대화형 사례 기반 계획 시스템의 설계 및 구현

김만수[a], 유창훈[b], 김인철[c]

[abc]*경기대학교 전자계산학과*
경기도 수원시 영통구 이의동 산94-6, 443-760
Tel: +031-243-9670, E-mail:{i3rew[a],ych1026[b],kic[c]}@kgu.ac.kr

Abstract

사례 기반 계획 시스템은 과거의 유사한 사례 계획들을 이용함으로써 새로운 문제를 위한 계획을 효율적으로 생성할 수 있다. 하지만 대부분의 기존 사례 기반 계획 시스템들은 사례 검색 및 사례 일반화를 위한 제한적 기능들만을 제공할 뿐만 아니라, 계획 생성과정에 사용자 참여를 허용하지 않는다. 이러한 문제점들을 극복하기 위해, 본 논문에서는 새로운 사례 기반 계획시스템인 JCBP를 제안한다. 본 논문에서는 먼저 JCBP 시스템의 설계와 구현에 대해 설명하고, 실험을 통해 JCBP 시스템의 성능을 분석한다. JCBP 시스템은 효율적인 메모리 사용과 사례 검색을 위해 각 도메인의 동일한 작업목표를 가진 사례들을 개별 사례베이스로 그룹화하고, 이들에 대한 색인들을 유지한다. 또 이 시스템은 문제모델로부터 자동으로 추출한 휴리스틱 지식을 사례 검색과 적응 단계에 이용하며, 목표 회귀를 통한 사례 일반화 기능을 제공한다. 또한 JCBP 시스템은 대화형 모드를 통해 혼합 주도 계획 생성 기능을 제공한다. 따라서 사용자의 지식과 선호도를 이용할 수 있어, 계획 생성의 복잡도를 줄이고 사용자의 만족도를 높일 수 있다.

Keywords:

Case-Based Planning; Indexing; Case Adaptation; Case Generalization; Interactive Planning

서론

현재에는 실시간 계획 응용 시스템의 요구가 증가하고 있다. 지능 로봇의 제어, 우주 탐사선 및 인공위성의 제어처럼 복잡한 시스템부터 인터렉티브 게임이나 워크플로우까지 계획을 자동으로 생성하고자 할 때 계획 시스템을 필요로 한다. 하지만 기존의 생성적 계획 시스템(generative planning system)은 문제 해결에 있어서 몇 가지 한계점을 지니고 있다. 우선 복잡도가 높은 실 세계 문제 해결에 성능이 낮으며, 새로운 문제가 주어질 때마다 처음부터 계획을 생성해야 한다. 따라서 빈번하게 발생하는 실시간 계획 문제에 효과적으로 대처하기 어렵다.

이러한 문제를 극복하기 위한 노력의 하나로 사례 기반 계획 시스템(case-based planning system)이 제안되었다. 사례 기반 계획 시스템은 유사한 과거의 사례 계획(case plan)을 재사용하는 시스템으로서 계획 생성에 필요한 노력을 줄이고 빠르게 해 계획을 생성 할 수 있는 시스템이다. 하지만 기존의 사례 기반 계획 시스템은 몇 가지 제한점들을 가지고 있다. 첫 번째, 비효율적인 사례 분류와 색인으로 사례 검색의 성능이 낮다. 두 번째, 대용량의 사례 베이스 저장을 위한 관계형 데이터베이스 시스템에 대한 연동 기능이 미흡하다. 세 번째, 기존 시스템은 사례 일반화를 위한 제한적 기능들만 제공한다. 네 번째, 기존 시스템은 계획 생성 과정에 사용자가 참여할 수 없고 사용자의 도메인 지식과 선호도를 반영할 수 없다. 따라서 본 논문에서는 이러한 기존의 사례 기반 계획 시스템들의 제한점을 극복하기 위해 새로운 사례 기반 계획 시스템인 JCBP 시스템을 제안한다.

JCBP 시스템에서는 시스템의 효율적 메모리 사용과 사례 검색을 위해 각 도메인의 동일한 작업 목표를 가진 사례를 개별 사례베이스로 그룹화하여 관리하고, 이들에 대한 색인을 통해 접근한다. 대량의 사례를 저장하고 관리하게 위해 사례 데이터베이스는 외부의 저장장치를 사용하며, 실제 입출력이 많은 사례 그룹만 메모리에 올라온 상태로 시스템은 동작한다. 이 시스템에서 사용하는 사례 검색은 CMR 기반이며, 초기의 조건이 얼마나 유사한지 측정하여 판단한다. 이 방법은 속도가 빠르기 때문에 많은 사례들 중에서 적합한 사례를 판단하는데 적합하다. 또한 이 시스템은 문제 모델로부터 자동으로 추출한 휴리스틱 지식을 적응 단계에 이용한다. 생성적 계획 적응 방법은 검색한 사례를 사용하기 위해, 새로운 문제에 대한 계획을

생성함으로서 진행한다. 또한 사례가 적응 가능하면 항상 해를 보장해 주기 때문에 높은 신뢰성을 갖는다.

사례 저장을 하기 위해 목표 회귀를 통한 사례 일반화 기능을 제공 한다. 문제에서 제시한 목표 조건으로부터 그 목표가 도달하기까지, 회귀 과정을 반복하면서 그 목표가 도달하는 초기 조건의 집합을 생성해서 일반화 한다. 목표회귀를 통해 만들어진 초기 조건의 집합은 새로운 사례 계획이 적용되기 위한 최소한의 조건으로만 이루어진 집합이다. 따라서 JCBP 시스템은 사례 일반화를 통해 사례의 적용 범위를 넓힐 수 있고, 사례 베이스의 크기를 적정한 수준으로 유지할 수 있다. 또한 JCBP 시스템은 대화형 모드를 통해 혼합 주도 계획 생성 기능을 제공한다. 따라서 사용자의 지식과 선호도를 이용할 수 있어, 계획 생성의 복잡도를 줄이고 사용자의 만족도를 높일 수 있다.

사례 기반 계획

사례 기반 계획(case-based planning)은 사례 기반 추론(case-based reasoning)의 한 응용 형태이다. 일반적으로 계획 검색, 계획 적응 그리고 계획 저장의 단계를 거치며 계획을 생성한다. 계획 검색은 현재의 계획과 가장 유사한 해결된 문제를 검색하는 단계이다. 계획 적응은 새로운 문제를 해결하기 위해서 검색된 계획을 적합하게 바꾸는 과정이다. 계획 저장은 추후 계획을 더 효율적으로 사용하기 위해서 새로운 계획을 저장하는 방법을 설계한다. 만일 새로운 계획이 실패한다면 그것의 실패에 대한 정당성까지 저장한다.

계획의 검색, 적응 및 저장은 사례 기반 계획 시스템의 가장 기본적인 작업이며 사례 기반 계획 순환 구조(case-based planning cycle)를 이룬다. Figure 1은 새로운 계획 문제가 주어졌을 때 그것을 해결하는 사례 기반 계획 순환 구조를 보여준다. 새로운 문제가 주어지면, 가장 먼저 메모리 내에 사례들의 집합인 사례 베이스가 표현되고, 문제를 해결하기 위한 도메인 지식이 표현된다. 그 이후 사례 베이스 중에서 문제를 해결하기 위한 최적의 사례를 검색하는 사례 검색(case retrieval)단계를 거치게 된다. 사례 검색 과정에서는 사례를 재사용하는데 최소한의 노력을 기울이기 위한 사례를 구하기 위한 단계이다. 다음 단계는 사례 적응(case adaptation)과정이다. 이 과정은 검색된 사례를 실제로 사용하기 위해 변형을 거치는 과정이며, 주어진 문제는 적응된 사례를 사용함으로써 해를 구할 수 있게 된다. 이렇게 해결된 문제는 새로운 사례로써 다시 저장되어야 하며 이 과정은 사례 저장(case retention)과정을 거치며 이루어지게 된다. 사례 저장 과정을 거친다는 의미는 다음 문제가 주어졌을 시 재사용을 위한 형태로 변형되어 저장됨을 의미하며 이 과정을 거치면서 색인이 부여되거나, 추상화 단계를 거치게 되면서 검색이 용이하게 된다.

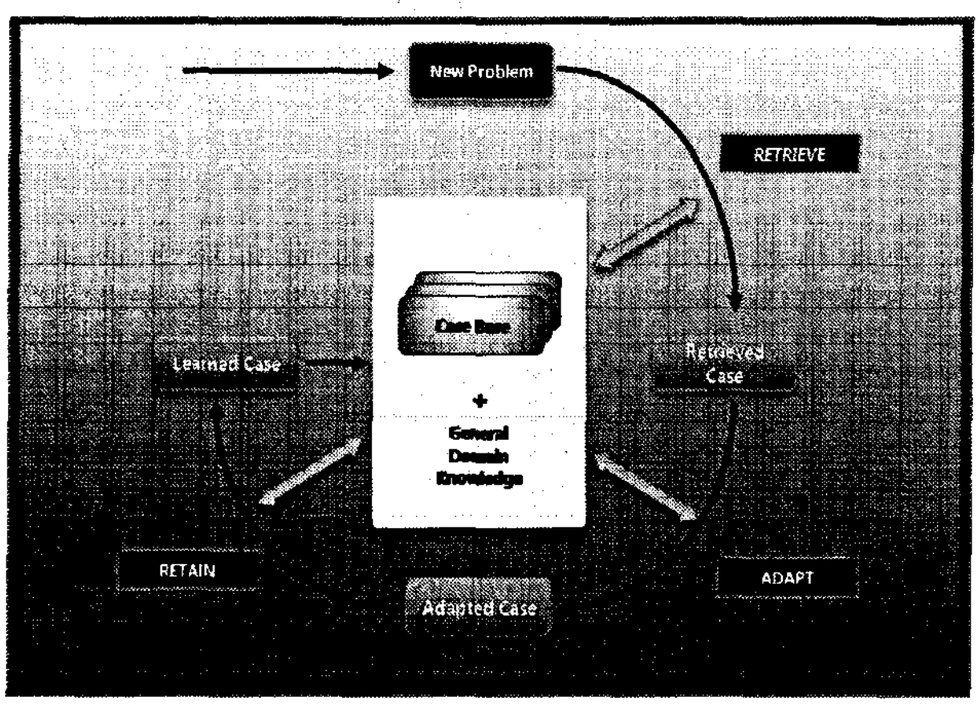

Figure 1 – 사례 기반 계획 순환 구조

사례 기반 계획 문제와 연관된 기초 개념들을 다음과 같이 정의할 수 있다.

