論文 89-26-2-9

다중로보트 시스템을 위한 로보트 언어 개발에 관한 연구

(A Study on the Development of Robot Language for Multi-Robot System)

朴 鍾 憲*, 張 哲*, 崔 秉 旭*, 鄭 明 振*

(Jong Hun Park, Cheol Chang, Byoung Wook Choi and Myung Jin Chung)

要 約

산업기술의 발달로 조립현장에서는 특수한 기능을 갖는 센서와 투울들을 정착한 지능로보트들이 현재 많이 이용되고 있다. 이러한 지능 로보트들을 사용하는 자동 제조 시스템들이 보다 발전되고 복잡해짐에 따라 작업장내에 있는 여러 로보트들 뿐만 아니라 주변 장치들을 동시에 제어할 수 있는 보다 발전된 형태의 로보트 언어에 대한 필요성이 점증하고 있다. 본 논문에서는 제조 공정내에 존재하는 여러 로보트들 뿐만 아니라 주변 장치들을 동시에 제어할 수 있는 새로운 형태의 로보트 언어, ARL (assembly robot language)을 제안하고 이를 구현하였다. 본 언어 시스템에서는 기존 로보트 언어들이 갖는 공통적인 기능들과 디버깅 기능 이외에 대부분의 기존 언어들이 하드웨어 종속적인 점을 감안하여 별도의 전담 프로세스와 프로세스간 통신을 위한 공통 메모리를 이용하여 로보트나 주변장치에 종속적인 요소들을 최소화 하도록 하여 확장성이나 환경변화에 따른 적응성이 좋도록 하였다.

Abstract

Many intelligent robots that are equipped with special tools and sensors are currently used in assembly line. As automatic manufacturing systems including such robots become advanced and complicated, there are increasing needs for the development of the sophisticated programming systems which can control several robots and other manufacturing equipments in workcell at a time. In this paper a programming language, ARL (Assembly Robot Language), is proposed and developed, which can control the manufacturing devices as well as robots in workcell. It has not only all the common features of modern textual robot language but also debugging facilities. In this language system machine dependecy is minimized by using dedicated processes and a shared memory for communication between processes. Extensibility and adaptability of the programming system is increased by using such a technique against the changes of workcell environment.

I. 서 론

60년대초 로보트가 처음으로 die casting을 위한 작업에 사용된 이래 로보트의 이용은 단순한 spot welding 이나 arc welding, material handling 등과 같은 비교적 간단한 작업으로 부터 조립작업이나 제품검사와 같이 복잡한 작업에 이르기까지 그 이용 분야는 날로 넓어지고 있다.[1] 초기에는 주로 인간에게 유해한 작업환경으로 부터 작업자를 보호하기 위하여

*正會員, 韓國科學技術院 電氣 및 電子工學科
(Dept. of Electrical Eng., KAIST)

接受日字 : 1988年 5月 24日

로보트가 이용되었으나 로보트 관련기술의 발달로 로보트 능력이 고급화되고 가격이 저렴해지면서 현재는 경제적 채산성 및 생산성 향상을 위한 로보트 이용이 점차 증가하고 있다.[1~2]

로보트의 가장 두드러진 특징은 Robot Industries Association에서 정의한 바와같이 "재 교시가 가능한 범용 머니플레이터"[1] 라는 데 있다. 이와 같은 범용 머니플레이터인 로보트를 이용함에 있어 무엇보다 중요한 것은 '주어진 작업을 수행하기 위하여 어떻게 로보트를 효과적으로 교시할 것인가' 하는 것이다. 현재 널리 사용되고 있는 교시 방법으로는 작업내용을 작업자가 직접 로보트팔을 끌고 다니며 교시하거나 교시상자를 이용하여 교시하는 leadthrough 방식이 있으나 작업 내용이 복잡해짐에 따라 교시내용이 단순히 로보트의 경로만을 나타내는 이외에 복잡한 산술계산을 하거나, 센서로부터 받은 정보나 계산 결과에 따라 판단을 하거나, 또는 다른 computer based system과 정보를 교환하는 등의 기능이 중요해짐에 따라 이러한 교시방법이 한계에 부딪히게 되고 컴퓨터 언어와 유사한 형태의 textual language를 이용한 교시방법이 점차 그 중요성을 더해가고 있다.[1~2]

산업사회의 발달로 인하여 로보트이용이 매년 급격이 증가하고 있을 뿐만아니라 그 응용 범위도 날로 확대되면서 보다 많은 지능을 갖는 로보트를 필요로 하고 있다.[1~2] 일반적으로 자동 제조공장의 경우 여러 로보트들이 작업장내에 있는 센서나 공작기계들과 같은 외부장치들과 서로 밀접한 관계를 갖고 작업을 수행하게 되므로 기존의 stand alone 방식을 위한 대부분의 로보트언어들로는 전체적인 공정을 효과적이고 편리한 방법으로 제어하거나 배치하는데 어려움이 많으므로 작업장내에 있는 여러장치들을 전체적으로 제어할 수 있는 로보트언어가 필요하다.[4]

로보트의 지능은 불확실한 주변환경에 적응하기 위하여 센서나 도구들을 어느 정도 이용할 수 있는가에 달려 있다.[1~2] 그러나 그들의 이용은 관련 하드웨어에 따라 매우 다르게 되므로 일률적으로 정하기 어려워 기존 로보트 언어들의 경우 센서나 도구들을 매우 한정적인 상황에서만 사용할 수 있도록 하고 있으며 그 적용범위도 극히 제한되어 있어 이들을 바꾸거나 추가하기 어렵다. 따라서 주변환경의 변화에 쉽게 적응할 수 있는 기능이 로보트언어에 필요하다.

