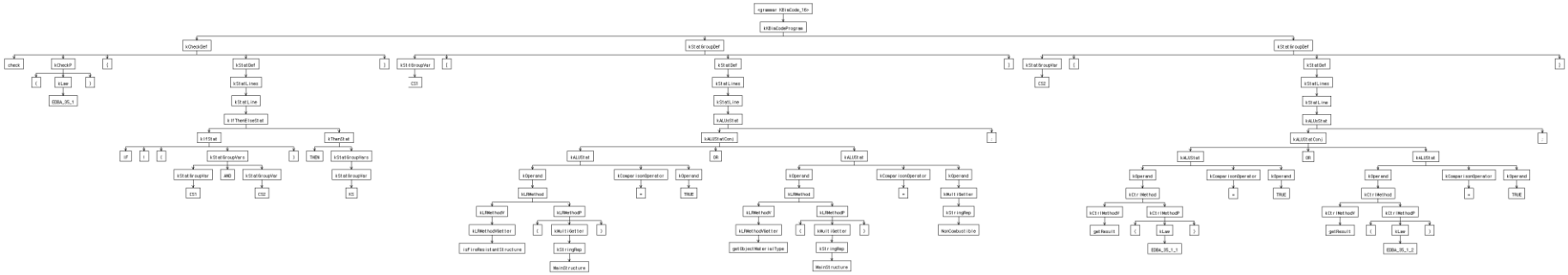
KBimCode

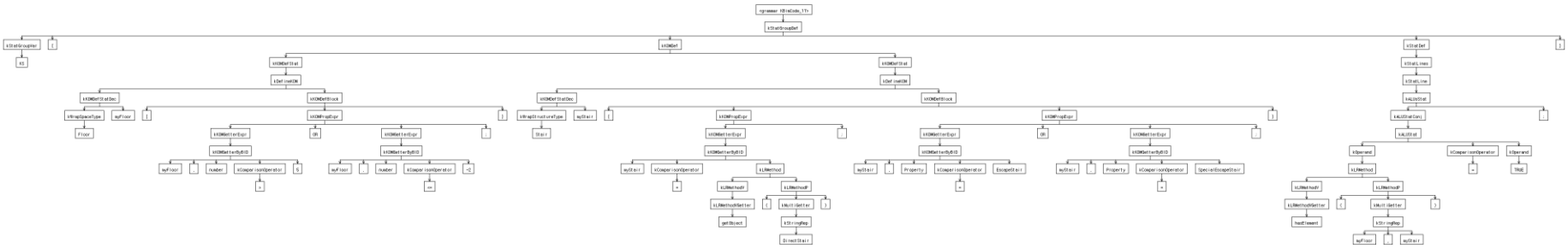
```
//건축법 시행령 35 조 1 항
check(EDBA_35_1) {
        IF !(CS1 AND CS2) THEN KS
}
CS1 {
        isFireResistantStructure(MainStructure) = TRUE
        OR getObjectMaterialType(MainStructure) = NonCombustible;
}
CS2 {
        getResult (EDBA_35_1_1) = TRUE
        OR getResult(EDBA_35_1_2) = TRUE;
}
KS {
        Floor myFloor {
                myFloor.number >5
                OR myFloor.number <=-2;
                }
        Stair myStair {
                myStair = getObject(DirectStair);
                myStair.Property = EscapeStair
                OR myStair.Property = SpecialEscapeStair;
                }
hasElement(myFloor , myStair) = TRUE;
}
//건축법 시행령 35 조 1 항 1 호
check(EDBA_35_1_1){
        Floor myFloor {
                myFloor.number >= 5;
        }
        getTotalFloorArea (myFloor) <= 200;
```

```
}
//건축법 시행령 35조 1항 2호
check(EDBA_35_1_2){
        Floor myFloor {
                myFloor.number >=5;
        }
        isGrouped(myFloor, FireZone, 200) = TRUE;
}
```