계획 도메인(planning domain)

계획 도메인 ***D***는 (***C***, ***A***)로 나타내며 ***C***는 논리 조건(condition)들의 집합이고 ***A***는 동작들의 집합이다

동작(action)

동작 ***a***는 (***Preconds***, ***Effects***)의 쌍으로 이루어지며, Preconds는 전 조건(preconditions)들의 집합을 말하며 Effects는 그 동작이 수행 되었을 때 나타나는 효과(effect)를 나타내는 조건들의 집합이다. 동작 ***a***가 $a \in A$의 관계이면 동작 ***a***의 전 조건은 C_a로 표현하고, 효과는 E_a로 표현한다.

계획 문제(planning problem)

계획 문제 P는 (***D***, ***I***, ***G***)로 표현하며 ***D***는 계획 도메인, ***I***는 초기 상태(initial state)를 나타내는 조건들의 집합 ***G***는 목표 조건(goal conditions)들의 집합을 나타낸다. ***I***와 ***G*** 모두 도메인에서 정의한 술어 논리 (predicate logic)로 표현한다.

사례 베이스(case base)

사례 베이스 ***CB***는 $\{z \mid z = (P, S)\}$로 표현하며 ***P***는 계획문제, ***S***는 ***P***에 대한 하나의 해 계획(solution plan)을 나타내며 ***z***는 사례(case)를 표현한다.

사례 기반 계획 문제(case-based planning problem)

사례 기반 계획 문제 ***CP***는 (CB, P_{new})로 나타내고 ***CB***는 과거 사례들의 집합인 사례 베이스이며 P_{new}는 새로운 계획 문제를 표현한다.

시스템의 설계

JCBP 시스템은 상태공간계획(state-space planning)에서 동작하는 사례 기반 계획 시스템이다. 상태공간 상에서 수행되는 상태 공간 계획은 술어 조건식들의 집합으로 각 월드 상태를 표현하고 동작들의 적용 효과로 상태 변경을 나타낸다. 따라서 JCBP 시스템에서는 사례를 표현하고 저장 관리하는 방법부터 그것을 이용하고 변형시키는 모든 과정이 상태공간 상에서 이루어진다.

시스템 구조

Figure 2는 JCBP 시스템 구조를 설명하고 있다. 전체 시스템은 크게 여섯 개의 부분으로 구성된다. 사례 메모리(case memory), 사례 검색기(case retriever), 사례 적응기(case adapter), 사례 저장기(case retainer), DB 관리기(DB manager) 그리고 사례 데이터베이스(case DB)이다.

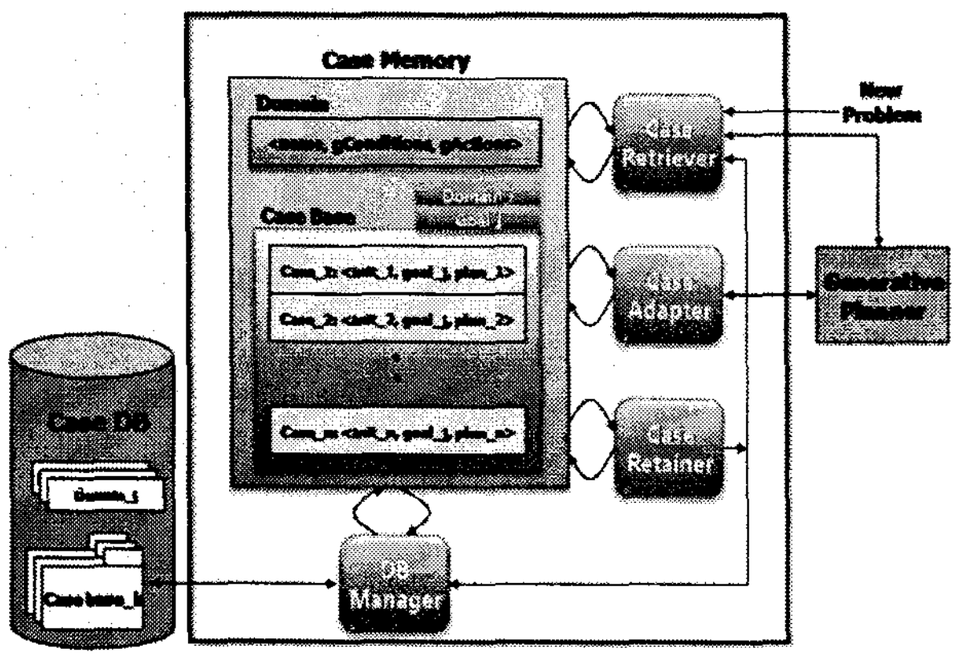

Figure 2 – JCBP 시스템 구조

사례 데이터베이스는 전체 사례 집합을 저장하는 관계형 데이터베이스이다. 문제를 해결하는 과정에서 실제로 적용되는 사례는 하나이며, 그 하나의 사례를 선택하기 위해서는 검색의 과정을 거치게 된다. 문제가 주어질 때마다 전체 사례 집합을 포함하고 있는 사례 데이터베이스에서 문제를 해결하는데 적합한 사례를 찾는 것은 비효율적이다. 본 시스템에서는 모든 사례가 아닌 일부의 사례만 데이터베이스에서 가져와서 메모리에 유지시켜 놓는다. 이러한 과정이 진행되기 위해서는 사례가 그룹화 된 상태로 분류해서 저장되어야 한다. 본 시스템에서는 사례 데이터베이스에 문제의 도메인에 따라서, 그리고 문제의 목표 조건에 따라서 그룹화한 후 분류 저장한다.

사례 검색기는 사례 메모리로부터 문제 해결에 가장 적합한 사례를 찾는 역할을 한다. 사례 검색기에 의해 선택된 사례는 사례를 적응하는 과정에서 최소의 노력으로 문제를 해결할 수 있어야 한다. 사례 적응기는 검색한 사례를 재사용 가능하게 변환하는 작업을 담당한다. 실제 계획을 생성하는 핵심적인 부분이다. 사례 저장기는 해결한 계획을 사례화 한다. 많은 정보를 잃지 않으면서도 재사용 가능하도록 계획을 변형 시킨다. 또한 추후 편리한 사용을 위해서 색인화 작업을 한다. 보다 자세한 내용은 이후의 절에서 다룬다.

사례 표현과 메모리 구성

사례 표현은 이전의 해결한 계획 문제를 메모리 내에 표현하고 조직하는 것을 의미한다. 사례 기반 계획 시스템에서 사례가 어떻게 표현되느냐에 따라 시스템 전체에 효율성과 능률에 영향을 미친다. JCBP 시스템은 도메인 정보(domain information)와 작업(task), 사례의 세 가지 개념을 기본으로 사례를 표현하고 분류한다. 각 도메인 정보는 계획 문제가 정의될 수 있는 하나의 계획 도메인을 나타내며, 앞서 정의한 바와 같이 해당 도메인의 각 상태를 표현하기 위한 논리 표현식들의 집합과 상태를 변경할 수 있는 동작들의 집합으로 구성된다. 예컨대 (on cup table)은 하나의 조건식을, (pickup robot tablearea cup table)은 동작을 각각 나타낸다.

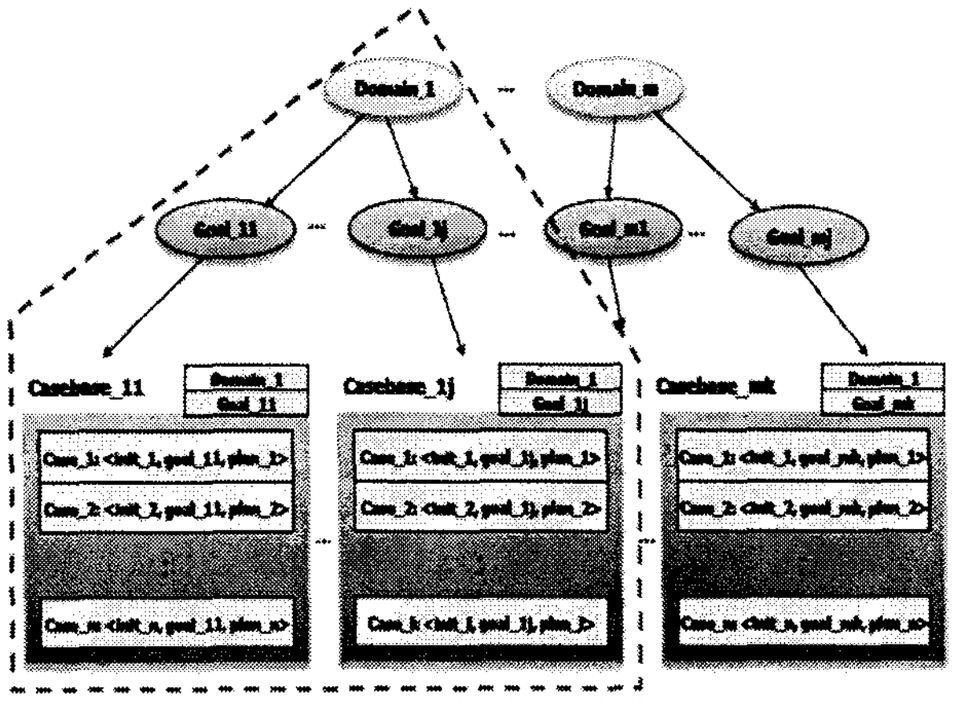

Figure 3 – 사례 표현과 분류

Figure 3은 JCBP 시스템의 메모리 내 사례 표현과 분류를 나타낸 것이다. 도메인 정보에 따라서 최초 분류가 되며, 목표 조건에 따라서 그 다음 분류가 된다. 이렇게 분류된 사례들은 개별 사례 베이스로 그룹화 되며, 문제를 해결할 때 메모리에서 읽고 쓰는 최소의 단위이다. 대부분의 사례 기반 계획 시스템에서 계획을 생성하는 동안에 도메인이 바뀌지 않는다. 따라서 도메인 정보를 독립된 개념으로 유지하여 사용하면 한 번 메모리에 읽어 들인 정보를 계속 사용 할 수 있다. 따라서 같은 도메인이면서 계획 문제만 바뀌는 경우에는 개별 사례 베이스만 바뀌고 데이터베이스에서도 개별 사례 베이스의 정보만 가져와도 되므로 메모리의 효율성을 높일 수 있다.

