자연언어시스템을 위한 시스템언어의 설계와 구현*

(Design and Implementation of System Progrumming Language for Natural Language Understanding System)

박 종 구** 이 정 현***

(Park, choung goo) (Lee, Jung Hyun)

요 약

자연언어, 특히 한국어처리에 적합한 구조를 갖는 프로그램언어인 INPRAN을 정의하여 실현하였다.

자연언어처리에서 필수적으로 수반되는 동작을 분석하여 이것을 9개의 〈test〉와 12개의 〈action〉으로 형식화하여 INPRON의 기본구조로 하였다. 이들 기본구조는 기능별로 결합하여 〈network〉을 구성하여 계층적 구조를 갖는 자연언어처리시스템을 형성할 수 있다.

INPRAN은 자연언어가 갖는 stack기능을 중심으로 다양한 데이타구조를 수용하여 문법성분의 결합관계와 의미구조등을 특정 언어이론과는 무관하게 임의적으로 표현할 수 있고, 이들의 상호 연관처리도 가능하다. 또한 INPRAN은 재귀적인 기능뿐만아니라 비문법적 문장을 배제하는 상태속성처리와 다중 〈test〉의 유연한 제어구조를 갖고 있으며 자연언어처리시에 발생하는 실제적인 문제를 효과적으로 해결할 수 있다. 따라서 INPRAN은 자연언어이해시스템의 연구와 개발에 효과적인 도구로 응용될 수 있다.

INPRAN은 LISP 1.5을 사용하여 MACRO형식의 인터프리터로 실현하였다. 이로써 확장성이 우수한 interactive한 incremental 시스템이 되었으며, 컴파일러등의 보조 소프트웨어의 개발이 용이하다.

Abstract

A new Programming language, INPRON, adequate for Natural Language—especially Korean—is defined and implemented.

The basic structures of INPRON are formalized with 9〈test〉and 12〈action〉functions by analyzing the mandatory operations in natural language processing. A hierarchical

* 本 論文은 1985年度 校內學術研究費 支援에 依한 研究論文임

** 원광대학교 전자계산공학과

*** 경기대학교 전자계산학과

natural language processing System Can be organized with this basic functions combined functionally, The INPRON can represents the associative relations of grammartical constituents, and Semantic structures by using Varient data structures based on stack function of natural language, And the interrelation ships between syntatic and semantic structures can be also represented.

The INPRON has the flexible control structure to exclude ungrammatical sentences and perform multiple 〈test〉 for transition as well as recursive processing facilities. As INPRON deals with practical problems in natural processing, INPRON can be applied as a effective software tool for research and development of natural language understanding System

INPRON is implemented as interpreter with MACRO facilities using LISP 1.5. By means of MACRO. INPRON become a interactive and incremental system with powerful expansion abilities and supporting softwore, for example compiler, easily developed.

1. 서 론

반도체공학과 컴퓨터과학의 발달로 값싼 고성능의 컴퓨터가 일반적으로 보급되고 있으며, 컴퓨터의 응용영역도 급속히 확장되고 있다. 특히 Chomsky가 언어의 심층구조개념을 제안한 이래, 변형생성문법이나 격문법등의 언어의 보편적 구조를 규명하려는 노력과 더불어 자연언어를 이용하여 컴퓨터처리를 합리화하고 지능화하여 효과적인 man—machine communication을 이루려는 연구는 가속화되어 왔다.[2] 자연언어처리는 50년대의 기계번역(machine translation)을 시작으로 간단한 templete matching 연구에서부터 frame이나 script등의 의미표현에 이르기까지 인공지능과 제5세대 컴퓨터의 핵심과제로 활발한 연구가 진행중이다.

그러나 자연언어처리는 형태소의 결합방식, 구문구조분석과 문장의미의 표현등 언어자체의 문제뿐만아니라 컴퓨터에 의한 자연언어처리시스템의 실현에는 많은 문제점이 내재되어 있다. 언어의 환경적 자질과 선택적 자질을 포함한 형태론적 구조의 형식화와 이의 처리방법을 비롯하여 언어의 의미정보를 어떠한 방법으로 컴퓨터에서 실현할 것인가 등 수많은 문제가 수반된다. 이와같이 자연언어처리는 문법과 의미의 복합적인 문제가 내포되어 있기 때문에, 이를 위한 프로그램은 종래의 것과는 다른 차원에서 생각하지 않으면 안된다. 따라서 종래의 수치연산과 사무처리중심이거나 기호처리용의 프로그램언어가 아닌 자연언어의 고유한 구조와 특징을 처리할 수 있는 새로운 프로그램언어가 필요해 진다. 이러한 욕구를 충족시킬수 있도록 상태천이를 중심으로 한 A-