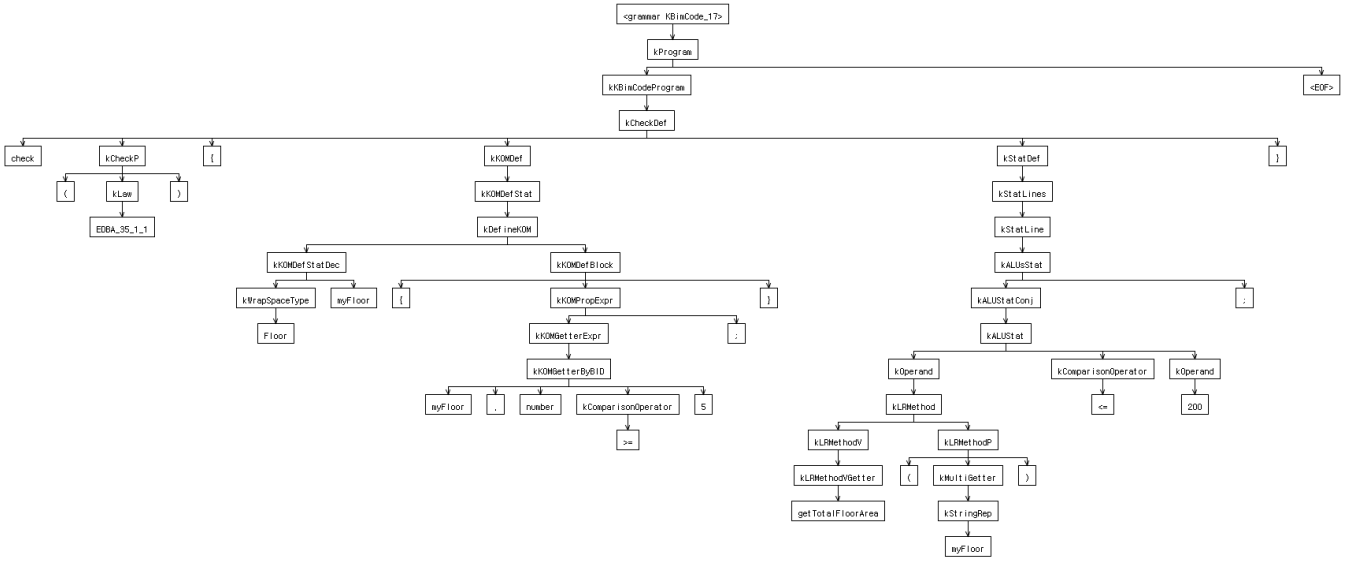
파싱트리

//건축법 시행령 35조 1항

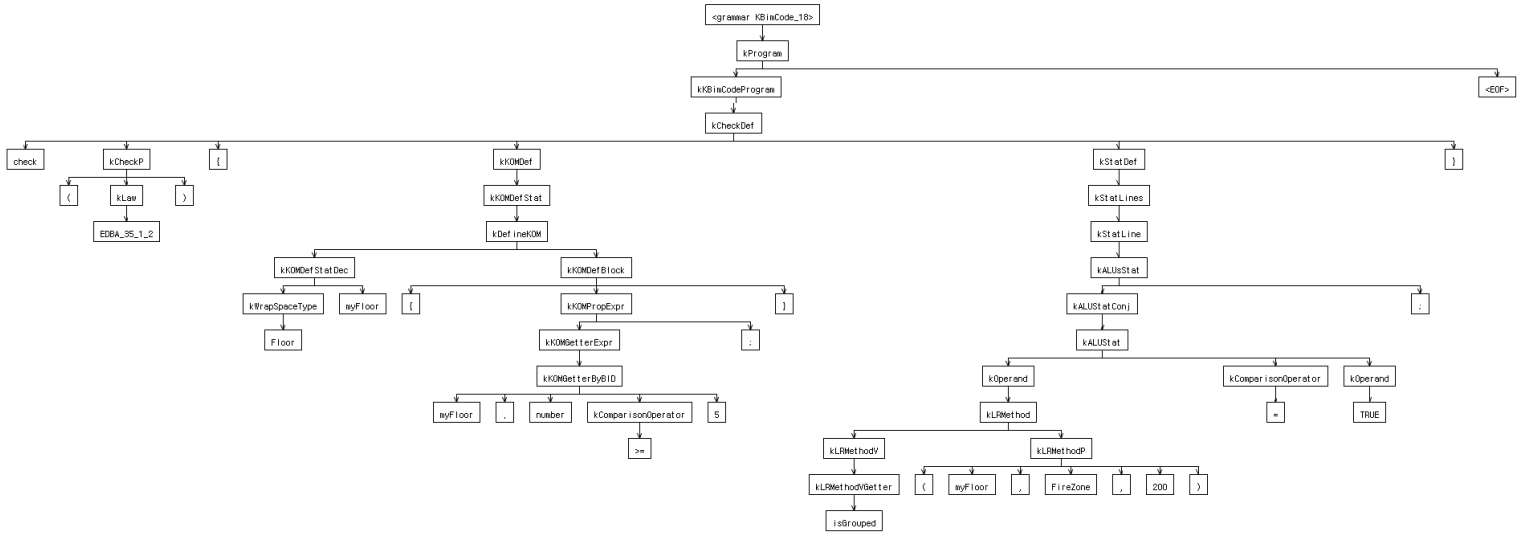




//건축법 시행령 35조 1항 1호



//건축법 시행령 35조 1항 1호



4. 부록

부록1: KBimCode 객체 모델 속성 목록

KBimCode 객체 모델과 해당 속성은 BERA 객체 모델을 확장 및 수정하여 대한민국 인허가 관련 건축법에 적용 가능하도록 개발되었다. 본 문서에서 설명하는 KOM 속성은 전체 인허가 관련법에 명기된 검토 대상 객체의 일부이며, 추후 연구를 통해 확장·보완될 것이다.

표n. 건물 객체 속성과 설명

속성명	설명
Building.id	1부터 시작하는 KBimCode 객체 식별 숫자. 마지막 id는 이 KOM 요소 수의 총합과 항상 같다. *대부분의 상황에서 건물은 하나이나 KOM은 모든 다른 객체들에 따라 하나, 혹은 다수로 정의한다.
Building.fileName	건물 모델 파일의 이름
Building.count	객체의 수. 예) 공간의 수, 층의 수, 문의 수, 벽의 수, 계단의 수 등
Building.area	건축면적. 건축물의 외벽의 중심선으로 둘러싸인 부분의 수평투영면적으로 한다. 세부 산정기준은 건축법 시행령 119조 2항을 따른다.
Building.totalFloorArea	건축물의 연면적. 하나의 건축물 각 층의 바닥면적의 합계이다. 세부 산정기준은 건축법 시행령 119조 1항 4호를 따른다.
Building.floorAreaRatio	건축물의 용적률. 세부 산정기준은 건축법 시행령 119조 1항4호를 따른다.
Building.height	건축물의 높이. 세부 산정기준은 건축법 시행령 119조 1항 5호를 따른다.
Building.eavesheight	처마높이. 세부 산정기준은 건축법 시행령 119조 1항 6호를 따른다.
Building.elevationHeight	지하층을 포함한 전체 건축물의 높이이다.
Building.volume	건물 부피의 총합
Building.Site	대지는 다수의 건물을 존재할 수 있다. (프로젝트와 비슷함. 향후 버전에서 더 정교화 될 예정)
Building.Floor	층 객체들은 건물에 포함된다.
Building.firstFloor	건물의 1층 객체. 층 고도의 높이에 의해 산출된다.
Building.storiesCount	건축물의 전체 층수. 세부 산정기준은 건축법 시행령 119조 1항 9호를 따른다.
Building.buildingType	건물의 유형. 예) "사무용 건물"
Buildng.Usage	건축물의 용도. 건축법 시행령 별표1에서 정의하는 용도별 건축물의 종류를 따른다.
Building.designPhase	객체의 수, 관계 등에 의해 추론되는 설계의 단계. 예) "컨셉 디자인"
Building.bimDesignTool	건물 IFC 모델을 내보내는 BIM 저작 도구의 이름.

	예) "Revit Architecture 2011"
Building.version	건물 모형 혹은 스키마의 버전. 예)"IFC 2X3"
Building.structuredParkingArea	건물 안의 주차공간 면적의 합.
Builidng.mepArea	건물 내부의 기계, 전기, 배관 공간의 합.
Building.externallWallArea	모든 외벽 면적의 총합.
Building.biggestFloorArea	건물의 가장 넓은 층 면적.