1. 기존 로보트 언어의 고찰

지금까지 발표된 로보트 언어들은 대부분 특정 로보트 시스템만을 위한 것으로 어림하여 100여 종류에 이르고 있다. 이와 같이 많은 종류의 로보트 언어가 생기게 된 데는 기존의 로보트 언어들이 특정 시스템에만 적합하게 되어 있어 다른 시스템을 위해서는 부득이 그에 적합한 새로운 언어가 필요했기 때문이었다. 따라서 세계적으로 많은 중복 투자가 이루어지고 있어 구미 여러나라에서 로보트 언어의 표준화를 위한 노력을 하고 있으며 독일의 경우 IRDATA (industrial robot DATA)를 표준언어로 지정하여 사용하고 있다.[4]

다중로보트 시스템을 위한 로보트 언어 개발에 관한 연구는 1974년 스탠포드 대학에서 AL에 대한 연구를 시작으로 1981년 멕도널 더글라스사에서 오프라인 프로그래밍을 위하여 NC 언어인 APT를 확장시켜 개발한 MCL과 1982년 General Electric사에서 개발한 HELP 등이 있으나 국내에서는 아직 초보적인 연구단계에 머무르고 있는 상태다.[7~8,10~11]

2. 새로운 언어에 대한 설계원칙

본 논문에서는 앞서 지적한 바와같이 발전하는 로보트 산업의 요구에 부합하기 위하여 여러종류의 로보트들과 센서, 공작기계들로 이루어지는 작업장을 효과적으로 제어할 수 있도록 하며, 로보트 주변에 있는 센서나 공작기계들과 같은 외부장치의 하드웨어에 종속적인 점들을 적도록하여 로보트를 포함한 작업장내의 환경변화에 쉽게 적응할 수 있도록 하였으며, 로보트 교시의 편리와 앞으로 오브젝트 레벨 언어로의 확장을 고려하여 조인트 공간에서 뿐만 아니라 structured Cartesian coordinate에서 로보트 프로그래밍[15]을 할 수 있도록 하는 한편, 사용자가 프로그램을 작성하거나 작성후 이해하기 용이하도록 기존의 컴퓨터 언어와 유사한 structured programming language가 되도록 하고, 프로그램 개발의 편의를 위하여 프로그램의 확인 및 수정이 용이하도록 하였다.

Ⅱ. ARL 시스템의 구조

로보트 프로그래밍 시스템은 일반적으로 그림 1에서와 같이 작업장내에 존재하는 로보트나 센서, 공작기계등과 같은 실제 시스템들 이외에 이들을 효과적이고도 쉽게 다룰 수 있는 로보트 언어와 프로그래밍 투울(tool), 언어 해석기, 월드 모델(world model)및 월드 모델러, 시뮬레이터 등과 같은 여러 모듈들로 구성된다.

사용자는 주어진 작업을 수행하기 위해 필요한 처리과정들을 로보트 언어와 월드 모델을 이용하여 프로그래밍하고 편집기를 통하여 시스템 내부에 입력시킨후 프로그램을 번역하게 되면 그 번역기에 의해 작업장내에 로보트를 포함한 여러장치들이 제어되게

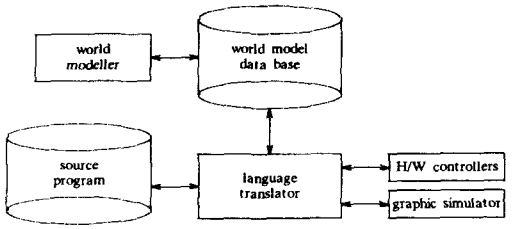


그림 1. 일반 로보트 프로그래밍 시스템의 구조
Fig. 1. Structure of general robot programming system.

되며, 필요한 경우 사용자는 디버깅 도구를 이용하여 잘못된 부분을 수정하게 된다. 이와는 별도로 사용자는 월드 모델러를 이용하여 그래픽 화면상에서 CAD 데이타나 사용자의 입력 내용에 따라 작업장과 그 내에 존재하는 로보트 등을 모델링하게 되며 이는 차후 프로그래밍 과정에서 사용되거나 변경되게 된다.[2,4]

1. ARL의 하드웨어 구조

로보트 언어를 설계하기 위해서는 그 언어에 의해 표현될 대상이 무엇인지를 우선 명백히 하여야 할 필요가 있다.

ARL에서는 그림 2 와 같이 작업장의 구성 요소들을 여러종류의 로보트들과 용접기나 feeder와 같은 주변장치들의 제어기, 작업장 환경을 궤환시켜 주기 위한 센서 시스템들로 보았으며, 로보트 언어가 하드웨어 종속적인 점들을 최소화하고, 환경의 변화에 쉽게 적응할 수 있도록하기 위하여 이들 각 제어장치들을 master로 부터 명령을 받아 독자적으로 그 명령을 수행할 수 있는 기능을 갖는 가상머신(virtual machine)들로 보았다.