하지만 메모리 내에 도메인을 표현하고자 할 때

술어 조건식이나 동작의 표현을 그대로 나타내는 것은 비효율적이다. 따라서 본 시스템은 술어 조건식과 동작에 색인 번호를 부여하고, 그것을 통하여 메모리를 구성하고 있다. 또한 술어 논리 조건식과 동작의 각각에 대한 색인 번호와 그것의 기호 표현을 맵핑 테이블을 운영하여 관리한다. 한편, 각 작업은 <초기조건, 목표조건>의 쌍으로 이루어진 하나의 계획 문제를 나타낸다. 각 사례는 하나의 계획 문제, 즉 하나의 작업과 이것을 해결하기 위한 사례 계획으로 구성된다. 사례 계획은 작업의 한 초기상태에서 시작하여 목표조건이 만족되는 상태까지 도달 가능한 일련의 동작들로 표현된다. 일반적으로 동일한 계획 도메인에서 서로 다른 여러 작업들이 정의될 수 있고, 동일한 하나의 작업에 대해서도 서로 다른 다양한 사례 계획 들이 존재할 수 있다. Figure 3에서 각 사례들은 초기조건, 목표조건, 사례 계획 등으로 표현되며, 이러한 각 사례들을 먼저 도메인 별로 분류하고, 다시 목표 조건이 일치하는 작업 별로 분류한다. 이와 같이 분류된 사례 집합은 동일한 도메인 정보와 동일한 목표조건을 가진 사례들의 소 그룹으로 나뉘어지며 이 소 그룹 별로 사례 베이스를 분할하여 사례 데이터베이스와 사례 메모라 간의 이동이 발생한다.

사례 검색

사례 검색(case retrieval)은 주어진 문제를 해결하기 위해서 기존의 사례들 중에서 현재 문제에 가장 유사한 사례를 선택하는 과정을 말한다. 사례 검색은 목표 조건이 같은 개별 사례 베이스에 있는 사례들을 대상으로 검색한다. 본 시스템은 유사도의 척도로 CMR(Condition Matching Ratio)를 사용한다. CMR은 새로운 계획 문제와 사례의 유사도를 측정하는 방법으로서, 계획 문제와 사례 간에 각각의 초기 조건이 얼마나 일치하는지를 측정하여 유사성을 판단한다.

$$Similarity(P_{new}, P_{case}) = \frac{|I_{new} \cap I_{case}|}{|I_{new} \cup I_{case}|} \quad (1)$$

Equation 1은 유사도 값을 구하는 수식을 나타내었다. 계획 문제의 초기 조건과 사례의 초기 조건 사이의 교집합을 구해서 일치하는 조건의 개수를 파악한다. 그리고 계획 문제와 사례의 초기 조건에 대한 전체 집합의 개수로 나누어 값을 계산한다. 전체 집합의 개수로 나누면, 초기 조건의 일치하는 정도를 전체 집합의 비율에 따라서 값을 판단하게 된다. 이 의미는 계획 문제의 모든 초기 조건을 고려해서 유사성을 판단하고자 함이다.

사례에 대해 유사도 값이 측정되면, 그 값으로 사례 베이스의 사례를 정렬하게 되고 가장 유사한 사례가 무엇인지 판단 할 수 있게 된다. 이후 사례를 선택하게 되는데, 선택의 과정은 일반적으로 가장 높은 값의 유사도를 갖는 사례를 선택한다. 하지만 사용자의 판단 하에 더 좋은 결과 값을 위해서 새로 계획을 작성할 수도 있으며, 임계치(threshold) 값을 부여함으로써 유사도 값이 낮은 사례는 선택 대상에서 제외시킬 수 있도록 하였다.

사례 적응

JCBP 시스템은 검색된 사례를 재사용하기 위해서 적응 과정을 진행한다.

JCBP 시스템의 사례 적응 방법은 생성적 적응(generative planning adaptation) 방법이다. 이 방법은 사례 문제 ***P***를 ***(D, I, G)***로 가정하고 현재 문제 ***P'***를 ***(D, I', G')***로 가정하였을 때, ***P***의 초기조건***I***와 ***P'***의 초기조건 ***I'*** 간의 불일치성을 해소하는 알고리즘이다. 이 불일치성을 해소하기 위해서, 현재 문제의 초기 조건부터 사례 문제의 초기 조건까지의 계획 문제를 생성적 계획 문제 ***P''***는 ***(D, I', I)***로 정의한다. 그리고 ***P''***에 대한 해 계획을 사례 계획에 추가하여 전체 해를 구한다.

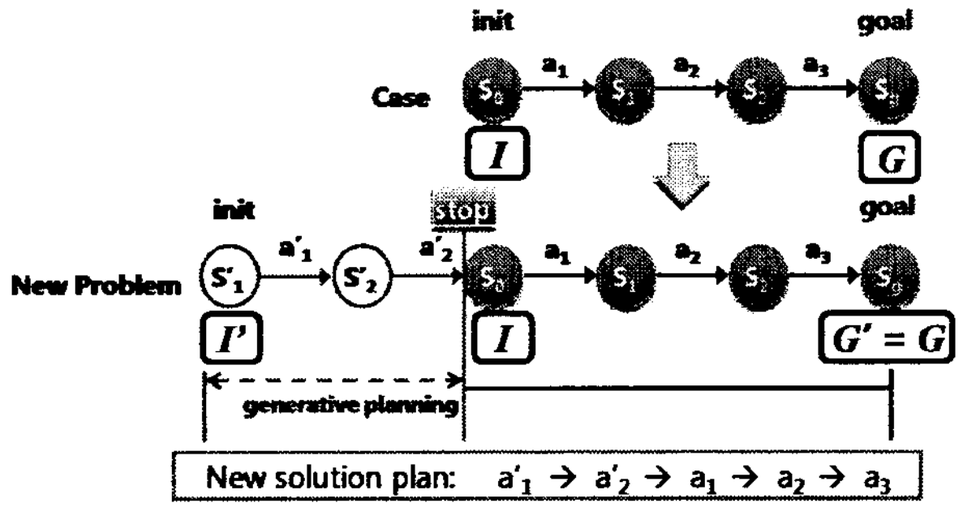

Figure 4 – 생성적 적응 방법

Figure 4는 본 적응 방법을 도식화 한 것이다. 기존 사례 문제 ***P***에 대한 해<a_1,a_2,a_3>는 그대로 이용하고, 생성적 계획 문제 ***P''***에 대한 해<a'_1,a'_2>를 앞쪽에 추가하면서 전체 해< a'_1,a'_2, a_1,a_2,a_3>을 구한다.

본 시스템에서 사용하는 적응 방법은 상태 공간 상에서 이루어진다. 상태 공간 계획은 기본적으로 전향 탐색(forward-search)로 해를 보장한다. 전향 탐색 방법으로는 최적 우선 탐색(best-first search), A* 알고리즘, 언덕 오르기(hill-climbing) 등이 있다.

본 시스템에서는 탐색에 휴리스틱 지식을 이용하는 A* 알고리즘을 적용한다. 휴리스틱 값을 구하는 알고리즘으로는 계획 그래프(planning graph)를 사용한다. 계획 그래프는 Graphplan 계획기[15]를 통해 처음 소개 되었고, 탐색 공간 상의 한 상태를 시작 노드로 삼아 실제에 근사한 탐색 트리를 펼쳐봄으로써 효과적인 도달성 휴리스틱(reachability

heuristic)을 손쉽게 계산할 수 있는 자료구조이다[14]. 또한 이 자료구조는 기존의 부정적 효과는 삭제하고 상호간의 간섭효과를 제거한 간략화 된 계획 그래프(relaxed planning graph)를 사용하고 있다.

사례 일반화 및 저장

사례 저장은 새로 구해진 계획을 또 다른 하나의 사례로 저장하는 단계이다. JCBP 사례 기반 계획 시스템은 새로운 사례를 저장하기 전에 사례 일반화(generalization) 과정을 수행한다.

일반화를 위한 방법으로는 미리 정의된 하나의 계층(hierarchy)구조에 따라 사례를 표현하는 구성요소를 추상화(abstraction)화는 방법이 있다. 또 다른 방법으로는 사례 표현에 등장하는 상수들을 파라미터화(parameterization)하는 방법도 있다. 예를 들어, (at circuitbreaker circuitbreakerarea)의 경우 (at ?staticObject ?location)로 바꾸어 상수를 모두 변수로 바꾸는 방법이다.

JCBP 시스템에서는 새로운 사례의 초기 상태 조건들 중 불필요한 부분을 제거해서 사례 일반화를 진행한다. 이러한 사례 일반화 방법 위해서 본 시스템은 목표 회귀(goal regression) 일반화 방법을 사용한다.

Table 2 – 목표 회귀 알고리즘

/* Given the plan as a sequence of solution actions $<a_1,a_2,\ldots,a_n>$ and the goal condition G */
I = G
  For i = n to 1
    I = Preconds(a_i) ∪
        {(I – Effects$^+$(a_i)) ∪ Effects$^-$(a_i)}
  End
Return I

Table 2는 본 시스템에서 사용한 목표 회귀 알고리즘을 나타내었다. 이 알고리즘은 최초 목표 상태에서부터 출발 하고, 사례의 행동 순서를 역으로 추출하며 시작한다. 가장 먼저 행동의 선행 조건, **Preconds(a_i)**를 추가한다. 이후 행동이 일어났을 시 발생하는 부정적인 효과, **Effects$^-$(a_i)**를 추가한다. 마지막으로 행동이 나타냄으로 인해 생기는 긍정적인 효과, **Effects$^+$(a_i)**를 삭제한다. 이러한 계산 방식은 주어진 동작들을 차례로 적용함으로써 목표 조건을 달성하기 위한 계획의 필수 전조건들(preconditions)을 가려내는 과정이다. 이 전조건들은 사례 계획이 성공적으로 수행되기 위한 최소한의 조건이 된다. 또한 외부 저장장치로 저장하는 부분에서도 최소한의 정보만을 취하고 있으므로 메모리 관리 면에서도 많은 장점을 갖고 있다.

다음은 JCBP 시스템에서 대용량의 사례를 저장하기 위한 DB의 스키마이다. 총 3 개의 테이블로 이루어졌으며, 각 계획 도메인의 정보를 포함하는 도메인 테이블과 각 개별 사례들의 세부 정보가 저장된 사례 테이블이 있다. 또한 이 두 개의 사례를 연결하는 역할 뿐만 아니라 실제 문제를 해결하는 작업 중심으로 이루어진 작업 테이블이 있다.