표n. 층 객체의 속성과 설명

속성명	**설명**
Floor.id	1부터 시작하는 KBimCode 객체 식별 숫자. 마지막 id는 이 KOM 요소 수의 총합과 항상 같다.
Floor.GUID	IFC로부터 나오는 전역적으로 고유한 ID 문자열.
Floor.name	층의 이름. 예)"Level 1"
Floor.area	바닥면적. 건축물의 각 층 또는 그 일부로서 벽, 기둥, 그 밖에 이와 비슷한 구획의 중심선으로 둘러싸인 부분의 수평투영면적으로 한다. 세부 산정기준은 건축법 시행령 119조 1항 3호를 따른다.
Floor.totalNetArea	층 안 공간의 실사용 면적의 합.
Floor.height	층고. 세부 산정기준은 건축법 시행령 119조 1항 8호를 따른다.
Floor.CeilingHeight	반자높이. 세부 산정기준은 건축법 시행령 119조 1항 7호를 따른다.
Floor.elevationHeight	지표면으로부터 해당 층의 바닥구조체 윗면까지의 높이이다. 건축법 시행령 119조
Floor.volume	층의 부피.
Floor.Building	층을 포함하는 건물 객체.
Floor.Space	층 안에 포함되는 공간 객체들.
Floor.number	층에 배정된 계산된 숫자. 지상 층 면적은 항상 1, 2, ...n 그리고 지하층은 항상 -1, -2, ...-m. 이다. 예)방이 두 번째 층에 있을 경우 숫자는 2이다. 만약 지하 3층이라면 숫자는 -3이 된다.

표n. 공간 객체의 속성과 설명

속성명	**설명**
Space.id	1부터 시작하는 BERA 객체 식별 숫자. 마지막 id는 이 KOM 요소 수의 총합과 항상 같다.
Space.GUID	IFC로부터 나오는 세계적으로 독특한 ID 문자열.
Space.name	공간의 이름. 예)"사무실"
Space.area	공간의 실사용 면적.
Space.height	공간의 높이(천장).
Space.volume	공간의 부피.

Space.numberofDoor	공간 객체에 속한 문의 개수.
Space.Building	공간을 포함하는 건물 객체.
Space.Floor	공간을 포함하는 층 객체.
Space.adjacentSpace	공간에 인접한 공간 객체들.
Space.directlyAccessibleSpace	공간에 직접적으로 접근 가능한 공간 객체들.
Space.basicRsf	공간의 기본 임대 면적 (RSF). ANSI/BOMA 표준 참고. [ANSI/BOMA, 2010] ANSI/BOMA 특별 속성.

표n. 공간 그룹 객체의 속성과 설명

속성명	**설명**
SpaceGroup.id	1부터 시작하는 BERA 객체 식별 숫자. 마지막 id는 이 BOM 요소 수의 총합과 항상 같다.
SpaceGroup.name	공간 집합의 사용자 정의 이름.. 예) "mySpaces", "CirculationSpaces"
SpaceGroup.type	KOM 내부 요소의 유형. 이번 버전의 실행에서는 오직 한가지의 유형만 가능하다. 예) "공간", "층", "통로"
SpaceGroup.Space	집합에 포함된 공간 객체들.
SpaceGroup.Floor	집합에 포함된 층 객체들.
SpaceGroup.numberOfSpace	집합 내 공간 객체들의 수.
SpaceGroup.numberOfFloor	집합 내 층 객체들의 수.
SpaceGroup.area	공간 그룹 면적의 총합. 모든 공간의 기본 Rsf. 의 합.
SpaceGroup.height	집합 내 모든 공간들의 평균 높이.
SpaceGroup.volume	공간 집합의 부피.

표n. 동선 객체의 속성과 설명

속성명	**설명**
Path.id	1부터 시작하는 BERA 객체 식별 숫자. 마지막 id는 이 KOM 요소 수의 총합과 항상 같다.
Path.name	공간 집합의 사용자 정의 이름. 예) "myCirculationPath", "fireExitPaths"
Path.start	순환 동선을 위한 시작 공간의 사용자정의 이름.
Path.end	순환 동선을 위한 종료 공간의 사용자정의 이름.
Path.startSpace	건물 내 실제로 감지된 시작 공간 객체.
Path.Space	해당 동선이 지나는 공간 집합.
Path.distance	예시 동선의 거리. 알고리즘 설명: [Lee J-K, 2010]

부록2: 논리규칙화 함수 및 컨트롤 함수 목록

논리규칙화 함수는 법규 문장의 서술부를 표현하며, 다양한 룰의 핵심적인 검토 내용을 담고있다. 논리규칙화 함수는 법규 문장으로부터의 객체와 그 속성을 파라미터로 가진다. 본 문서의 범위에서 추출된 논리규칙화 함수의 목록은 다음과 표n과 같다. 함수에 대한 자세한 사양은 "논리규칙화 함수 매뉴얼"에 설명되어있으며, 해당 문서는 http://designitlab.kr/bim/kbim/method/classification.asp를 통해 접근 가능하다.