ARL에서는 이들 각 가상머신들을 전체적으로 제어하기 위한 supervisory controller로 빠른 처리속도와 네트 워크 기능을 갖고, 범용 운영체계인 UNIX를 탑재한 Sun-3/110 워크 스테이션을 사용하였다.

2. ARL의 소프트웨어 구조

ARL에서는 기본적으로 로보트를 포함한 여러 제어기들을 별도의 독립적인 프로세스로 보고 다중프로세스 처리기법을 이용하여 주어진 작업을 수행하도록 하고 있으며, 프로세스간 정보교환을 위하여 별도의 공통 메모리(shared memory)를사용하였다. 또한 프로세스간 동기를 맞추거나 critical section 내에서 프로세스간 mutual exclusion을 위하여 일반적으로 여러 시스템들에서 많이 사용하고 있는 방법중 하나인 semaphore를 사용하였다.[5]

ARL 시스템에서 존재하게 되는 프로세스들로는 그림 3 과 같이 로보트에 할당된 작업들을 수행하기 위한 로보트 프로세스, 외부 장치들과 로보트 언어를 인터페이스 시켜주기 위한 찬넬 프로세스(channel process), 작업장내에서 발생될지도 모를 위험상황 등을 감시하거나 guarded motion을 위한 감시전용 프로세스, 작업장의 변화를 그래픽 화면상에 보여주기 위한 그래픽 프로세스[12] 들로 구성된다.

로보트 프로세스는 사용자가 프로그램내에서 지시한 내용에 따라 해당 로보트작업을 전담하여 처리하기 위한 프로세스로 사용자가 지정한 수만큼 동시에 존재하게 된다. 각 로보트 프로세스는 독자적으로 공간을 할당받아 사용하게 되므로 상호간섭 없이 작업을 수행하게 되며 사용자는 별도의 명령을 이용하여 로보트의 종류와 홈의 위치(home position)를 지정할 수 있다. 현재 ARL에서 사용 가능한 로보트 종류로는 국내에서 교육용 로보트로 널리 사용하고 있는 Puma-560과 Rhino-XR, 용접용 로보트로 대우중공업에서 생산하는 Nova-10이 있다.[12~13]

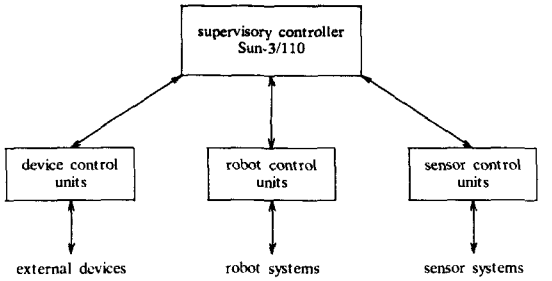


그림 2. ARL 시스템의 하드웨어 구조
Fig. 2. Hardware structure of ARL system.

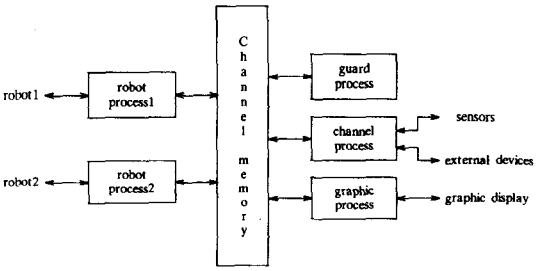


그림 3. ARL시스템의 소프트웨어 구조
Fig. 3. Software structure of ARL system.

로보트 프로그래밍을 위해서는 로보트 뿐만 아니라 그 주변 장치들의 상태 파악과 제어가 쉽게 이루어질 수 있어야 한다. ARL에서는 별도로 찬넬 프로세스를 두어 로보트 작업장내에 존재하는 여러 장치들을 글로벌 변수(global variable)로 추상화 시켜줌으로써 이들을 쉽게 이용할 수 있음은 물론 새로운 센서나 장치들을 추가하거나 변경할 경우, 이 부분만을 수정하면 되므로 환경변화에 쉽게 적응할 수 있다.

작업을 수행하다 보면 여러 원인에 의해 로보트가 다른 작업물과 충돌하거나 하는 위험상황이 발생할 수 있으며, guarded motion을 할 경우 작업장내에 있는 작업물과의 접촉정도가 어느 정도인지 계속 감시할 필요가 있어 별도의 감시전용 프로세스를 두고 있다.[2] 사용자는 프로그램내에서 명령을 통하여 감시하여할 내용들을 감시 프로세스에 등록하게 되며, 감시 프로세스는 등록된 조건들을 다른 프로세스의 작업과는 별도로 계속 감시하다 등록된 조건에 해당하는 사건이 발생할 경우 해당 프로세스에 인터럽트를 발생시킨다.