Table 3 – Domain Table

Field	Type	Key	Null
DomainName	varchar<100>	PRIMARY	no
stateMap	varchar<700>		no
actionMap	varchar<700>		no
gActions	varchar<1000>		no

Table 4 – Task Table

Field	Type	Key	Null
DomainName	varchar<100>	PRIMARY	no
GoalConditions	varchar<500>	FOREIGN	no

Table 5 – Case Table

Field	Type	Key	Null
GoalConditions	varchar<500>	PRIMARY	no
InitConditions	varchar<1000>		no
Plan	varchar<500>		no
Length	integer		no

Table 3,4,5 는 도메인 영역과 작업 영역 그리고 사례에 대한 각각의 테이블에 대한 스키마를 나타내었으며, 이러한 구조는 위에서 설명한 사례의 표현 방법에 따라 설계되었다. 따라서 JCBP 시스템이 문제를 해결함에 있어 최초 도메인 영역을 읽어오고, 같은 작업의 개별 사례 베이스만 메모리에 읽어옴으로써 메모리 효율성을 높였다. 결과적으로 대용량의 사례를 작은 메모리 용량으로도 제어 가능하게 되었다.

시스템의 구현

JCBP 시스템은 자동 모드(auto mode)와 대화형 모드(interactive mode)를 지원 한다. 자동 모드는 시스템에서 그래픽 사용자 인터페이스(GUI)의 지원 없이 계획을 생성해 나가는 모드이다. 따라서 사용자가 도메인 및 문제 설정을 직접 입력한 후 프로그램을 실행시키면서 문제를 해결해 나간다. 사용자는 프로그램을 실행 시킨 후 결과 값을 바로 확인 할 수 있다는 장점도 있으나, 해를 구하는 과정 중 사용자가 참여할 수 없다는 단점도 있다. 대화형 모드는 사용자가 시스템과 함께 계획을 생성할 수 있도록 하는 모드이며, 다음 절에서 자세히 다룬다.

대화형 모드

본 시스템은 대화형 모드를 지원하기 위한 GUI를 제공한다. GUI는 Figure 5와 같이 사용자와 시스템 간에 서로 지식과 결정사항이 인터페이스를 통해 계획을 생성할 수 있도록 기능을 제공한다. 현재 일어나고 있는 시스템 내부의 상황을 사용자가 이해하기 쉽게 표현할 수 있고, 또한 사용자의 의도가 잘 반영되도록 구성되어야 한다. 따라서 Figure 5처럼 시스템의 계획 진행에 따라 많은 정보를 보여주기 위해 탭 브라우저 (tab browser) 방식을 택했다.

탭 브라우저의 가장 처음 탭 메뉴인 도메인 관리기(domain manager)는 특정 도메인을 선택하고 관리하는 일을 담당한다. 도메인 관리기는 DB에 저장된 특정 도메인을 지정할 수 있고, 그 지정된 도메인을 사용자가 수정 가능하게 하였다. 또한 새로운 도메인을 생성할 수 있는 기능을 제공한다.

다음 탭 메뉴는 문제 관리기(problem manager)이다. 사용자는 문제 관리기를 통해 도메인 영역에 맞는 계획 문제를 새롭게 정의 할 수 있다. 하지만 계획 생성에 대한 전문 지식이 없는 사용자의 경우, 계획 문제를 새롭게 정의 하기란 쉽지 않다. 따라서 문제 관리기에서는 도메인과 관련 있는 계획 문제의 리스트를 보여주고, 특정 계획 문제를 수정함으로써 계획 문제를 정의할 수 있는 기능도 갖고 있다. 또한 도메인 관리기나 문제 관리기에서 다루는 표현은 모두 술어논리이며, 일반 사용자가 술어논리를 보고 이해하기란 힘들다. 따라서 도메인과 문제 마다 사용자는 자연어로 설명(comments)을 추가할 수 있으며, 이 정보는 사용자가 쉽게 알아볼 수 있기 때문에 도메인과 문제의 정의에 도움을 준다.

다음 탭 메뉴는 사례 검색기이다. 유사한 사례를 검색하기 위해서 중앙의 테이블 뷰어에 관련 사례들을 열거하게 된다. 사용자가 유사도 측정 알고리즘을 선택하게 되면, 사례 마다 유사도 값을 갖게 되며 유사도 값을 기준으로 정렬할 수 있다. 사용자는 간단한 마우스 드래그로 유사도의 임계치를 부여하면서 특정 임계치 이상의 유사도를 가지는 사례들만을 검색할 수 있다. 이러한 기능은 사용자가 대량의 사례 베이스로부터 원하는 사례를 효과적으로 검색하도록 도움을 준다. 또한 사례 검색기 GUI에서는 사례를 검색하는 중간에도 도메인과 문제에 대한 정보를 항상 참조할 수 있도록 간략화 된 정보를 제공한다.

다음 탭 메뉴는 사례 적응기이다. 사례를 적응하기 위해서는 사례 검색기를 통해 검색된 사례를 이용한다. 따라서 사례 적응기에서는 검색된 사례의 정보를 알 수 있는 뷰어가 제공된다. 사례 적응은 선택된 사례를 사용하여 해를 구하는 과정을 사용자가 확인하고, 그 과정에 참여해서 더 좋은 해를 구할 수 있도록 GUI를 지원한다. 생성적 적응을 위한 전 과정을 투명하게 보여주기 위해 중간 상태들과 적용 동작들을 모두 확인할 수 있도록 하였으며, 또한 적응과정에 소요된 시간을 보여줌으로써 사례 적응에 든 노력을 알 수 있도록 하였다. 또한 사용자가 적응 결과로 얻어진 해 계획을 시각적으로 이해하고 서로 비교할 수 있도록 계획 브라우저와 계획 시뮬레이터를 구현하여 제공한다.

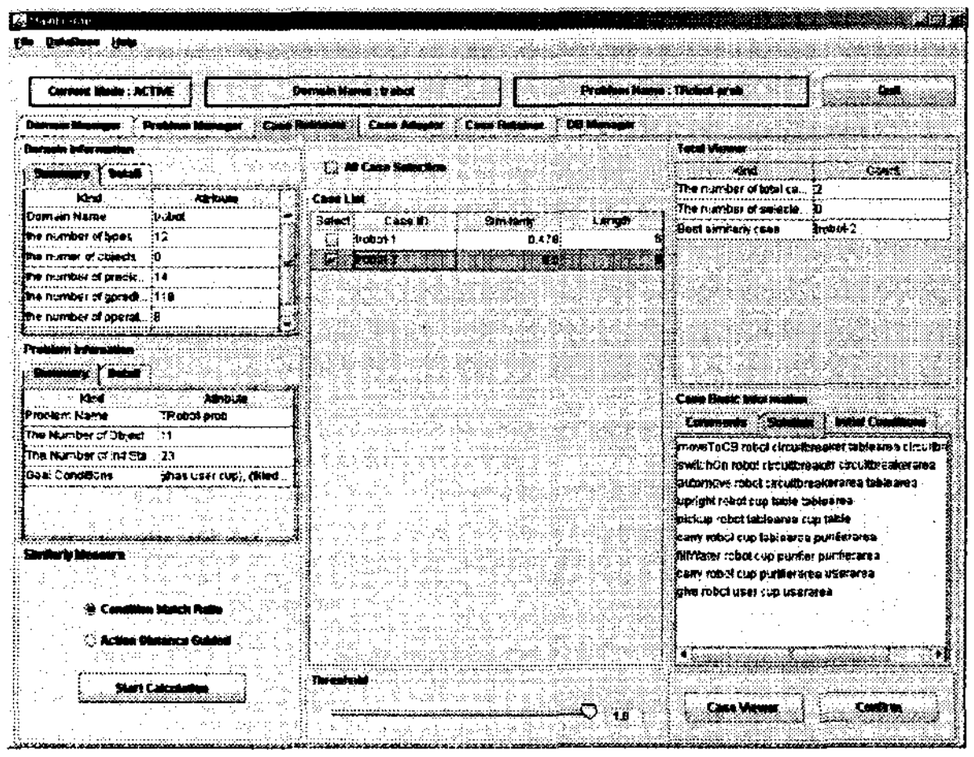

Figure 5 – 사용자 대화형 모드를 지원하는 GUI

다음 탭 메뉴는 사례 저장기이다. 사례 저장기에서는 해결 된 문제를 일반화 한 후 새로운 사례로 저장하는 일을 담당한다. 일반화 과정을 사용자가 확인 후 저장의 여부를 판단하게 된다. 또한 모든 정보의 저장을 위해서 일반화 과정을 생략할 수 있는 기능도 있다.

다음 탭 메뉴는 DB관리기이다. 많은 DB의 내용을 쉽게 확인하기 위해서 트리구조를 갖는 GUI를 지원하며, 이것은 곧 사례가 저장된 계층구조와 색인 구조를 그대로 구현한 것이다. 따라서 사용자는 사례에 쉽게 접근할 수 있고 관리가 용이하다. 또한 DB관리기는 DB와 메모리 영역과의 자료의 동기화 기능, 현재 작업한 내용의 저장 기능, DB의 자료 삭제 및 초기화 기능을 갖고 있다.

계획 브라우저

계획 브라우저는 사용자가 계획 문제를 풀어가는 도중에 각 계획이나 새로 생성된 해 계획에 대한 상세한 정보를 보고자 할 때 그것을 시각화 해주는 도구이다. Figure 6은 본 시스템의 계획 브라우저 의 실행화면이다.

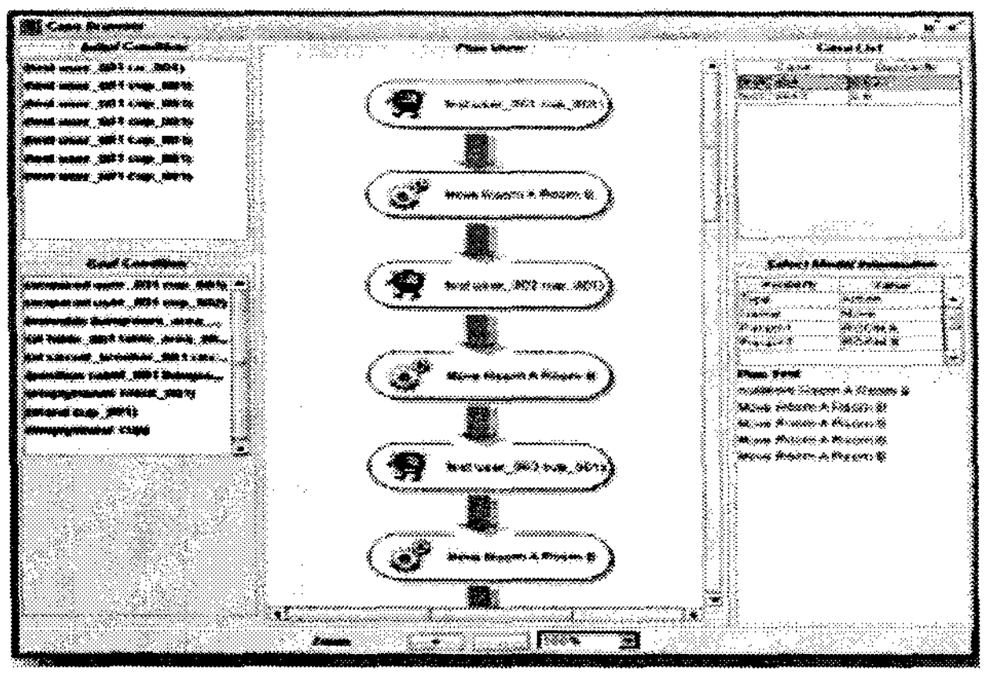

Figure 6 – 계획 브라우저의 실행화면

계획 브라우저는 초기 조건부터 목표 조건을 달성하기 위한 동작의 순서를 보여주는 것이 주 기능이다. 따라서 계획 브라우저는 초기 조건과 목표 조건의 정보를 자세히 보여주고 있으며, 특히 동작의 순서를 사용자가 알기 쉽게 아이콘으로 표기하여 나타내고 있다.