TN(Augmented Transition Network)[1, 4]문법기술을 중심으로 한 LINGOL[6] pattern matching기법을 도입한 PLATON[7] 기계번역을 위한 GRADE등 많은 문법인식방법이 개발되었다.[8] 특히 ATN은 자연언어의 특징인 stack 구조를 효과적으로 이용하고 있기때문에 recursive처리가 가능하고 문법규칙에 대응하는 arc에 임의의 LISP프로그램을 부가할 수 있어 널리 사용되어 왔다.

그러나 이들 방법은 문법인식에 치중하여 언어의 의미표현을 경시하였거나 애매한(ambiguous) 문장처리가 복잡한 단점이 있으며 한국어와 같이 어순이 자유로운 굴착어에는 획일적인 left-to-right 처리에 기초한 프로그램언어는 효과적이지 못하다.

본 논문에서는 한국어등의 자연언어이해시스템에 적합한 새로운 프로그램 언어인 INPRAN(Incremental Pragramming Language for Natural Language Understanding)을 설계하여 실현하였다. INPRAN은 자연언어이해시스템의 연구와 응용에 효율적인 도구가 될 수 있도록 다음사항을 고려하여 설계하였다.

① 언어이론과 무관하다.

종래의 ATN의 BUILDQ와 같은 함수는 주로 변형생성문법적인 tree구조만을 다루고 있다. 실제의 자연언어처리에 필요한 데이타구조는 tree뿐만아니라 frame또는 network등 다양하며 언어이론도 격문법, Systemic grammar등 많은 이론이 있으며 아직 자연언어처리를 위한 명확한 데이타구조나 언어이론이 정립되어 있지 않은 상태에서는 이들 이론을 적절히 수용할 수 있는 방법이 필요해진다.

② 문법처리와 의미처리의 연관성을 부여하였다. 의미처리를 분리하므로써 구문구조분석이 정확하지 않고, 시스템의 상호연관처리가 효율적으로 이루어지지 않는다. 의미표현과 처리가 자연언어이해에 중요한 요소가 언식되고 있기 때문에 문법과 의미의 상호 의존성을 처리할 수 있어야 한다.

③ 한국어의 특성을 고려하였다.

서구어는 형태소분석이 복잡하지 않지만 한국어의 경우에는 조사와 어미가 다양하게 발달하여 있어 형태소 분석이 차지하는 비중이 크며 또한 어순이 자유롭기 때문에 left-to-right 방법만으로 해결할수 없는 문제가 많이 발생한다.

④ 비문법적 문장의 파생을 억제할 수 있게 하였다. INPRAN에서는 상태자체에 속성을 부여하여 처리함으로써 ATN의 다중 arc에서와 같은 비문법적 문장의 발생을 개선하였다.

⑤ 생성을 고려하였다.

생성문제는 자연언어처리시 대두되는 중요문제중의 하나이므로 이 부분에도 적용할 수 있도록 하였다.

이것을 위해 INPRAN은 종래와 같은 register중심처리가 아니라 stack을 중심한 처리가 되며 register는 보조처리로만 이용하게 된다. 이것은 자연언어의 구조적 특징과 일치하며 문법성분의 결합관계와 의미분석이 명확하게 되어진다.

INPRAN은 표준적으로 사용되고 있는 LISP

1.5로 인터프리터형식으로 실현하였다. 실현에 있어서는 자연언어처리시스템의 연구개발의 효용성을 고려하여 MACRO expansion을 사용하였고 trace와 backrace 지정을 가능케하여 user friendness하게 하였다. MACRO 형식을 이용한 incremental 형식이기 때문에 개발용 컴파일러의 작성도 용이한 부수적인 이점이 있다.