다중 프로세스 기법중 각 프로세스간 정보교환과 프로세스간 충돌회피, 프로세스간 동기 등이 매우 중요하게 된다. ARL에서는 각 프로세스간 정보교환을 위하여 특정 메모리공간을 각 프로세스들이 공유하도록 하고, 이를 100개의 셀(cell)로 세분하여 각 셀들을 정보교환 찬넬로 사용하고 있으며, semaphore를 이용하여 각 찬넬을 단지 한 프로세스만 사용할 수 있고 나머지 프로세스들은 먼저 사용중인 프로세스가 끝날 때까지 기다리게 함으로써 프로세스간 충돌을 피하도록 하고 있다.[5]

Ⅲ. ARL 언어

로보트 언어 발달 과정에서 살펴본 바와 같이 로보트 언어의 일반적인 특징중 하나가 그것이 단순히 로보트의 경로를 교시하거나 작업장내에 있는 여러 장치들과의 상호 작용을 기술하기 위한 명령들 이외에 범용 컴퓨터 언어와 유사한 제어 구조나 자료형, 연산자들을 갖는다는 점이다.[2,6]

RPL이나 RCCL 등과 같은 경우 기존의 범용 컴퓨터 언어에 로보트 프로그래밍에 필요한 함수나 자료구조 등을 첨가하여 사용하고 있는 경우도 있으나[7~8,15] 본 논문에서는 표현 방법이 간략하고 풍부한 연산자를 갖고 있으며, 날로 사용자가 늘고 있는 C 언어에 기반을 둔 새로운 형태의 머니플레이터 레벨 로보트 언어를 다음과 같이 설계하였다.

1. 로보트 제어 명령

ARL에서는 별도의 로보트를 지정하지 않는한 로보트가 한 대인 것으로 간주되나 여러 로보트를 동시에 제어하고자 할 경우 사용자는 각 로보트마다 고유번호와 함께 그 로보트의 작업을 교시하여야 한다. 이 때 사용자는 필요하다면 프로그램상에서 로보트의 종류와 홈의 위치를 별도로 지정할 수 있다. ARL에서는 로보트마다 각기 독립된 변수들을 사용할 수 있으므로 다른 로보트의 간섭없이 독립적으로 작업을 수행할 수 있으며, 필요한 경우 interlock을 위한 명령들을 이용하여 로보트간 정보를 교환하거나 동기를 맞출 수 있다.

이상과 같은 로보트 제어 명령들을 BNF(backus naur form)로 표시하면 다음과 같다.

```
statement : =with Rob (robot_no) statement
          | define robot robot_name;
          | defin home home_position;
          | move_id to position_list;
          | move_id to position_list while
            (guard_conditions);
          | speed expression;
          | home;
          | other_statements
move_id   : =jmove | lmove | cmove | smove
```

로보트 교시내용중 어떻게 위치관련 정보를 효과적으로 정확하게 교시할 수 있는가가 매우 중요하다. 이를 위해 ARL에서는 로보트의 조인트 공간에서 뿐만아니라 기준 좌표계를 중심으로한 여러 중간 좌표계와 그에 따른 상대 위치들을 지정할 수 있도록 하고 있으며 위치교시를 보다 편리하게 할 수 있도록 메트릭스 연산자와 위치를 나타내기 위한 여러 함수들을 표 1과 같이 제공하고 있다.

표 1. 위치교시를 위한 함수

Table 1. Functions for teaching position and orientation.

종 류	함 수	내 용
orientation	RPY(ϕ, θ, ψ) Euler(ϕ, θ, ψ) Rot (axis, θ)	roll, pitch, yaw Euler angles rotate w. r. t axis θ deg. axis is x, y, or z
position	Trans (x, y, z) Cyl (z, α, r) Sph (α, β, γ)	Cartesian coordinates cylindrical polar coordinates spherical polar coordinates
joint	Joint (Θ)	joint coordinates

2. 작업장 제어 명령

작업장을 제어하기 위한 명령들로는 작업장내에 있는 장치들을 제어하거나 현재의 상태를 파악하기 위한 channel expression과 로보트나 가상 머신들간에 서로 동기를 맞추기 위한 interlock 명령, 용접기를 제어하기 위한 용접 명령들이 있다.

앞서 기술한 바와 같이 ARL에서는 작업장내에 존재하는 장치들을 변수로 (찬넬) 추상화 시켜 사용하고 있으므로 이들의 현재 상태를 알기 위해서는 해당 찬넬을 읽어보면 알 수 있고 또 제어하기 위해서는 해당 찬넬에 명령을 주면 된다. 로보트 작업환경이 복잡해 질수록 더욱 중요해지는 것이 각 장치들간에 동기를 맞추는 일이다. ARL에서는 channel expression과 wait 명령을 이용하여 비동기적인 방법으로 정보를 교환하고 있으며, 동기가 필요한 경우 send/receive 명령을 이용하도록 하고 있다. 이와는 별도로 머니플레이터의 동작을 일정시간 멈추도록 하기 위한 pause 명령을 제공하고 있다.

앞으로 예상되는 로보트 이용중 중요한 위치를 차지하게 될 부분중의 하나가 아크 용접이다. 대부분 용접용 로보트의 경우 용접용 팁(tip)을 용접 부위를 따라 움직이도록 하기 위한 머니플레이터 제어기와 용접 과정을 제어하기 위한 용접 제어기로 분리되어 있으며 용접기 제어는 여러 파라미터들에 의해 이루어지고 있다.[14] 따라서 용접을 위한 실제 과정은 매우 복잡하나 사용자는 필요한 파라미터값을 미리 정하고 용접 부분을 따라 로보트를 움직이면 되므로 쉽게 사용할 수 있도록 되어 있다.