이와 같은 계획 브라우저의 도움으로 사용자는 사례 문제와 사례 계획에 대한 정보를 쉽게 파악할 수 있다. 따라서 사용자가 이전의 해결된 계획에 대한 자세한 정보를 열람하고, 그것으로부터 현재의 문제를 어떻게 해결해 나갈지 올바른 결정을 할 수 있게 된다.

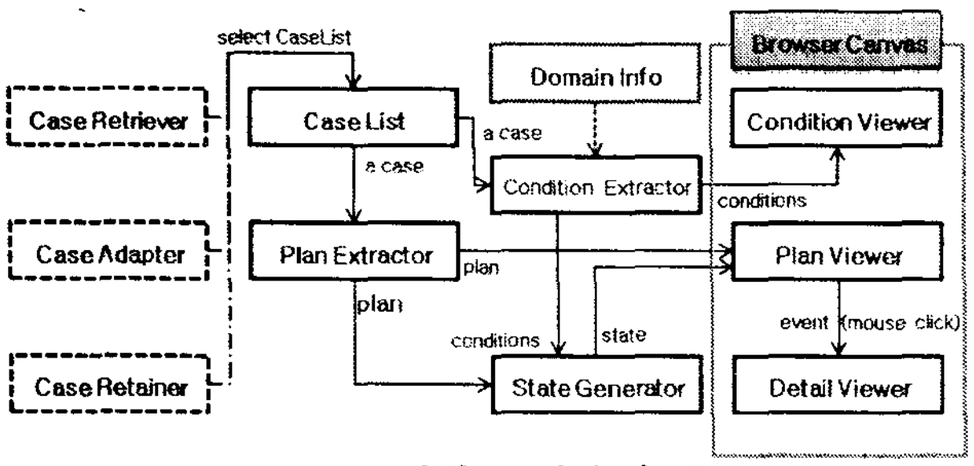

Figure 7 – 계획 브라우저 구조도

Figure 7은 계획 브라우저의 내부 구조를 나타낸 것이다. 사례 검색기, 사례 적응기, 사례 저장기에서 사용자가 사례에 대한 자세한 정보를 요구할 때, 하나의 사례나 사례들의 집합을 선택한 후 계획 브라우저를 호출 한다. 그 후 계획 브라우저에 사례의 내용이 출력되며, Figure 8은 사용자의 호출부터 사례의 내용이 출력되기까지의 프로세스 및 데이터의 흐름을 나타낸다.

조건 검출기(condition extractor)는 사례로부터 초기 조건과 목표 조건을 뽑아 내는 컴포넌트이다. 초기 조건과 목표 조건은 조건 뷰어(condition viewer)를 통해 보여주게 된다.

계획 검출기(plan extractor)는 사례의 내용 중 계획을 검출한다. 검출 된 계획은 동작과 상태로 각각 표현하고 있다. 계획은 일련의 동작들의 집합이므로 검출하는데 큰 무리가 없으나, 상태는 계획을 상태 생성기(state generator)로 거쳐서 만들어지게 된다.

상태 생성기는 최초 동작의 전조건과 초기 목표를 조합하여 초기 상태의 집합을 만들고, 초기 동작 이후의 각 동작에 대한 전조건과 효과를 조합하여 상태를 만든다. 이렇게 만들어진 상태는 동작과 함께 계획 뷰어(plan viewer)에 나타낸다.

또한 상태와 동작 각각에 대한 정보는 한 화면에 보여주기에 정보량이 클 수 있기 때문에 꼭 필요한 정보만 축약해서 화면에 보여준다. 하지만 세부 뷰어(detail viewer)를 통해 사용자가 원하는 동작이나 상태에 대한 자세한 정보를 접근할 수 있도록 하고 있다.

계획 브라우저의 구조를 이와 같이 구성하게 되면 정보의 종류에 따라서 브라우저 내에 정보를 보여줄 수 있게 된다. 각각의 뷰어들이 모두 다른 정보를 전달 하지만 이들은 하나의 사례에 대한 서로 다른 관점에서 보여주고 있다. 따라서 이러한 정보 전달 방법은 사용자로 하여금 사례에 대한 이해를 높이게 된다.

계획 시뮬레이터

Figure 8은 계획 시뮬레이터의 실행화면을 나타내고 있다. 가상의 환경에서 해 계획(solution plan)을 미리 실행해 볼 수 있는 기능을 제공한다. 또한 시스템이 계획을 생성한 후 결과는 사용자가 이해하기 어려울 수 있다. 하지만 시뮬레이터를 실행하게 되면 움직이는 객체로 그것을 확인 할 수 있다. 따라서 사용자의 이해가 높아지고 더욱 높은 품질의 해를 구 할 수 있게 된다. 그러므로 계획 시뮬레이터 모드의 지원은 직접 실행하기 전 사용자에게 판단의 기회를 부여함으로써 사용자에게 선택의 다양성을 보장해준다.

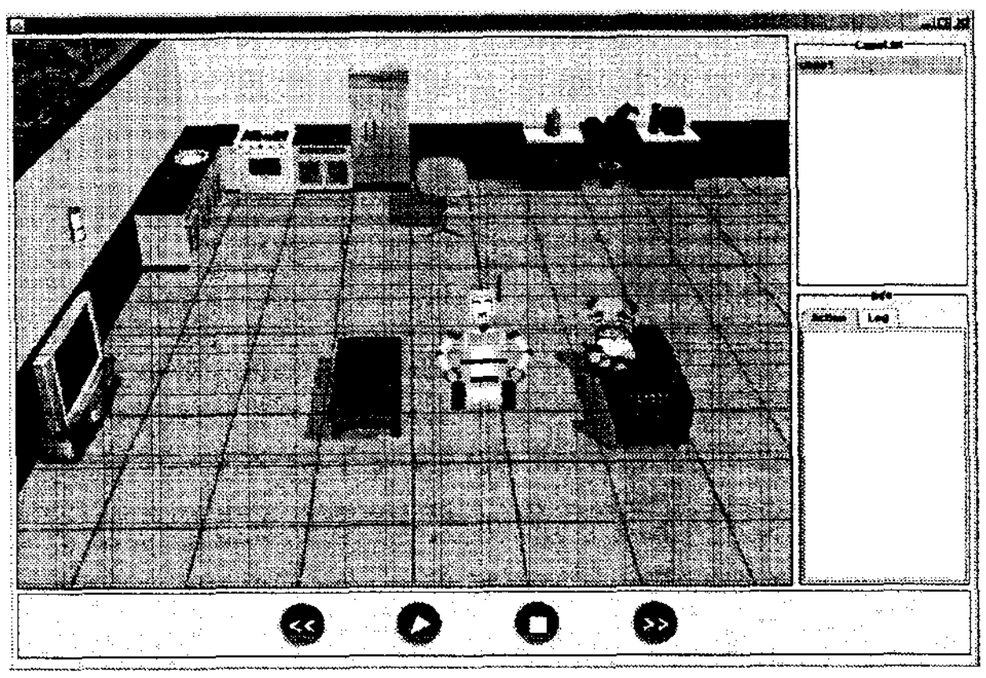

Figure 8 – 계획 시뮬레이터의 실행화면

계획 시뮬레이터가 호출 될 수 있는 시점은 사례 적응 단계를 거쳐 해 계획이 구해진 이후이며, 계획 실행을 위해 이 단계에서 해 계획을 계획 실행기로 보내는 역할도 하게 된다. 사례 적응 단계에서 구한 해가 과연 올바른지 오류가 없는지를 사용자가 알아내기란 어렵다. 하지만 시뮬레이터를 호출하여 구한 해를 미리 시뮬레이션 해보면, 사용자는 문제를 어떻게 해결하는지 직관적으로 쉽게 파악할 수 있다. 시뮬레이션 해본 결과 해 계획이 사용자의 만족도에 미치지 못하면 사용자는 새로운 해를 얻기 위해 앞서 수행한 단계들인 사례 적응단계나 사례 검색단계로 되돌아 갈 수 있다.

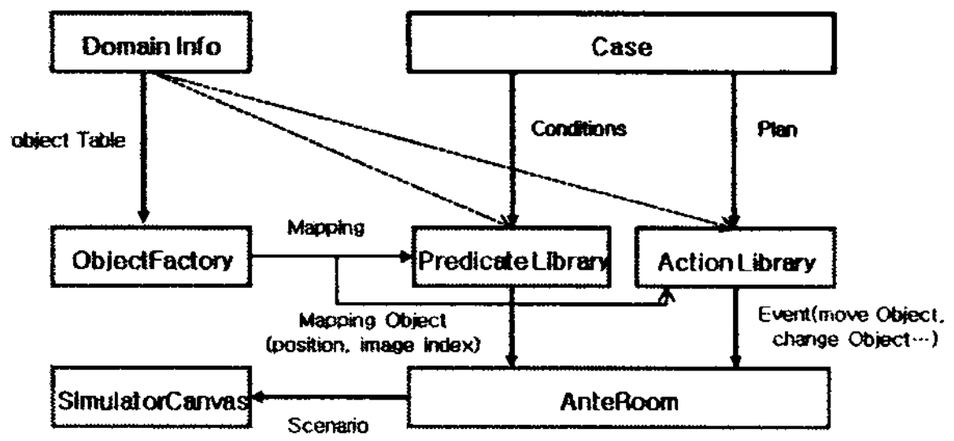

Figure 9 – 계획 시뮬레이터 내부 구조도

Figure 9은 계획 시뮬레이터의 구조도 이다. 객체팩토리(object factory)는 시뮬레이터에서 쓰이는 모든 객체에 대한 정보를 도메인 정보로부터 가져오고, 객체와 그 객체에 대한 타입(type)의 쌍으로 저장한다. 그리고 객체의 타입에 따라 속성이 부여되며, 속성은 이미지에 대한 정보와 객체 사이의 관계 좌표(relation point)를 갖는다.