표n. 논리규칙화 함수 목록

No	함수명	설명
1	`getObject(obj)`	건물에서 구하고자 하는 객체를 쿼리하는 함수이다
2	`getSpace()`	getObject()의 확장함수이다. 공간객체를 쿼리하는 함수이다.
3	`getFloor()`	getObject()의 확장함수이다. 층을 쿼리하는 함수이다.
4	`getWall()`	getObject()의 확장함수이다. 벽 객체를 쿼리하는 함수이다.
5	`getWindow()`	getObject()의 확장함수이다. 창문 객체를 쿼리하는 함수이다.
6	`getStair()`	getObject()의 확장함수이다. 계단 객체를 쿼리하는 함수이다.
7	`isExist(obj)`	객체의 모델링 유무를 확인하는 함수이다
8	`getObjectCount(obj)`	객체의 개수를 구하는 함수이다
9	`getBuildingStoriesCount()`	getObjectCount()의 확장함수로, 건물에 존재하는 층의 개수를 구하는 함수이다.
10	`getElevatorCount()`	getObjectCount()의 확장함수로, 건물에 존재하는 엘레베이터의 개수를 구하는 함수이다.
11	`getStairCount()`	getObjectCount()의 확장함수로, 건물에 존재하는 계단의 개수를 구하는 함수이다.
12	`getObjectProperty(obj)`	대상 객체의 특정한 속성을 확인하는 함수이다
13	`getDoorType()`	getObjectProperty()의 확장함수이다. 문 객체의 유형을 구하는 함수이다.
14	`getWindowType()`	getObjectProperty()의 확장함수이다. 창문 객체의 유형을 구하는 함수이다.
15	`getFloorNumber()`	getObjectProperty()의 확장함수이다. 해당 층의 층수를 확인하는 함수이다.
16	`getElevLiveLoad()`	getProperty()의 확장함수이다. 엘레베이터의 적재하중을 구하는 함수이다.
17	`getObjectMaterial(obj)`	대상 객체의 재료명을 확인하는 함수이다
18	`getObjectUsage(obj)`	대상 객체의 용도를 확인하는 함수이다
19	`getBuildingUsage()`	getObjectUsage()의 확장함수이다. 건물의 용도를 구하는 함수이다.
20	`getFloorUsage()`	getObjectUsage()의 확장함수이다. 층의 용도를 구하는 함수이다.
21	`getSpaceUsage()`	getObjectUsage()의 확장함수이다. 공간의 용도를 구하는 함수이다.
22	`getSiteUsage()`	getObjectUsage()의 확장함수이다. 대지의 용도를 구하는 함수이다.

23	getObjectHeight(obj, type)	대상객체의 높이를 구하는 함수이다
24	getBuildingHeight()	건물의 높이를 구하는 함수이다
25	getBuildingElevationHeight()	지하층을 포함한 건축물의 전체 높이를 구하는 함수이다.
26	getSpaceHeight()	대상 공간객체의 높이를 구하는 함수이다
27	getCeilingHeight()	반자높이를 구하는 함수이다.
28	getFloorHeight()	층고를 구하는 함수이다.
29	getFloorElevationHeight()	지표면으로부터 해당 층의 바닥구조체 윗면까지의 높이를 구하는 함수이다.
30	getElementHeight()	대상 건축객체의 높이를 구하는 함수이다
31	getDoorHeight()	문의 높이를 구하는 함수이다.
32	getStairStepHeight()	계단 한단의 높이를 구하는 함수이다.
33	getObjectWidth(obj, type)	대상객체의 길이를 구하는 함수이다
34	getSpaceWidth()	대상 공간객체의 폭, 너비, 길이 등을 구하는 함수이다
35	getCorridorWidth()	복도의 폭을 구하는 함수이다.
36	getStairWidth()	계단실의 폭을 구하는 함수이다.
37	getElementWidth()	대상 건축객체의 길이를 구하는 함수이다
38	getDoorWidth()	문의 폭을 구하는 함수이다.
39	getWallWidth()	벽의 두께를 구하는 함수이다.
40	getStairStepWidth()	계단 한 단의 폭을 구하는 함수이다.
41	getObjectArea(obj, type)	대상 객체의 면적을 구하는 함수이다
42	getFloorArea()	대상 공간객체의 바닥면적을 구하는 함수이다
43	getLivingRoomArea()	거실의 용도로 사용하는 공간의 면적을 구하는 함수이다.
44	getUndergroundFloorArea() getBasementFloorArea()	지하층의 바닥면적을 구하는 함수이다.
45	getTotalFloorArea() getGrossFloorArea()	대상 건축물의 용적률을 구하는 함수이다
46	getFloorAreaRatio()	대상 건축물의 건폐율을 구하는 함수이다
47	getbuildingArea()	대상 건축물의 건축면적을 구하는 함수이다.
48	getSiteArea()	대상 건축물의 대지면적을 구하는 함수이다.
49	getElementArea()	대상 건축객체의 면적과 관련된 값을 구하는 함수이다
50	getWallArea()	대상 벽 객체의 면적을 구하는 함수이다
51	getWindowArea()	대상 창문 객체의 면적을 구하는 함수이다
52	getObjectSectionalArea(obj, Type)	객체의 단면적을 구하는 함수이다.
53	getObjectDiameter(obj, type)	객체의 지름을 구하는 함수이다.
54	getObjectGradient(obj)	대상객체의 경사도를 구하는 함수이다
55	getObjectMaterialType(obj)	대상 객체의 재료타입을 구하는 함수이다
56	isGrouped(obj, type1, type2)	객체가 법정 기준 또는 다른 객체와 구획되어있는 여부를 나타내는 함수이다.
57	isGroupedFirePartition()	객체가 방화구획으로 구획되어있는 상태를 확인하는 함수이다.
58	isGroupedFireWall()	객체가 방화벽으로 구획되어있는 상태를 확인하는 함수이다.