ARL에서도 용접을 위하여 용접용 제어기의 파라미터 값들을 정하기 위한 명령과 용접의 시작과 끝을 표시하기 위한 명령들을 다음과 같이 마련하고 있다.

```
statement : =set_welding_parameter schedule_no;
                {welding_conditions}
          | start_weld schedule_no;
          | stop_weld;
welding_conditions
          : =welding_parameter=expression;
          | welding_conditions weiding_para-
            meter=expression;
```

3. 선언문

현재 ARL에서 제공하고 있는 자료형으로는 정수형 변수를 선언하기 위한 int, 실수형 변수를 선언하기 위한 real, 문자나 바이트 단위의 변수를 선언하기 위한 char, 로보트의 위치나 좌표계를 나타내기 위한 frame이 있으며 자료 구조로는 배열만 제공하고 있다.

4. 연산자

로보트의 작업 환경이 보다 복잡해 질수록 필요한 자료를 정리하거나 외부 장치들로 부터 받은 정보들을 처리하여 판단을 하거나 하는 일들을 보다 편리하게 수행할 수 있는 여러 종류의 연산자들이 필요하다. ARL에서도 범용 컴퓨터 언어인 C가 갖고 있는 다양한 형태의 자료형 뿐만 아니라 연산자들을 제공하고 있으며, 특히 여러 좌표계를 이용한 위치교시의 편의를 위하여 메트릭스 연산을 실수 연산과 같은 방법으로 사용할 수 있도록 하고 있다.

5. 프로그램 제어문

ARL에서는 일반 컴퓨터 언어가 갖는 큰 장점중의 하나인 structured program을 위한 제어구조를 갖고 있다. 사용자가 작성한 source program은 전체적으로 함수들의 선언과 글로벌 변수들의 선언으로 되어 있으며 함수들은 하나의 주함수와 여러개의 보조함수들로 이루어져 있다. 함수를 호출할 경우 호출함수와 피호출함수 사이의 정보교환은 글로벌 변수나 파라미터를 통하여 이루어진다. 파라미터를 통하여 정보를 교환할 경우 ARL에서는 "call-by-value"를 원칙으로 하나 파라미터가 배열명이거나 번지를 나타내기 위한 특수기호인 "@"와 함께 사용한 변수의 경우 "call-by-reference"로 파라미터 전달이 이루어진다.

ARL에서 각 변수들의 범위는 일반 컴퓨터 언어에서와 같이 "nested-scope-rule"에 의하여 결정되도록 함으로써 프로그램을 작성하거나 수정하기 용이하도록 하였다.

참고로 ARL에서 제공하고 있는 제어문들을 정리하면 다음과 같다.

- 판단을 위한 if, if~else
- looping을 위한 while, do~while, for
- branching을 위한 goto, break, return
- 프로그램 블럭을 나타내기 위한 {…}

6. 입출력 명령

로보트의 작업 환경이 방대해지고 복잡해지면서 이미 만들어진 정보를 이용하거나 차후 다시 이용하게 될 정보들을 저장하는 기능과 오퍼레이터와 대화할 수 있는 기능이 점차 중요해짐에 따라 ARL에서도 일반 컴퓨터 언어에서 사용되는 정도의 입출력 명령을 제공하고 있으며, 카메라나 tactile array sensor 등과 같이 한번에 같은 종류의 많은 데이타를 입출

력할 경우를 위하여 한 frame씩 정보를 입출력할 수 있는 명령(mread/mwrite)을 제공하고 있다.

7. 디버깅 명령

ARL에서 제공하고 있는 디버깅을 위한 명령으로는 BASIC 언어에서와 같이 트레이스(trace) 하고자 하는 부분을 trace on~trace off 명령으로 명시하면 그 구간내에서 사용자와 대화 형식으로 변수들의 내용을 확인하거나 각 명령들을 한 스텝씩 혹은 여러 스텝씩 실행할 수 있다.

8. ARL 명령어 모음(ARL instruction set)

지금까지 기술한 ARL의 명령어들을 정리하면 표 2와 같다.

표 2. ARL 명령어 모음

Table 2. ARL instruction set.

statement 구 분	statement 종 류
motion control statements	move, jmove, lmove, cmove, smove speed, home
channel address	CH (channel-no)
robot address	Rob (robot-no)
interlock statements	wait, send, receive, pause
welding statements	start-weld, stop-weld, set-welding-parameter
position related functions	RPY, Euler, Rot Trans, Cyl, Sph, Joint
robot initialization statements	define robot define home
debugging statements	trace on, trace off
I/O statements	open, close, read, write, mread, mwrite
operators	scalar/matrix arithmatic operators logical and relational operators, bitwise and assign opperators
mathmetical functions	sin, cos, tan, ·····
data types	character, integer, real, frame, array
flow control statements	if ~else while, do~while, for goto, break, return

Ⅳ. ARL 인터프리터

로보트 언어에 대한 번역기로는 실행속도를 빠르게 하기위한 컴파일러 타입과 사용자와의 편리한 interaction을 강조한 인터프리터 타입이 있으나 오프라인으로 만족할만한 프로그램을 얻기 위해서는 여러번의 시행착오를 거쳐야 되므로 일반적으로 로보트 언어에 대한 번역기로는 인터프리터가 많이 이용되고 있다. 이외에도 언어에 보다 많은 지능을 부여하기 위해서는 보다 풍부한 어의와 자료구조들을 필요로 한다는 점에서 인터프리터가 중요한 의미를 갖는다.