2. INPRAN의 정의

구문분석의 효과적인 도구로 transition network이 다방면으로 사용되어 왔으며,이것의 기능을 강화하여 정식화한 것을 ATN(augmented transition network)이라 할 수 있다. ATN은 언어의 인식능력과 표현의 효율성에 있어 장점을 갖고 있기때문에 질의 응답등 자연언어이해시스템의 중요부분으로 사용되었다.[1,4]

ATN은 그림 1과 같이 finite state machine을 변형한 것으로 arc에 임의의 〈test〉와 〈action〉을 첨가할 수 있으며 상태천이를 재귀적(recursive)으로 할 수 있어 결과적으로

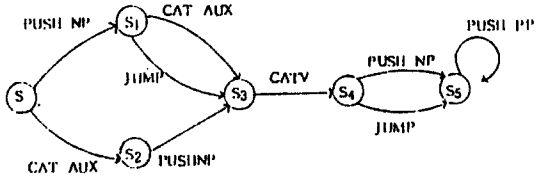

그림 1 : ATN의 예

Fig 1 :Partial Example of ATN

Turing machine과 등가의 기능을 갖게 되었다. 이러한 언어인식능력은 자연언어처리에 아주 적합한 것이다.

INPRAN은 이러한 언어인식능력을 갖을 수 있도록 상태천이에 있어 state와 arc의 개념을 확장하여 사용한다. 상태천이에 의한 언어인식은 언어학적으로도 그 타당성이 검토된 바가 있다.[3]

INPRAN의 구성은 다음과 같이 정의한다.

〈INPRAN〉 ::=〈subsystem-unit〉*

〈subsystem-unit〉 ::=〈network〉*

〈state〉::=〈state-name〉

(〈state-test〉)*

〈arc〉 ::=〈test〉〈action〉*

INPRAN의 Top level은 자연언어이해 시스템에 필요한 〈subsystem-unit〉의 총체적인 관리로 되어있다. 〈subsystem-unit〉는 자연언어시스템에서 형태소분석, 구문구조분석등 독립적인 처리단위로 존재하던 부분들이다. INPRAN은 이들 상호의 독립성뿐만아니라 연계관계를 형성할 수 있도록 되어있다. 각 〈subsystem-unit〉는 독립적인 기능처리를 하는 〈network〉으로 구성된다. 이 〈network〉은 transition network형식을취하지만, 〈state〉자체에 〈state-test〉을 갖는다. 종래에는 〈state〉를 상태천이를 위한 연결점으로 밖에 생각하지 않았기때문에 비문법적 문장의 파생등의 난점이 있었으나, INPRAN에서의 〈state〉는 자연언어의 구성성분 또는 기능요소의 논리적인 표현으로 간주한다.

2-1 〈test〉의 형태

arc의 일반형은 다음의 LISP S-표현식으로 표시된다.

(〈arc-name〉〈test〉〈action〉*)

art는 상태천이의 중추적 기능을 담당하고 있기때문에 자연언어의 여러가지 현상을 수렴할 수 있어야 한다.

INPRAN은 구문분석과 의미처리의 가능성을 고려하여 다음과 같은 arc형을 갖는다.

① (CAT 〈category-list〉 〈action〉*)

처리중인 단어의 구문분류가 〈 category-list〉의 일원이면 동작한다.

② (MEM 〈letical-list〉 〈action〉*)

현재의 단어가 〈lexical-list〉의 일원이면 동작한다. 여기서 〈lexical-list〉는 실제적인 단어집합이나 이에 상응하는 〈 file-name〉 을 쓸 수 있다. 한국어의 불완전 명사 또는 보조동사처리에 유용한 것으로 종래 ATN의 WORD 의 실제적인 확장이다.

③ CHK 〈property-value-list〉〈 action〉*)

현재 단어의 의미속성을 검사한다.

④ (TST 〈property〉 〈property-Value-list〉 〈action〉*)

단어의 〈property〉 값이 〈property-Value-List〉내에 존재할 때 동작한다. 〈property〉는 자연언어처리시스템에서 정의된 임의의 속성을 지칭하는 것으로, TST는 구문분석이나 의미처리시에 광범위한 응용성을 갖는다.

⑤ (ARK 〈arc-test〉 〈form〉 〈action〉*)

〈form〉을 〈arc-test〉에 적용하여 검사한다. 이 〈arc-test〉는 정의된 〈test〉뿐만아니라 사용자가 정의한 새로운 〈test〉 또는 임의의 LISP함수를 사용할 수 있다. 종래에는 상태천이능이 고정되어 있었지만 INPRON에서는 필요에 따라 다양한 〈test〉을 부가할 수 있는 강한 확장성을 갖는다.