59	isFireResistantStructure(obj)	객체가 내화구조임을 확인하는 함수이다. 대상 부재가 법규에서 명시하는 특정 조건을 만족하는 (건축물의 피난・방화구조 등의 기준에 관한 규칙 제3조의 기준에 따라 내화성능을 인정받은) 내화구조인지 확인하는 함수이다
60	isFireProofStructure(obj)	객체가 방화구조임을 확인하는 함수이다. 대상 부재가 국토교통부령(건축물의 피난・방화구조 등의 기준에 관한 규칙 제4조)의 기준에 따라 방화성능을 인정받은 방화구조인지 확인하는 함수이다
61	isFirePartition(obj)	객체가 방화구획임을 확인하는 함수이다. 대상 공간객체가 방화구획으로 구획 되어있는지 확인하는 함수이다. 방화구획은 발화점에서 옆방이나 위층으로 불이 옮겨가지 않도록 화재에 견딜 수 있는 내화구조로 된 바닥, 벽 및 갑종방화문이나 자동방화셔터 등으로 구획하는 것을 말한다
62	isLoadBearing(obj)	하중을 받는 객체인지 확인하는 함수이다.
63	getObjectFoundation(obj)	재료관련 구조를 확인하는 함수이다.
64	getObjectStructure(obj)	대상 객체의 건축구조(예시.돌구조,벽돌구조, 철근콘크리트구조 등)를 확인하는 함수이다.
65	hasObject(obj1, obj2)	obj1이 obj2를 포함하고 있는지 판단하는 함수이다.
66	hasSpace(space1, obj1)	특정 공간객체(space1)가 obj1을 포함하고 있는지 판단하는 함수이다
67	hasElement(element1, element2)	공간객체를 제와한 객체(element1)가 (공간객체를 제외한)다른 객체(element2)를 포함하는지 판단하는 함수이다.
68	getObjectDistance(obj1, obj2, type)	두 객체 사이의 거리를 구하는 함수이다
69	getObjectVerticalDistance(obj1, obj 2, type)	두 객체 사이의 수직거리를 구하는 함수이다
70	isConnectedTo(obj1, obj2)	두 객체의 연결 여부를 확인하는 함수이다
71	isExternal(obj)	객체가 건물 외부와 면하고 있는지 판단하는 함수이다
72	isAdjacent(obj1, obj2)	두 공간객체가 인접하고 있는지 판단하는 함수이다
73	isSurrounded(obj1, obj2)	특정 객체(obj1)가 다른 객체(obj2)에 둘러쌓여있음을 확인하는 함수이다.
74	isAccessible(obj1, obj2)	두 공간객체의 연결여부 또는 이동가능 여부를 판단하기위한 함수이다.
75	isDirectlyAccessible(obj1, obj2)	두 공간객체가 다른 공간객체를 사이에 두지 않고 직접 연결 또는 이동가능 여부를 판단하기 위한 함수이다.
76	isGoThrough(obj1, obj2, obj3)	두 공간객체간 이동시 특정 공간을 지나야만 하는지 판단하는 함수이다
77	getObjectDirection(obj)	객체의 방향을 확인하는 함수이다.
78	getDooSwingDirection()	문 객체가 열리는 방향을 확인하는 함수이다
79	isEgressDirection(obj)	문이 열리는 방향과 피난방향이 일치하는지 확인하는 함수이다

논리규칙화 함수가 룰을 실질적인 검토 내용을 표현하기 위해 사용되는 반면 컨트롤 함수는 KBimCode의 문법적인 측면에 관련되어있다. 컨트롤 함수의 종류와 설명은 다음 표n과 같다.

표n. 컨트롤 함수 목록

No	함수명	설명
1	`check(targetLaw)`	targetLaw에 대한 KBimCode를 작성함을 선언하는 함수이다.
2	`getResult(targetLaw)`	[Boolean] targetLaw의 검토결과를 쿼리하는 함수이다. 특정 법규문장에서 다른 법규문장의 검토결과를 가져올 때 사용된다.
3	`setResult(targetLaw)`	[Boolean] 조건에 따라 targetLaw의 검토결과를 결정하는 함수이다.

부록3: 법규 문장 식별자 표기방법

2.1 check 선언에서 필요로하는 법규 문장에 대한 식별자 표기방법은 다음과 같다.

법규 식별자 '_' 조 ('_' 항) ('_' 호) ('_' 목)

- 법규 식별자, 조, 항, 호, 목은 법규의 단위를 나타내는 비종단 기호이다.
- ()안의 항목은 문장에 따라 선택적이다.
- 법규 식별자는 본 문서에서 정한 표기법을 따른다.
- 조 이하 단위 (조, 항, 호, 목)는 숫자 또는 숫자와 기호의 조합으로 표현하며, 해당 단위명과 함께 쓰지 않는다.
- 각 단위는 종단기호 _ 으로 연결한다.

법규 문장 식별자 표기방법에 대한 예시는 다음과 같다.

<올바른 표기법>

1) 건축법 제64조 제1항: BA_64_1

2) 건축물의 피난·방화구조 등의 기준에 관한 규칙 제8조의2 제3호: REF_8_2-3

본 문서의 범위에 해당하는 인허가 관련 건축법규에 대한 식별자 목록은 다음과 같다.

법규명	식별자
간이스프링클러설비의 화재안전기준 (NFSC 103A)	NFSC103A
건설기술진흥법 시행령 (Construction Technology Promotion Act)	CTPA
건축물의 구조기준 등에 관한 규칙 (Regulation for Structure Standard in Building)	RSSB
건축물의 마감재료의 난연성능 및 화재 확산 방지구조 기준 (Standard for Combustibility and Fire Protecting Performance of Finishing Material)	SCFFM
건축물의 설비기준 등에 관한 규칙 (Regulation for Facility in Building)	RFB
건축물의 에너지 절약설계기준 (Energy Saving Designing Standard)	ESDS
건축물의 피난·방화구조 등의 기준에 관한 규칙 (Regulation for Evacuation and Fireproof Construction in Building)	REFB
건축법 (Building Act)	BA
건축법 시행규칙 (Enforcement Regulation of Building Act)	ERBA
건축법 시행령 (Enforcement Decree of Building Act)	EDBA
국토의 계획 및 이용에 관한 법률	LPUA

(National Land Planning and Utilization Act)	
국토의 계획 및 이용에 관한 법률 시행령 (Enforcement Decree of the National Land Planning and Utilization Act)	EDLPUA
다중이용시설 등의 실내공기질관리법 (Indoor Air Quality Control in Public-Use Facilities, ETC. Act)	IAQA
다중이용시설 등의 실내공기질관리법 시행규칙 (Enforcement Regulation of Indoor Air Quality Control in Public-Use Facilities, ETC. Act)	ERIAQA
다중이용시설 등의 실내공기질관리법 시행령 (Enforcement Decree of Indoor Air Quality Control in Public- Use Facilities, ETC. Act)	EDIAQA
다중이용업소의 안전관리에 관한 특별법 (Special Act on the Safety Control of Publicly Used Establishments)	SASP
다중이용업소의 안전관리에 관한 특별법 시행령 (Enforcement Regulation of Special Act on the Safety Control of Publicly Used Establishments)	ERSASP
도로법 (Road Act)	RA
비상경보설비의 화재안전기준(NFSC 201)	NFSC201
비상콘센트설비의 화재안전기준(NFSC 504)	NFSC504
사도법 (Private Road Act)	PRA
서울특별시 건축조례 (Seoul Metropolitan Building Act)	SMBA
소방시설 설치·유지 및 안전관리에 관한 법률 시행규칙 (Enforcement Regulation of Installation, Maintenance, and Safety Control of Fire-Fighting System Act)	ERIMSFA
소방시설 설치·유지 및 안전관리에 관한 법률 시행령 (Enforcement Decree of Installation, Maintenance, and Safety Control of Fire-Fighting System Act)	EDIMSFA
소방시설 설치・유지 및 안전관리에 관한 법률 (Installation, Maintenance, and Safety Control of Fire-Fighting System Act)	IMSFA
소화기구 및 자동 소화장치의 화재안전기준(NFSC 101)	NFSC101
스프링클러설비의 화재안전기준(NFSC 103)	NFCS103
연결살수설비의 화재안전 기준(NFSC 503)	NFSC503
연결송수설비의 화재안전 기준(NFSC 502)	NFSC502
연소방지설비의 화재안전기준(NFSC 506)	NFSC506
옥내소화전설비의 화재안전기준(NFSC 102)	NFSC102
유도등 및 유도표지의 화재안전기준(NFSC 303)	NFSC303
장애인·노인·임산부 등의 편의증진보장에 관한 법률 (Act on Guarantee of Promotion of Convenience of Persons with Disabilities, the Aged, pregnant women, Etc.)	CDAPA
장애인·노인·임산부 등의 편의증진보장에 관한 법률 시행규칙 (Enforcement Regulation of Act on Guarantee of Promotion of Convenience of Persons with Disabilities, the Aged, pregnant women, Etc.)	ERCDAPA