1. 인터프리터 설계개요

ARL에서는 필요에 따라 언어의 구조나 내용을 확장하거나 변경하기 용이하도록 UNIX 시스템에서 컴파일러 작성에 편의를 위하여 제공하고 있는 투울인 "Lex"를 이용하여 어휘분석기를, "Yacc"를 이용하여 구문분석기를, C 언어의 프리프로세서인 "Cpp"를 변형시켜 프리프로세서를 각각 구현하였으며 프로세스의 생성과 소멸, 공통메모리의 할당등은 시스템 호출에 의하여 이루어진다.

ARL에서의 언어해석은 본격적인 프로그램 해석에 앞서 우선 초기화 루틴에서 해석에 필요한 글로벌 변수들이나 테이블, 스택등을 초기화 시키는 이외에 프로세스간 통신을 위하여 사용되는 공통메모리를 할당하고 찬넬프로세스와 감시프로세스를 생성시켜 작업을 수행하도록 한다. 그외 여러 단계에서의 처리 내용은 매우 일반적인 것으로 프리프로세스에 의해 마크로 확장등과 같은 프리프로세서 명령이 처리되어 그 결과를 코아에 저장하면 어휘분석기는 이를 인터프리터의 내부 표현 형태인 토큰열(token string)로 바꾸고 새로운 변수나 상수일 경우 이를 차후 구문분석이나 어의해석에 이용하도록 심벌테이블에 저장한다.

구문분석기에서는 사용자가 작성한 문장들이 문법에 맞는지 여부를 확인하고 문법에 서촉될 경우 적절한 에러메세지와 함께 계속적인 해석을 위하여 에러에 대한 복구작업을 수행하며 문법에 맞을 경우 심벌테이블과 글로벌변수, 공통메모리, 시스템호출등을 이용하여 해당 문장의 어의에 따라 필요한 작업을 수행한다.

2. 인터프리터의 자료구조

사용자가 작성한 프로그램을 해석하는 과정에서 각 심벌들에 관련된 정보를 저장하거나 이미 저장된 정보들을 이용하기 위하여 심벌테이블을 사용하고 있으며 심벌의 수에 관계없이 거의 일정한 처리속도를 갖도록 hash table을 이용하여 구현하였다. 심벌테이블 엔트리에 저장되는 내용들을 C 구문으로 표시하면 다음과 같다.

```
struct symtab {
        char    *s_name;          /*symbol name      */
        unsigned s_type;          /*data type        */
        unsigned s_size;          /*size of the symbol*/
        char    *s_lval, *s_rval; /*l_value & r-value */
        int      s_blknum;        /*block depth      */
        int      s_lineum;        /*line number      */
        char    *s_funentry;      /*function entry   */
        struct   dim_list *s_dlist; /*dimension list  */
        struct   symtab *s_next;  /*next symbol entry */
      };
```

이밖에도 언어가 갖는 고유한 특성인 nested structure를 처리하기 위한 여러 종류의 스택과 각 로보트의 현재 상태를 표시하기 위한 로보트 테이블, 심벌들의 스코프(scope)를 나타내기 위한 글로벌 변수 등이 있다.

V. ARL을 이용한 예제 프로그램

ARL을 이용한 예제 프로그램으로 로보트 2대를 이용한 간단한 조립작업과, Nova-10을 이용한 용접 작업을 보이고자 한다.

1. 로보트 2대를 이용한 조립작업

로보트 두대를 이용한 조립작업으로 init점에 있는 작업물을 로보트1(puma560)이 mid점으로 옮기고, 로보트2(rhino)는 로보트1이 작업물을 mid점으로 옮기고 나면 그 작업물을 hole점에 있는 구멍에 끼워 넣는 작업을 하기 위한 ARL프로그램은 다음과 같다.

```
/* set up environment */
#define  CHANNEL     CH(1)
#define  GRIPPERI    CH(2)
#define  GRIPPER2    CH(3)
#define  WRIST2_CUR  CH(4)
#define  OPEN    1
#define  CLOSE   0
#define  CRT     1
#define  PASSED  103
#define  offset  Trans(0, 0, 0.2)

main( )
{
real  init[4, 4]=Trans(0.6, -0.2, 0.1)*
      Euler(0, 90, 0);
real  mid[4, 4]=Trans(0.3, -0.4, -0.1)*
      Euler(0, 60, 0);
real  hole[4, 4]=Trans(-0.5, -0.2, -0.3)*
      Euler(0, 30, 0);
with Rob(1) {
  real p4[4, 4]=Trans(-0.3, 0.3, 0.0)*
                Euler(0, 90, 0);
  frame Base=Trans(0.0, 0.0, 0.0)*
            RPY(0, 0, 0);
  trace on
  define robot "puma560";
  define home p4;
  speed 0.5;
  move to init* offset, Base. init;
  GRIPPER1=CLOSE;
  move to mid* offset, mid;
  GRIPPER1=OPEN;
            /*place object at mid point*/
  CHANNEL=PASSED;
            /*signal to robot2      */
  home;
  write(CRT, "\nEND Rob(1)\n");
  }
with Rob(2) {
  define robot "rhino";
  wait while (CHANNEL != PASSED);
            /*wait for robot1's signal*/
  pause(1);
  move to mid*offset, mid;
  GRIPPER2=CLOSE;
            /*grasp the object        */
  move to mid*offset, hole
    while(WRIST2_CUR<10.4);
            /*insert object in hole   */
  home;
  write(CRT, "/nEND Rob(2)/n");
  }
}
```