술어 라이브러리(predicate library)는 도메인 정보를 참조하여, 사례로부터 받은 조건과 객체팩토리에 있는 객체를 이용해 술어 논리 조건식들의 집합을 유지한다. 이러한 술어 논리 조건식들에 따라 객체들의 초기 좌표와 초기 이미지를 맵핑한다. 따라서 술어 논리식들이 시뮬레이터 상에 모든 상태를 표시 할 수 있게 된다.

동작 라이브러리(action library)는 사례로부터 받은 계획을 동작의 순서로 갖고 있는 자료구조이다. 따라서 동작을 수행 하게 하는 이벤트에 대한 정보도 갖고 있다. 객체가 움직이는 일반적인 동작 이외에 2개 이상의 객체가 함께 움직이거나, 동작이 수행되면서 객체가 사라지며, 또한 다른 객체로 대체되는 등의 모든 정보를 포함한다.

대기실(anteroom)은 시뮬레이터의 재생(play) 전까지 화면을 준비하는 구성요소이다. 대기실에서는 초기 상태 모습을 실제 이미지로 보여주기 위해 객체의 좌표를 설정한다. 또한 각각의 객체들은 동작의 순서에 따라 움직이는 동선을 설정하며, 동작들 간에 시간의 지연(delay)등을 설정해주어 시나리오 만들게 된다. 대기실에서는 하나의 사례에 하나의 시나리오가 쌍으로 저장되어 있으며, 사용자가 사례를 선택하면 시뮬레이터 화면에 초기 상태가 표현된다. 재생 버튼을 누르면 시나리오에 따라 실행된다.

시뮬레이터는 사용자가 입력한 사례를 입력 받으면, 그 사례로부터 조건식과 계획을 분석하고 도메인 정보를 참조하여 술어 라이브러리와 동작 라이브러리를 생성한다. 술어 라이브러리를 통한 상태의 표현을 통해 객체를 화면에 출력할 수 있게 되며, 동작 라이브러리를 통해 표현된 객체를 화면상에서 움직일 수 있게 된다. 대기실에서는 사례에 따른 시나리오를 계속 유지시키기 때문에 사용자가 여러 시나리오를 실행시켜보면서 사례를 비교해 볼 수 있는 기회를 제공한다.

또한 Figure 9의 구조도는 사례의 계획이나 조건식이 바뀔 때마다 함께 바뀌는 술어 라이브러리와 동작 라이브러리가 모듈화 되어있으므로 사례에 따라 유연하게 동작한다. 그리고 시뮬레이터의 환경이 바뀌게 되어도 객체의 이미지와 동작의 좌표설정만 바꿔주면 되므로 시뮬레이터 자체가 갖고 있는 의존적 속성을 보완해 줄 수 있다.

실행기와의 연동

JCBP 시스템은 작성된 해 계획의 실행을 위해 계획 실행기와의 연동기능을 제공한다. 계획 실행기의 실행 모드는 수동적 모드(passive mode)와 적극적 모드(active mode) 두 가지를 지원한다.

Figure 8의 (a)는 수동적 모드에서 JCBP 시스템과 실행기의 상호 작용을 시퀀스 다이어그램으로 나타낸 것이다. 그림을 보면, 계획 실행기가 JCBP 시스템에 목표를 주고, 월드 모델에 대한 정보를 갖고 있는 블랙보드시스템(blackboard system)[16]에서 초기 상태를 받아온다. 그리고 목표를 달성할 수 있는 계획을 생성하여 실행기에 넘긴다. 그 후 실행기는 계획에 의해 수행되게 된다.

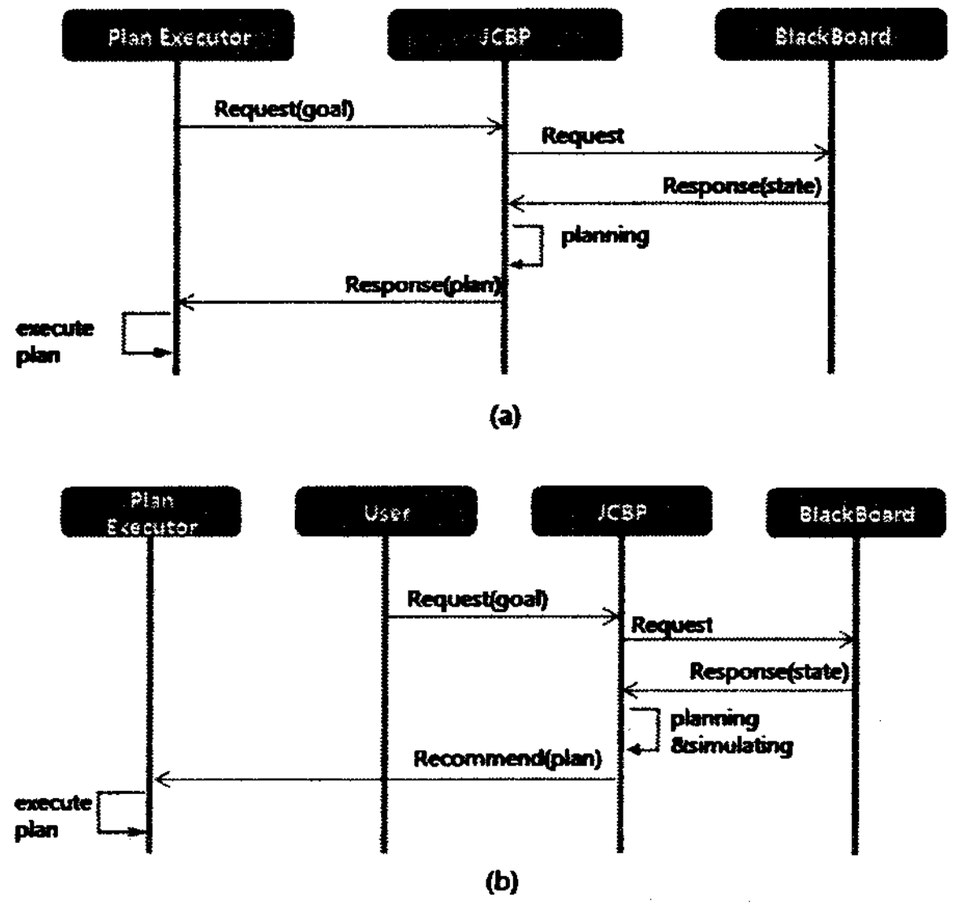

Figure 10 – 실행기와 연동

Figure 10의 (b)는 적극적 모드를 시퀀스 다이어그램으로 나타내었다. 이 모드는 사용자가 직접 목표 조건을 준다. JCBP 시스템은 목표 조건을 받고 그것을 사용자와 함께 해결하며, 구한 해를 실행기에 전달하기 전에 사용자가 그 해의 적합성을 판단할 수 있다. 사용자는 여러 사례를 문제에 적용시키면서 최적의 해를 찾아내고, 또한 시뮬레이터를 사용하여 미리 실행시켜보게 된다. 이러한 과정 이후에 나온 해는 사용자와 시스템 간의 지식과 결정사항을 서로 공유하며 나온 해이므로, 사용자의 만족도가 높은 해가 나오게 된다. 또한 시뮬레이션을 해 본 결과이므로 계획 실행기는 오류율이 적은 안정된 실행을 할 수 있다.

예제

본 절에서는 로봇 작업 계획 생성 예제를 통하여 지금까지 설명한 JCBP 시스템이 어떻게 동작하는지를 소개한다. 도메인은 로봇 환경 도메인이며 서비스 로봇이 가정환경에서 이동하거나 손을 사용함으로써 사용자의 심부름을 수행한다. 계획 문제는 사용자가 로봇에게 '물 좀 가져와라' 는 명령을 목표 조건을 부여했을 경우이다. Table 6은 여러 방이 연결되어있는 집안에서 테이블과 물컵, 정수기 등이 초기 상태로 표현되어있다. 또한 '물 좀 가져와라'는 목표 조건이 '(has user cup)'으로 술어 논리로 표현되어 있다. 사례 기반 계획 시스템은 이 문제를 해결하기 위해 이전의 유사한 사례를 찾을 것이다.

Table 6 – 계획 문제

초기 조건
(movable inroom tablearea) … < 중략 > … (movable purifierarea userarea) (at table tablearea) (at purifier purifierarea) (at circuitbreaker circuitbreakerarea) (position robot tablearea) (position user userarea) (emptyhands robot) (on cup table) (fall_over cup) (emptywater cup) (dark)
목표 조건
(and (has user cup))

또한 Table 6는 계획 문제를 PDDL(Planning Domain Definition Language)로 기술한 모습이다. PDDL은 계획 도메인과 문제를 기술하는 언어로써IPC(International Planning Competitions)에서 2002년도에 제시한 PDDL 2.1 명세에 맞춰 기술하였다. 기술된 계획 문제는 초기 상태의 일부와 목표 조건 전부를 명시하였으며 서비스 로봇이 어두운 환경에서 물이 채워진 컵을 사용자에게 갖다 주는 것을 목표로 하고 있다.

Table 7 – 선택한 사례

초기 조건
(movable tablearea purifierarea) (movable purifierarea userarea) (movable circuitbreakerarea tablearea) (emptyhands robot) (position robot tablearea) (on cup table) (emptyhands robot) (at purifier purifierarea) (at circuitbreaker circuitbreakerarea) (at table tablearea) (fall_over cup) (emptywater cup) (bright)
목표 조건
(has user cup)
계획
0 (upright robot cup table tablearea) 1 (pickup robot tablearea cup table) 2 (carry robot cup tablearea purifierarea) 3 (fillWater robot cup purifier purifierarea) 4 (carry robot cup purifierarea userarea) 5 (give robot user cup userarea)

Table 7은 검색 과정을 통해서 선택한 사례의 초기 조건을 표현하였다. 총 13개의 초기조건 중 12개가 일치하며, 실제 CMR 유사도 측정 방법에 의한 유사도 값은 0.85이다. 주어진 계획 문제가 어두운 환경이라면, 선택한 사례는 밝은 곳에서 사용자에게 물 컵을 가져다 주는 해를 갖고 있다.