장애인·노인·임산부 등의 편의증진보장에 관한 법률 시행령 (Enforcement Decree of Act on Gurantee of Promotion of Convenience of Persons with Disabilities, the Aged, pregnant women, Etc.)	EDCDAPA
주차장법 (Parking Lot Act)	PLA
주차장법 시행규칙 (Enforcement Regulation of Parking Lot Act)	ERPA
주차장법 시행령 (Enforcement Decree of Parking Lot Act)	EDPA
주택건설 등에 관한 규정 (Regulation on Housing Construction)	RHC
주택법 시행령 (Enforcement Decree of the Housing Act)	EDHA
초고층 및 지하연계 복합건축물 재난관리에 관한 특별법 (Special Act on Management of Disasters in Super High-Rise Building and Complex Buildings with underground Connections)	SAHB
초고층 및 지하연계 복합건축물 재난관리에 관한 특별법 시행령 (Enforcement Decree of the Special Act on Management of Disasters in Super High-Rise Building and Complex Buildings with underground Connections)	EDSAHB
특별피난계단의 계단실 및 부속실 제연설비의 화재안전기준(NFSC 501A)	NFSC501A

부록4: KBimCode Language 문법

```
grammar KBimCode;


kProgram
	:	kKBimCodeProgram EOF
	;


kKBimCodeProgram
	:	kCheckDef kStatGroupDef*
	;



// ********************************
kCheckDef
	:	CHECK kCheckP
		'{'LT* (kStatDef | kKOMDef)+ LT* '}'
	;


kCheckP
	:	'(' kLaw ')'
	;
kLaw
	:	BID
	;



// ********************************
// kStatGroupDef
kStatGroupDef
	:	kStatGroupVar
		'{' ((kKOMDef | kStatDef) LT?)+ '}'
	;
kStatGroupVar
	:	BID
```

```
        ;


///KOM
kKOMDef
        :       (kKOMDefStat | kKOMDecLines)+
        ;
kKOMDefStat
        :       kDefineKOM LT*
        ;
kDefineKOM
        :       kKOMDefStatDec kKOMDefBlock
        ;
kKOMDefStatDec
        :       (kWrapSpaceType | kWrapStructureType) BID LT*
        ;
kKOMDefBlock
        :       '{' LT* ( kKOMDecLines | kKOMPropExpr )+ '}'
        ;
kKOMPropExpr
        :       kKOMGetterExpr ((AND|OR) kKOMGetterExpr)* ';' LT*
        ;


kKOMGetterExpr
        :       kKOMGetterByProp
        |       kKOMGetterByBID
        ;


kKOMGetterByProp
        :       (kQuantifier '.')?
                (kWrapSpaceType | kWrapStructureType)
                ( ('.' BID) | ('.' (kWrapSpaceType | kWrapStructureType)) )+
                (kWrapSpaceType | kWrapStructureType)
                ( BID | kStringQuot | kLRMethod | INTLITERAL | DOUBLELITERAL )
        ;
kKOMGetterByBID
        :       (kQuantifier '.')?
```

```
                BID
                ( ('.' BID) | ('.' (kWrapSpaceType | kWrapStructureType)) )+
                kComparisonOperator
                ( BID | kStringQuot | kLRMethod | INTLITERAL | DOUBLELITERAL )
        ;
kKOMDecLines
        :       kKOMDecLine LT*
        ;
kKOMDecLine
        :       kWrapSpaceType kDecSingle ';'
        ;
kDecSingle
        :       BID '=' kKOMGetter
        ;
kKOMGetter
        :       (kGetterVerbs kSpaceGetterP ( ('+'|'-') kKOMGetter )?)
        |       kKOMGetterExpr
        ;
kGetterVerbs
        :       kLRMethodV
        |       kCtrlMethodV
        |       kVerb ( kWrapSpaceType | kWrapStructureType )
        ;
kSpaceGetterP
        :       '('
                ( kMultiSpaceGetter | kKOMGetterExpr )
                ')'
        ;
kMultiSpaceGetter
        :       kStringRep | kStringQuotRep
        ;

// ******************************
// kStatDef
kStatDef
        :       kStatLines+
```

```
        ;
kStatLines
        :       kStatLine LT*
        ;
kStatLine
        :       (kIfThenElseStat | kALUsStat)
        ;

///Condition Statement
kIfThenElseStat
        :       kIfStat  kThenStat  kElseIfStat*  kElseStat* ENDIF?
        ;
kIfStat
        :       IF NEG? '(' NEG?(kStatGroupVars| kALUStatConj)')'
        ;
kThenStat
        :       THEN? (kStatGroupVars| kALUStatConj)
        ;
kElseIfStat
        :       ELSEIF NEG?'(' (kStatGroupVars| kALUStatConj)')'
                kThenStat
        ;
kElseStat
        :       ELSE (kStatGroupVars| kALUStatConj)
        ;
kStatGroupVars
        :       kStatGroupVar ((AND|OR) kStatGroupVar)*
        ;

///ALU
kALUsStat
        :       kALUStatConj ';'
        ;
kALUStatConj
        :       kALUStat LT* ((AND|OR) kALUStat)*
        ;
```

```
kALUStat
        :       kOperand kComparisonOperator kOperand
        ;
kOperand
        :       kLRMethod
        |       kCtrlMethod
        |       BID
        |       BOOLEAN
        |       INTLITERAL
        |       DOUBLELITERAL
        ;



kComparisonOperator
        :       ( '=' | '==' | '>' | '<' | '>=' | '<=' )
        |       kComparisonOperatorNegation
        ;
kComparisonOperatorNegation
        :       ( '!=' | '! =' | '!==' | '! ==' )
        ;
///kMethod
kLRMethod
        :       kLRMethodV kLRMethodP
        ;
kLRMethodV
        :       kLRMethodVGetter
        ;
kLRMethodP
        :       '(' (
        (kWrapSpaceType| kWrapStructureType | BID| '"' BID '"')(','(kWrapSpaceType|
kWrapStructureType | BID| '"' BID '"'|

INTLITERAL))*
        | kLRMethod
        | kKOMGetterExpr
        )? ')'
        ;
```

```
kCtrlMethod
        :       kCtrlMethodV kCtrlMethodP
        ;
kCtrlMethodV
        :       KgetResult | KsetResult
        ;
kCtrlMethodP
        :       '(' kLaw ')'
        ;



kQuantifier
        :       KQuantifier
        ;

// *************
// Wrapper Data Type
kWrapSpaceType
        :       KBuilding
        |       KFloor
        |       KSpace
        |       KSpaceGroup
        |       KPath
        ;

// Extensible Structure Keywords
kWrapStructureType
        :       KStructure
        |       KColumn
        |       KWall
        |       KDoor
        |       KStair
        ;

kStringRep
```

```
        :       (BID |kWrapSpaceType| kWrapStructureType) ( ',' (BID |kWrapSpaceType|
kWrapStructureType) )*
        ;
kStringQuotRep
        :       kStringQuot ( ',' kStringQuot )*
        ;
kStringQuot
        :       '"' (BID |kWrapSpaceType| kWrapStructureType) '"'
        ;


kVerb
        :       Kget
        |       Kis
        |       Khas
        ;


kLRMethodVGetter
        :       KgetObject
        |       KgetBuilding
        |       KgetFloor
        |       KgetSpace
        |       KgetSpaceGroup
        |       KisExist
        |       KgetObjectCount
        |       KgetBuildingStoriesCount
        |       KgetElevatorCount
        |       KgetStairCount
        |       KgetObjectProperty
        |       KgetDoorType
        |       KgetWindowType
        |       KgetFloorNumber
        |       KgetElevLiveLoad
        |       KgetObjectMaterial
        |       KgetObjectUsage
        |       KgetBuildingUsage
        |       KgetFloorUsage
```