2. Nova-10을 이용한 용접 작업

간단한 용접 예로 용접 로보트인 Nova-10을 이용하여 정면을 바라보고 있는 작업물 중앙에 지름이 300㎜인 원을 따라 용접하는 작업을 하기 위한 ARL 프로그램은 다음과 같다.

```
#define schedule 6
weld-circle(p1, p2, p3)
{/* weld circle, which composed of
    p1, p2, p3 points */
    start_weld schedule;
    cmove to p1, p2, p3, p1;
    stop_weld;
}
```

```
main()
{
    frame  p0, p1, p2, p3;
    p1=Trans(0.70,  0.1,  0.4)*
        RPY(0.0,  90.0,  -50.0);
    p2=Trans(0.55,  0.1,  0.1)*
        RPY(0.0,  90.0,  -50.0);
    p3=Trans(0.40,  0.1,  0.3)*
        RPY(0.0,  90.0,  -50.0);
/*  Set welding parameters for schedule no.,
    "schedule" */
    set_welding_parameter schedule
        {
    cf_cur=90;  /* crater fill current */
    cf_vol=25;  /* crater fill voltage */
    weav_type= 3;
        }
    move to p1 *Trans(0.0,  0.05,  0.0);
    weld-circle(p1,  p2,  p3);
    home;
}
```

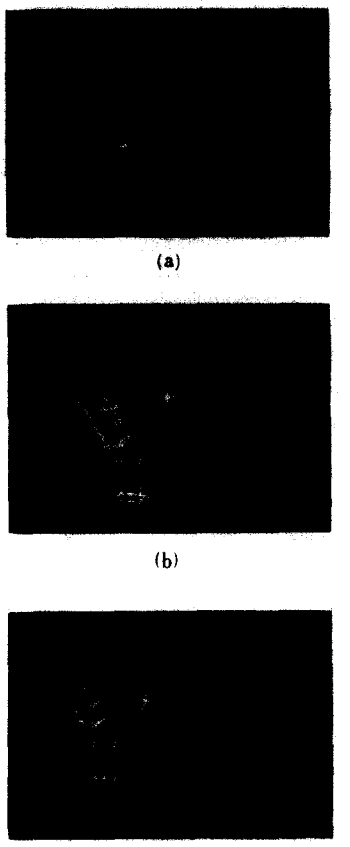
Ⅵ. 결　　론

본 논문에서는 기계 종속적인 점을 최소로하고 작업장내에 존재하는 로보트 뿐만 아니라 센서나 용접기 같은 여러 주변 장치들을 쉽게 제어 수 있고 작업장 환경변화에 적응하기 용이한 새로운 머니플레이터 레벨 로보트 언어를 제안하고, UNIX를 탑재한 Sun-3/110 워크스테이션에서 번역기 개발 투울인 Lex와 Yacc 컴파일러를 이용하여 어휘분석기와 구문분석기를 구현하고 나머지 부분은 C언어를 이용하여 구현하였으며, 다른 로보트 언어들과의 비교는 부록에 상세하게 여러 항목에 걸쳐 나타냈다.

참고로 전체 프로그램의 크기는 C 언어로 약 15,000줄 가량되며, 해석 속도는 명령의 복잡정도에 따라 다르겠으나 초당 약 500개의 명령을 수행한다.

한편 프로그래밍의 편의를 위하여 source program을 한 스텝씩 실행하거나, 필요한 경우 변수들의 내용을 볼 수 있도록 하기 위한 디버깅 투울을 제공하고 있으며, 이미 본 연구실에서 개발된 바 있는 삼차원 컴퓨터 그래픽 시뮬레이터와의 인터페이스를 마련하여 한 대의 로보트로 이루어지는 작업장에 대해서는 로보트와 작업물의 움직임을 그래픽 화면을 통하여 확인할 수 있도록 하였다.

본 논문의 연장선상에서 현재 사용 가능한 로보트



(a)

(b)

(c)

그림 4. 용접작업에 대한 그래픽 시뮬레이션 결과
(a) 용접하기전 상태
(b) 300㎜원을 용접하는 중간과정
(c) 원을 용접하고난 후의 상태

Fig. 4. Graphic simulation for a simple welding task.
(a) Initial state.
(b) Intermediate state.
(c) Final state.

(Puma-560, Nova-10, Rhino-XR) 이외에 다른 로보트들에 대해서도 데이타 베이스를 구축하고 있으며, 프로세스간 정보교환을 위하여 찬넬등을 사용하는 것을 보다 일반적인 방법인 원드모델로 확장중에 있다. 이 외에도 다중로보트를 위한 디버깅 투울로써 프로그램의 진행과정에 따라 작업장의 상태를 볼 수 있도록 하기 위한 그래픽 시뮬레이터를 개발중에 있다.

參 考 文 獻

[1] Shimon Y. Nof, Handbook of Industrial

부 록

표 3. 다른 로보트 언어와의 비교
Table 3. Comparison with other robot languages.