Table 8 – 최종 해 계획

0 (moveToCB robot circuitbreaker tablearea circuitbreakerarea) 1 (switchOn robot circuitbreaker circuitbreakerarea) 2 (automove robot circuitbreakerarea tablearea) 3 (upright robot cup table tablearea) 4 (pickup robot tablearea cup table) 5 (carry robot cup tablearea purifierarea) 6 (fillWater robot cup purifier purifierarea) 7 (carry robot cup purifierarea userarea) 8 (give robot user cup userarea)

Table 8은 선택한 사례를 통해 최종 해를 얻은 결과를 보여준다. 0번부터 2번까지가 생성적 계획기를 통하여 새로운 해를 구한 부분이며, 3번부터 8번까지가 기존의 사례를 재사용한 부분이다.

Table 9 – 일반화 결과

초기 조건
(movable tablearea purifierarea) (movable purifierarea userarea) (movable circuitbreakerarea tablearea) (emptyhands robot) (dark) (position robot tablearea) (on cup table) (emptyhands robot) (at purifier purifierarea) (at circuitbreaker circuitbreakerarea) (at table tablearea) (fall_over cup) (emptywater cup)

목표 조건
(has user cup)
계획
0 (moveToCB robot circuitbreaker tablearea circuitbreakerarea) 1 (switchOn robot circuitbreaker circuitbreakerarea) 2 (automove robot circuitbreakerarea tablearea) 3 (upright robot cup table tablearea) 4 (pickup robot tablearea cup table) 5 (carry robot cup tablearea purifierarea) 6 (fillWater robot cup purifier purifierarea) 7 (carry robot cup purifierarea userarea) 8 (give robot user cup userarea)

Table 9는 예제의 일반화된 결과를 보여준다. Table 6를 보면 목표 조건은 (has user cup)로 나와 있다. 이 조건부터 목표회귀 방법을 통하여 일반화 과정을 거치면 Table 6의 총 24개의 초기 상태가 13개의 초기 조건으로 바뀌게 된다. 이러한 초기 조건은 이 사례가 앞으로 쓰일 수 있는 최소한의 조건을 의미한다. 로봇이 물을 사용자에게 가져다 주는 문제에서 정수기의 위치와 테이블의 위치 그리고 스위치의 위치 관계에 대해 직접적으로 이동하는 경로를 제외한 나머지 위치 관계는 모두 제거되었음을 알 수 있다.

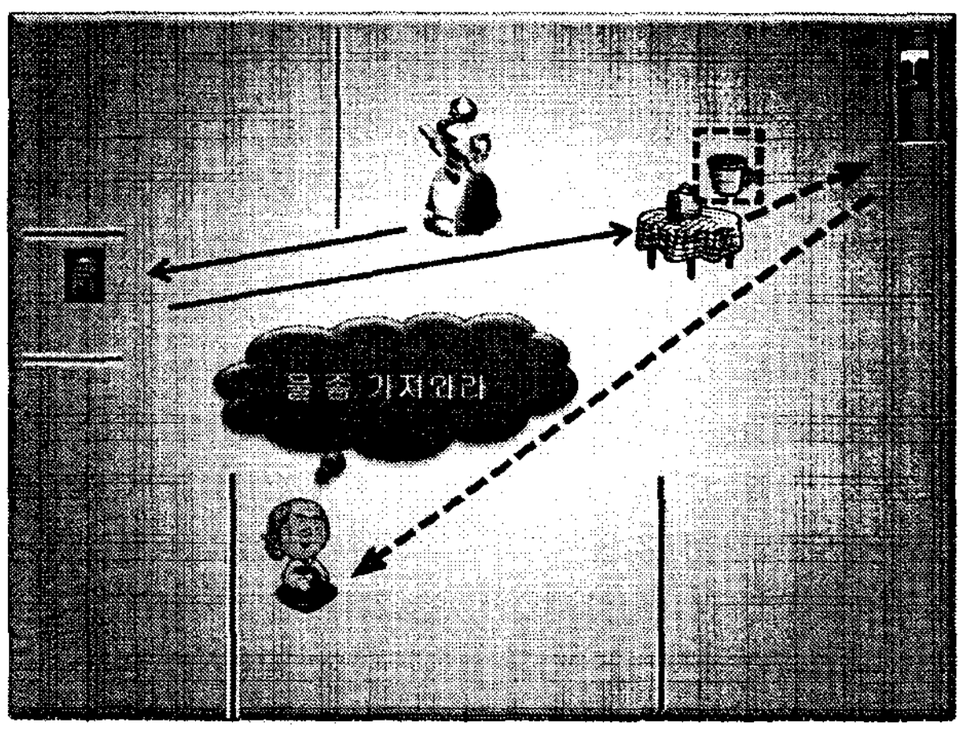

Figure 11 – 서비스 로봇 예제

Figure 11는 본 절에서 다룬 예제를 그림으로 설명하고 있다. 사용자는 서비스 로봇에게 '물을 가져 와라' 는 목표를 주었으며, 로봇은 이것을 해결하고자 한다. 로봇은 점선으로 표시한 부분인 물을 떠다 주는 이미 갖고 있는 지식을 그대로 사용하고 있다. 그리고 그것을 달성하기 위한 어두운 환경에서 전원 스위치를 올려 밝은 환경으로 바꿔주는 계획을 작성한 부분을 실선으로 표시하였다.

실험 및 평가

JCBP시스템을 이용한 계획의 품질과 그 효율성을 입증하기 위해서 몇 가지 실험을 해 보았다. 우선 계획 과정의 효율성을 위해서 계획 문제를 해결하는데 소요한 시간을 측정하며 JCBP 전체과정, JCBP 적응 단계, 생성적 계획기의 경우를 비교 분석한다. 또한 적용 사례의 유사도에 따른 소요시간의 변화도 분석해서 그 효율성을 입증하고자 한다. 두 번째는 계획의 품질에 대한 타당성의 검증으로 계획 문제의 해 계획의 길이를 측정하여 생성적 계획기와 비교하고 최적성을 판단한다.

실험에는 총 3가지의 서로 다른 영역의 문제들이 사용되었으며 각 영역 당 3개의 계획 문제를 실험에 적용시켰다. 실험에 사용된 3가지 영역은 4개의 블록(block)을 갖고 테이블 위에서 문제를 해결하는 blockworld와 서비스 로봇의 영역인 trobot, 마지막으로 화재 발생 시 로봇이 취해야 할 행동을 계획하는 irobot이다. blockworld는 행위(operator)의 수가 적고 문제가 간단학고 직관적이다. 따라서 사용자가 문제를 쉽게 변형하며 실험이 가능하며 여타 계획 생성 시스템에서도 실험을 위해 많이 쓰이는 도메인이다. trobot 도메인은 사용자와 로봇 간에 일어날 수 있는 서비스에 대해 기술하였다. 본 논문에서 제시한 사용자와 로봇간의 대화를 가장 잘 나타낼 수 있는 도메인이며, 로봇에게 주어지는 목표 조건은 사용자가 가정환경에서 직면하게 되는 일상적인 문제가 주어진다. irobot 도메인도 사용자와 로봇간의 문제에 대해 기술하였다. 하지만 일상적인 생활보다 우발적 상황에 직면하였을 때의 상황을 잘 나타낼 수 있는 도메인이다. 예를 들면 화재 발생 시 로봇이 어떻게 행동하는가에 대한 문제를 들 수 있다.

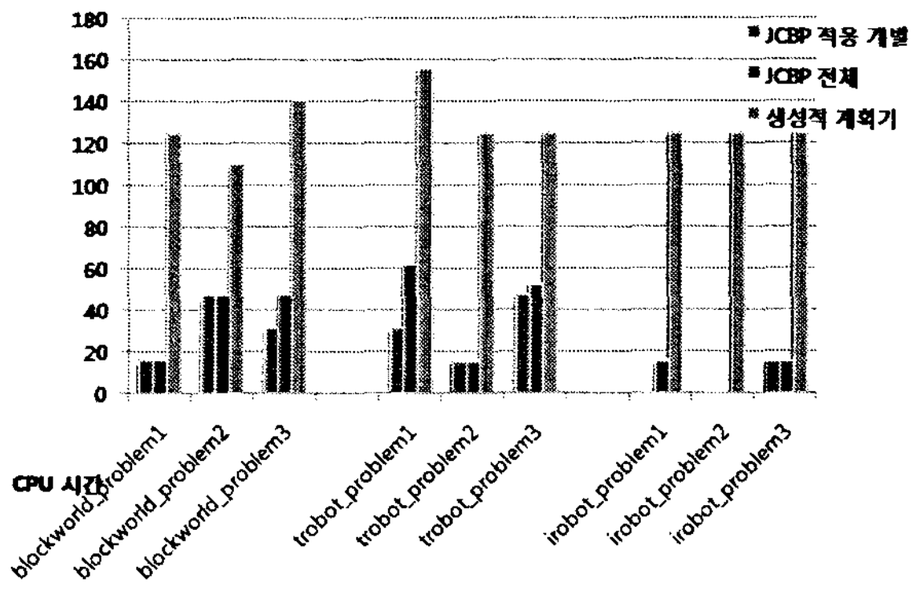

Figure 12 – 문제 별 소요시간

Figure 12은 각 도메인에 해당하는 계획 문제마다 문제를 해결하는 시간을 측정하여 그 결과 값을 나타낸 것이다. JCBP 시스템이 생성적 계획기 보다 일반적으로 우수한 성능을 나타냄을 알 수 있다.