| KgetSpaceUsage
| KgetSiteUsage
| KgetObjectHeight
| KgetBuildingHeight
| KgetBuildingElevationHeight
| KgetSpaceHeight
| KgetCeilingHeight
| KgetFloorHeight
| KgetFloorElevationHeight
| KgetElementHeight
| KgetDoorHeight
| KgetStairStepHeight
| KgetObjectWidth
| KgetSpaceWidth
| KgetCorridorWidth
| KgetStairWidth
| KgetElementWidth
| KgetDoorWidth
| KgetWallWidth
| KgetStairStepWidth
| KgetObjectArea
| KgetFloorArea
| KgetLivingRoomArea
| KgetUndergroundFloorArea
| KgetBasementFloorArea
| KgetTotalFloorArea
| KgetGrossFloorArea
| KgetFloorAreaRatio
| KgetBuildingArea
| KgetSiteArea
| KgetElementArea
| KgetWallArea
| KgetWindowArea
| KgetObjectSectionalArea
| KgetObjectDiameter
| KgetObjectGradient

```
        |       KgetObjectMaterialType
        |       KisGrouped
        |       KisGroupedFirePartition
        |       KisGroupedFireWall
        |       KisFireResistantStructure
        |       KisFireProofStructure
        |       KisFirePartition
        |       KisLoadBearing
        |       KgetObjectFoundation
        |       KgetObjectStructure
        |       KhasObject
        |       KhasSpace
        |       KhasElement
        |       KgetPath
        |       KgetObjectDistance
        |       KgetObjectVerticalDistance
        |       KisConnectedTo
        |       KisExternal
        |       KisAdjacent
        |       KisSurrounded
        |       KisAccessible
        |       KisDirectlyAccessible
        |       KisGoThrough
        |       KgetObjectDirection
        |       KgetDoorSwingDirection
        |       KisEgressDirection
        ;


/*******************************
 KBimCode LEXER DEFINITION AREA
*******************************/

// *******************************
// KBimCode Keywords
CHECK          :       'check' | 'CHECK' | 'Check';
```

```
KQuantifier    :       'all'
               |       'one'
               |       'two'
               ;


// KOM - Spatial Keywords
KBuilding      :       'Building' ;
KFloor         :       'Floor' ;
KSpace         :       'Space' ;
KSpaceGroup    :       'SpaceGroup' ;
KPath          :       'Path' ;

// KOM - ExtensiKle Structure Keywords
KStructure     :       'Structure' ;
KSlaK          :       'SlaK' ;
KColumn        :       'Column' ;
KWall          :       'Wall' ;
KDoor          :       'Door' ;
KStair         :       'Stair';

// KBimCode Execution Verbal Keyworkds
Kget           :       'get' ;
Kis            :       'is';
Khas           :       'has';

KgetResult     :       'getResult';
KsetResult     :       'setResult';

// Shortcut keywords
KgetBuilding   :       'getBuilding' ;
KgetSpace      :       'getSpace' ;
KgetFloor      :       'getFloor' ;
KgetSpaceGroup :       'getSpaceGroup' ;
KgetPath       :       'getPath' ;
```

```
KgetDoor       :      'getDoor' ;


//LogicRule Method
KgetObject     :      'getObject';
KisExist       :      'isExist';
KgetObjectCount       :      'getObjectCount';
KgetBuildingStoriesCount
                      :      'getBuildingStoriesCount';
KgetElevatorCount     :      'getElevatorCount';
KgetStairCount        :      'getStairCount';
KgetObjectProperty    :      'getObjectProperty';
KgetDoorType          :      'getDoorType';
KgetWindowType        :      'getWindowType';
KgetFloorNumber               :      'getFloorNumber';
KgetElevLiveLoad      :      'getElevLiveLoad';
KgetObjectMaterial    :      'getObjectMaterial';
KgetObjectUsage               :      'getObjectUsage';
KgetBuildingUsage     :      'getBuildingUsage';
KgetFloorUsage        :      'getFloorUsage';
KgetSpaceUsage        :      'getSpaceUsage';
KgetSiteUsage         :      'getSiteUsage';
KgetObjectHeight      :      'getObjectHeight';
KgetBuildingHeight    :      'getBuildingHeight';
KgetBuildingElevationHeight
                      :      'getBuildingElevationHeight';
KgetSpaceHeight               :      'getSpaceHeight';
KgetCeilingHeight     :      'getCeilingHeight';
KgetFloorHeight               :      'getFloorHeight';
KgetFloorElevationHeight:    'getFloorElevationHeight';
KgetElementHeight     :      'getElementHeight';
KgetDoorHeight        :      'getDoorHeight';
KgetStairStepHeight   :      'getStairStepHeight';
KgetObjectWidth               :      'getObjectWidth';