	AL	AML	HELP	JARS	MCL	RAIL	RPL	VAL	ARL
Language type									
subroutines				×			×		
extension					×				
new language	×	×	×			×		×	×
Geometric data types									
frame	×			×	×	×		×	×
joint angles		×		×				×	×
vector	×	×		×	×			×	
Ability to control multiple arms									
sigle arm only		×		×		×	×	×	
multiple arms	×		×		×				×
Control structures									
statement labels		×	×	×	×		×	×	×
if-then	×	×	×	×	×	×	×	×	×
if-then-else	×	×	×	×	×	×	×		×
while-do	×	×	×	×	×	×	×		×
do-until	×	×		×		×	×		×
case	×			×		×	×		
for	×	×		×		×	×		×
begin-end	×	×							×
cobegin-coend	×								×
subroutine	×	×	×	×	×	×	×		×
Control modes									
position	×	×	×		×	×	×	×	×
guarded moves	×	×	×						×
conveyor tracking				×	×				
Motion types									
joint interpol.	×	×	×	×		×	×	×	×
straight-line	×			×	×	×	×	×	×
spline	×	×		×	×		×	×	‡
circle						×			×
Sensor interfaces									
vision	×	×	×	×	×	×	×	×	‡
force	×	×	×						‡
limit s/w	×	×							‡

	AL	AML	HELP	JARS	MCL	RAIL	RPL	VAL	ARL
Support modules									
interpreter	×	×	×	×		×	×	×	×
compilwer	×			×	×		×		
simulator	×	×			×				×
macro	×		×		×		×		×
Debugging features									
single stepping		×	×			×	×		×
break points	×	×				×	×		
trace		×	×		×		×		
dump	×		×		×		×		×

‡ *Currently being developed at KAIST.*

Robotics, pp. 336-381, 807-820, John Wiley & Sons, 1985.

[2] Mikell P. Groover et al, Industrial Robotics Technology, Programming, and Applications, pp. 189-289, 311-343, McGraw-Hill, 1986.

[3] Richard P. Paul, Robot Manipulators Mathematics, Programming, and Control, MIT press, 1982.

[4] Ulrich Rembold Klaus Horman, Languages for Sensor-based Control in Robotics, pp. 3-68, Spinger-Verlag, Germany, 1987.

[5] Maurice J. Bach, The Design of UNIX Operating system, pp. 355-388, Prentce-Hall, 1986.

[6] T. Lozano-Perez, "Robot programming," *Proc. IEEE,* vol. 71, no. 7, pp. 821-842, 1983.

[7] Susan Bonner and K.G. Shin, "A comparative study of robot languages," *IEEE computer,* vol. 15, no. 12, pp. 82-86, 1982.

[8] William A. Gruver et al, "Industrial robot programming languages: A comparative evaluation," *IEEE Trans. Syst. Nan.Cybern.,* vol. SMC-14, no. 4, pp. 565-570, 1984.

[9] V. Hayward, "introduction to RCCL: A Robot Control "C" Library," Technical report, Purdue Univ. Oct. 1983.

[10] Brain O. Wood et al, "MCL, The manufacturing control language," *Proc.* of the 13th ISIR, 1983.

[11] 김 형수, "로보트 교시를 위한 off-line programming system의 설계", 한국과학기술원, 석사학위논문, 1987.

[12] 장 원, 정명진, 변증남, "컴퓨터 그래픽스를 이용한 로보트 매니퓰레이터의 구현방법", 대한전자공학회논문지, vol. 24, no. 2, pp. 207-214, 1987.

[13] 장 철, 장 원, 정명진, 변증남, "로보트와 제어기의 개발을 위한 로보트 시뮬레이터의 설계", 대한전자공학회논문지, vol. 25, no. 1, pp. 8-17, 1988.

[14] NOVA-10 Robot User's Manual, DAEWOO Heavy Industries Ltd.

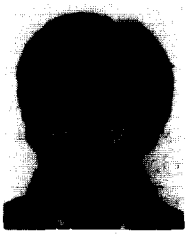
[15] Kunikatsu takase et al, "A structured approach to robot programming and teaching, *IEEE Trans. Syst. Nan. Cybern.* vol. SMC-11, no. 4, pp. 274-289, 1981. *

著 者 紹 介

朴 鍾 憲 (正會員)

1959年 3月 20日生. 1986年 2月 고려대학교 전자공학과 공학사 취득. 1986年 3月～1988年 2月 한국과학기술원 전기 및 전자공학과 공학석사학위 취득, 현재 한국과학기술원 전기 및 전자공학과 박사과정 재학중. 주관심분야는 로보트언어와 로보트 프로그래밍 등임.

●



張 哲 (正會員)

1962年 12月 28日生. 1984年 2月 한국항공대학 전자공학과 공학사 취득. 1984年 3月～1986年 2月 한국과학기술원 전기 및 전자공학과 공학석사학위 취득. 현재 한국과학기술원 전기 및 전자공학과 박사과정 재학중. 주관심분야는 다중로보트 시스템, 로보트충돌 회피계획 및 로보트 프로그래밍 등임.

崔 秉 旭 (正會員) 第25卷 第10號 參照

현재 한국과학기술원 전기 및 전자공학과 박사과정

●

鄭 明 振 (正會員) 第25卷 第 8號 參照

현재 한국과학기술원 전기 및 전자과 조교수