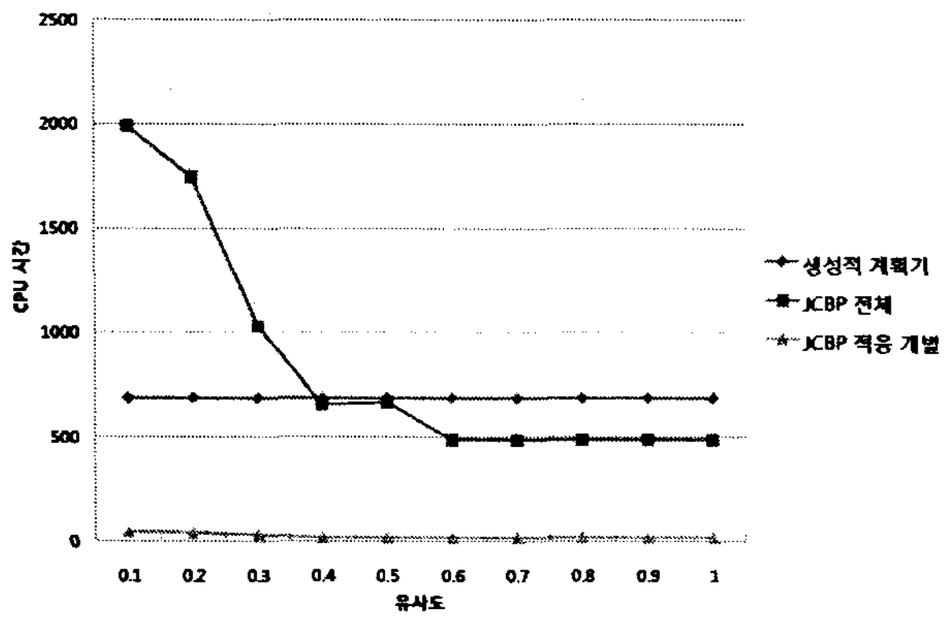

Figure 13 – 문제의 유사도에 따른 소요시간

Figure 13은 blockworld 도메인으로 본 시스템을 실행하였을 때의 결과를 나타낸 것으로 JCBP 시스템 전체 및 적응 개별 시간 그리고 생성적 계획기을 사용하여 문제를 해결 하면서 유사도에 따른 시간을 측정하여 그래프로 나타내었다. 본 실험 결과를 통해 알 수 있는 사실은 유사도 값을 높게 주면 검색된 사례의 개수가 줄어든다. 따라서 전체적으로 봤을 때 문제를 해결하는 시간이 줄어드는 것을 확인할 수 있다. 따라서 본 논문에서 제시한 유사도 측정 방식의 타당성을 입증한다고 볼 수 있으며, 적당한 유사도의 한계치를 부여함으로써 시스템 자체 성능을 높일 수 있음을 보여준다

Table 10 – 해 계획의 길이

문제	JCBP	최적 해
blockworld_1	5	3
blockworld_2	5	3
blockworld_3	7	5
trobot_1	7	6
trobot_2	7	6
trobot_3	11	8
irobot_1	3	3
irobot_2	4	2
irobot_3	5	3

Table 10은 유사도 값에 따른 JCBP시스템과 생성적 계획기 각각에 따른 행위의 수이다. 이 값이 의미하는 것은 문제를 해결하고 난 후 결과물의 값으로써 계획의 길이를 뜻한다. 생성적 계획기는 문제를 해결하면서 항상 최적의 해만을 제공하는 알고리즘으로 구현되어있다. 따라서 생성적 계획기에서 나온 값은 가장 최적화된 값이다. JCBP 시스템의 값과 비교를 해보면 일반적으로 JCBP 시스템의 값이 크게 나오는데 이것은 저장된 사례를 사용하게 됨으로써 나오는 결과이다. 하지만 최적화된 결과 값인 생성적 계획기의 값과 비교해 보면 크게 차이가 없는 것을 알 수 있다. 따라서 JCBP 시스템은 문제를 해결하면서 최적화 값과 가깝게 도출해 내며 Figure 13과 함께 결과를 종합해 볼 때 JCBP 시스템은 합리적인 시간 안에 원하는 결과를 출력하므로 사례 기반 계획 시스템으로써 충분한 가치가 있음을 보여준다.

관련 연구

지금까지 설명한 JCBP 시스템을 기존의 사례 기반 계획 시스템들과 비교하여 특징을 분석해본다. 특히 시스템의 특징을 구분 짓는 요소는 사례 표현 방법, 사례 검색 방법, 사례 적응 방법의 3가지가 가장 중요하다고 판단된다. 따라서 이 요소에 따라서 JCBP 시스템과 타 시스템과의 유사점 및 차이점을 알아본다.

사례 표현 방법은 일반적으로 그 표현 방식에 따라 변형적(transformational) 표현 및 유도적(derivational) 표현방법으로 나눌 수 있다. JCBP 시스템은 사례를 변형적으로 표현하였으며, 변형적 표현은 메모리 내에 계획의 결과인 동작과 하위 목표(sub-goal)을 모두 저장한다. CHEF[3][4], DIAL[5] 등의 시스템도 변형적 표현 방법을 사용하였다. 유도적 표현 방식의 시스템은 DERSNLP[6], SPA[7] 등이 대표적이며, 변형적 표현 방법과는 다르게 모든 계획 수행 중 발생했던 결정 사항과 계획을 생성하면서 관련된 하위 목적과 그 과정 등을 모두 포함하고 있다. 따라서 유도적 표현 방법이 저장 공간을 더 많이 차지하고 있지만 더욱 풍부한 정보를 갖고 있다.

JCBP 시스템의 사례 검색 방법은 유사도 측정 방식이다. 유사한 시스템으로는 HICAP[9], PRIAR[10] 등이 있다. 반면 계층 비교 방식의 시스템은 DERSNLP와 PARIS[8] 등이 대표적이다. 계층 비교 시스템의 특징은 사례를 계층구조로 표현하였을 경우에만 사용 가능하다. 계층 구조의 일반화 수준(level)에 따라서 계층이 이루어지며, 사례의 특징을 추상화(abstraction)/세분화(specialization)를 통해서 일치시키며 비교한다.

JCBP 시스템은 적응 방법으로 휴리스틱 기반 방법을 사용한다. 이 방법은 대부분의 시스템에서 적용하는 방식이며 생성적 계획기에 적용 가능할 경우에만 사용한다. 또 다른 사례 적응 방법은 유도 반복(derivational replay)방법이 있다. 이 방법은 현재의 문제를 해결하기 위해 새로운 문제를 원래의 계획으로부터 단계별로 유도시키며 생성하는 방식이다. 일반적으로 도메인에 독립적이지 않으며 일반적으로 계획 생성기를 사용해야만 한다. 이 방식을 사용한 시스템의 예로는 DERSNLP와 DIAL이 있다. 이 밖에 계획 생성 적응 방식도 있으며 이것은 다른 방식이 적응과정을 수행하지 못할 경우 사용된다. 계획을 하나씩 세워 나가면서

기존의 문제를 수정하고 그에 따라 목적에 도달하도록 적응해 나간다. 계획 생성 적응 방식을 사용한 시스템으로는 PRODIGY/ANALOGY[11]와 PARIS가 있다.

결론

본 논문에서는 기존의 사례 기반 계획 시스템의 부족한 점을 극복하고자 JCBP 시스템을 제안하였다. JCBP 시스템은 효율적인 메모리사용과 데이터베이스관리를 위해 도메인과 사례를 그룹화하여 관리하고 색인 정보를 유지한다. 또한 휴리스틱 지식을 사용한 생성적 계획 적응 방법을 사용하였다. 사례를 저장은 목표 회귀 방식으로 일반화 과정을 거친 후 실행한다. 또한 사용자의 지식을 활용한 계획을 생성하기 위해서 대화형 모드를 지원하는 GUI기반의 진행 방식을 갖고 있다. 또한 본 논문은 예제를 통해서 JCBP 시스템의 진행과정을 자세히 기술하였고, 다른 시스템과의 비교를 통해 객관적인 관점에서 평가하였다. 끝으로 실험을 통해서 JCBP 시스템의 성능 및 효율성에 대해 논했는데, 실험 결과를 보면 최적화 정도나 해를 얻기까지 측정된 소요 시간의 비교로 보아 본 시스템의 성능을 입증할 수 있었다. 향후 본 시스템은 사용자와 시스템이 서로의 지식을 공유하며 계획을 생성해 나갈 수 있을지 더욱 연구해야 할 것이며, 사례 검색 방법과 적응 방법의 다양화를 통해 보다 좋은 알고리즘을 찾도록 연구할 계획이다.

References

[1] Flavio Tonidandel, Marcio Rillo. (2001). "An Efficient Case-Based Planning System:Preliminary Results," 5o. SBAI - Simpósio Brasileiro de Automação Inteligente.

[2] Luca spalazzi. (2001). "A Survey on Case-Based Planning," Artificial Intelligence Review vol.16, pp.3-36.

[3] Hammond. (1898). "Case-Based Planning : Viewing Planning As A Memory Task," Boston : Academic press.

[4] Hammond. (1990). "Explaining and repairing plans that fail," Artificial Intelligence. Vol.45, pp.173-228.

[5] Leake. D, Kinley. A Wilson. D. (1997). "A Case Study of Case-Based CBR," Proceedings of the Second International Conference on Case-Based Reasoning(ICCBR-97), pp.371-382.

[6] Ihrig, L. H. & Kambhampati, S.(1997). "Storing and Indexing Plan Derivations Through Explanation-based Analysis of Retrieval Failures," Journal of Artificial Intelligence Research, Vol.7, pp.161-198.

[7] Hanks, S. & Weld, D. (1995). "A Domain-Independent Algorithm for Plan Adaptation," Journal of Artificial Intelligence Research. Vol.2, pp.319-360.

[8] Bermann, R. & Wilke, W. (1995). "Building and Refining Abstract Planning Cases by Change of Representation Language," Journal of Artificial Intelligence Research, Vol.3, pp.53-118.

[9] Muñoz-Avila, H., Hendler, J. A. & Aha, D. W. (1999). "Conversational Case-Based Planning," New Review of Applied Expert Systems, Vol.5.

[10] Kambhampati, S & Hendler, J. (1992). "A Validation-structure-based Theory of Plan Modification and Reuse," Artificial Intelligence vol.55, pp.193-258.

[11] Veloso, M., Carbonell, J., Pérez, A., Borrajo, D., Fink, E. & Blythe, J. (1995). "Intergrating Planning and Learning: The PRODIGY architecture," Journal of Experimental and Theoretical Artificial Intelligence. Vol.7, no.1

[12] Coupey, P., Fouquere, C. & Salotti. (1998). "S. Formalizing Partial Matching and Similarity in CBR with a Description Logic," Applied Artificial Intelligence, vol.12, no.1, pp. 71-112.

[13] Daniel D. Corkill. Blackboard Systems. AI Expert, 6(9):40-47, September, 1991

[14] Avrim L.Blum, Merrick L.Furst (1997). "Fast Planning Through Planning Graph Analysis," Journal of Artificial Intelligence, vol.90, pp.281-300.

[15] Alfonso Gerevini, Ivan Serina. (1999) "Fast Planning through Greedy Action Graphs", in Proceedings of Sixteenth National Conference of Artificial Intelligence (AAAI-99), AAAI-MIT Press, Orlando, Florida, USA.

[16] Daniel D. Corkill, Kevin Q. Gallagher, and Kelly E. Murray. (1986) "GGB: A generic blackboard development system", in Proceedings of the National Conference on Artificial Intellgence, pp.1008-1014