KgetSpaceWidth        :      'getSpaceWidth';
KgetCorridorWidth     :      'getCorridorWidth';
KgetStairWidth        :      'getStairWidth';
```

```
KgetElementWidth      :      'getElementWidth';
KgetDoorWidth         :      'getDoorWidth';
KgetWallWidth         :      'getWallWidth';
KgetStairStepWidth    :      'getStairStepWidth';
KgetObjectArea        :      'getObjectArea';
KgetFloorArea         :      'getFloorArea';
KgetLivingRoomArea    :      'getLivingRoomArea';
KgetUndergroundFloorArea
                      :      'getUndergroundFloorArea';
KgetBasementFloorArea :      'getBasementFloorArea';
KgetTotalFloorArea    :      'getTotalFloorArea';
KgetGrossFloorArea    :      'getGrossFloorArea';
KgetFloorAreaRatio    :      'getFloorAreaRatio';
KgetBuildingArea      :      'getBuildingArea';
KgetSiteArea          :      'getSiteArea';
KgetElementArea               :      'getElementArea';
KgetWallArea          :      'getWallArea';
KgetWindowArea        :      'getWindowArea';
KgetObjectSectionalArea       :      'getObjectSectionalArea';
KgetObjectDiameter    :      'getObjectDiameter';
KgetObjectGradient    :      'getObjectGradient';
KgetObjectMaterialType:      'getObjectMaterialType';
KisGrouped            :      'isGrouped';
KisGroupedFirePartition       :      'isGroupedFireParition';
KisGroupedFireWall    :      'isGroupedFireWall';
KisFireResistantStructure
                      :      'isFireResistantStructure';
KisFireProofStructure :      'isFireProofStructure';
KisFirePartition      :      'isFirePartition';
KisLoadBearing        :      'isLoadBearing';
KgetObjectFoundation  :      'getObjectFoundation';
KgetObjectStructure   :      'getObjectStructure';
KhasObject            :      'hasObject';
KhasSpace             :      'hasSpace';
KhasElement           :      'hasElement';
KgetObjectDistance    :      'getObjectDistance';
```

```
KgetObjectVerticalDistance
                        :       'getObjectVerticalDistance';
KisConnectedTo          :       'isConnectedTo';
KisExternal             :       'isExternal';
KisAdjacent             :       'isAdjacent';
KisSurrounded           :       'isSurrounded';
KisAccessible           :       'isAccessible';
KisDirectlyAccessible   :       'isDirectlyAccessible';
KisGoThrough            :       'isGoThrough';
KgetObjectDirection     :       'getObjectDirection';
KgetDoorSwingDirection  :       'getDoorSwingDirection';
KisEgressDirection      :       'isEgressDireciton';




// Java keywords
EXTENDS         :       'extends' ;
IF              :       'if' | 'IF' | 'If';
THEN            :       'then' | 'THEN' | 'Then' ;
ELSEIF          :       'elseif' | 'ELSEIF' | 'ElseIf' | 'elseIf';
ELSE            :       'else' | 'ELSE' | 'Else' ;
ENDIF           :       'ENDIF' | 'END IF' ;
FOR             :       'for' ;

AND             :       'AND';
OR              :       'OR';

BOOLEAN         :       'TRUE' | 'FALSE';

NEG             :       '!';

// ******************************
// Identifiers for variable names..

BID             :       BIDprefix ( BIDprefix | INTLITERAL )* ;
```

```
BIDprefix      :      'a'..'z'|'A'..'Z'|'_' ;

// ********************************
// System Tokens
INTLITERAL     :
               IntegerNumber
               ;
IntegerNumber
       : '0'
       | '1'..'9' ('0'..'9')*
       | '0' ('0'..'7')+
       ;

DOUBLELITERAL
   :  NonIntegerNumber
   ;
NonIntegerNumber
   :  ('0' .. '9')+ '.' ('0' .. '9')* Exponent?
   |  '.' ( '0' .. '9' )+ Exponent?
   |  ('0' .. '9')+ Exponent
   |  ( '+' | '-' ) ('0' .. '9')+
       ;
Exponent
   :  'e' | 'E'
   ;



// ********************************
// System Lexer
COMMENT        : '/*' (options {greedy=false;} : .)* '*/' {$channel=HIDDEN;}
               ;
LINE_COMMENT   : '//' ~('\n'|'\r')* '\r'? '\n' {$channel=HIDDEN;}
               ;

LT // new lines
```

```
        : '\n'          {$channel=HIDDEN;} // Line feed
        | '\r'          {$channel=HIDDEN;} // Carriage return
        | '\u2028'      {$channel=HIDDEN;} // Line separator
        | '\u2029'      {$channel=HIDDEN;} // Paragraph separator
        ;


WS // white spaces: Tab, vertical tab, form feed, space, any other unicode "space separator"
        : (' '|'\r'|'\t'|'\u000C'|'\n') {$channel=HIDDEN;}
        ;
```