⑥ (STAT (〈test〉 〈action〉*))

〈state〉의 처리상황을 점검하기위한 것으로 〈test〉는 정의된 〈test〉나 임의의 LISP함수가 사용된다. 이로서 〈state〉는 논리적인 구문의미단위의 성격을 갖는다.

⑦ (SEEK 〈state〉 〈pre-action〉* 〈post-action〉*)

〈pre-action〉을 실행하고 〈 state〉로 PUSH한다.

⑧ (SEND 〈action〉*)

최근에 SEEK한 상태로 POP한다. SEEK 와 SEND는 재귀적 (recursive)처리를 위한 상태천이함수이다.

⑨ (JUMP 〈state〉)

〈state〉로 무조건 천이한다.

INPRAN의 〈test〉는 구문구조와 의미처리의 효율성을 고려하였으며 강한 확장성을 갖고 있다.

2—2 〈action〉의 형태

자연언어의 생성과 전개는 재귀적으로 이루어진다. 특히 한국어는 삽입절등의 수식어구가 중심어의 앞쪽에 위치하는 전형적인 언어로 재귀적인 성격이 특히 강하다. 영어등의 서구어에서도 전개의 위치만이 다를뿐, 재귀적인 전개는 모든 언어의 공통적인 현상이다. 이러한 자연언어의 생성과 전개과정을 처리하기위해서는 종래의 register중심의 처리보다는 stack중심의 처리가 더 효율적이다.

INPRAN은 이러한 관점을 중시하여 stack중심의 〈action〉구조를 갖는다. stack기능은 ATN등의 register처리기능을 모두 포함할 수 있다.

또한, 자연언어처리에 필요한 데이타는 tree, frame 또는 network등 그 구조가 다양하다. 〈action〉는 이러한 구조를 처리할 수 있는 범용적인 구조를 갖지 않으면 안된다.

① (PUSHT 〈stack-name〉 〈form〉)

〈form〉을 〈stack-name〉의 top에 PUSH한다.

② (PUSHB 〈stack-name〉 〈form〉

〈form〉을 〈stack〉의 bottom에 PUSH한다.

③ (POPT 〈stack-name〉)

〈stack-name〉의 top을 pop한다.

④ (POPB 〈stack-name〉)

〈stack-name〉의 bottom을 pop한다.

⑤ (LIFT 〈stack-name〉)

〈stack-name〉의 내용을 모두 pop하고 clear한다.

⑥ (SETR (register-name) 〈form〉)

〈form〉을 〈register-name〉에 set한다.

⑦ (GETR 〈register-name〉)

〈register-name〉의 내용을 참조한다.

⑧ (LOOKA 〈value〉)

〈value〉값에 따라 look-ahead기능을 수행한다.

⑨ (LOOKB 〈value〉)

〈value〉값에 따라 look-back기능을 수행한다. LOOKA 와 LOOKB는 한국어의 보조동사구조처리에 효과적이다.

⑩ (TO 〈state〉)

〈state〉로 상태천이이다.

⑪ (ERR 〈state〉)

디버깅또는 시스템의 처리기능을 검사하기 위해 사용한다. error의 종류에 따라 〈state〉을 형성하고 이것을 하나의 독립된 〈network〉으로 간주할수있다.

⑫ (MAKE (〈label〉〈action〉)*)

데이타구조 형성을 위한 〈action〉으로 〈action〉에 임의의 LISP함수나 정의된 기능을 사용하여 다양한 데이타구조를 형성할 수 있다.

2-3 데이타구조

자연언어처리에서는 다양한 데이타형식이 이용되고 이용되고 있으므로 조직적이며 체계적인 데이타처리가 이루어져야 한다.

INPRAN은 처리의 효율성과 일반성을 고려하여, 처리에 필요한 원시 데이타구조(primitive data structure)을 property을 중심으로 한 연상리스트(association list)를 이용한다. 형태소나 구문구조의 특성과 의미소성(semantic primitives)등 자연언어처리시에 발생하는 데이타구조는 일반적으로 property을 중심으로 전개되기 때문에 연상리스트구조는 원시데이타형식으로 적합하다. 또한 LISP에서는 연상리스트의 다양한 처리방법을 제공하고 있기 때문에 유연한 처리를 보장 받을 수 있다.

그러므로 INPRON의 lexicon등은

(〈entity〉 (property1 . value1)

(property 2. value 2)...(property n. value n))

의 형식으로 구조화된다.

처리에 필요한 데이타구조가 일의적으로 결정되므로 시스템설계가 간편해지는 장점이 있다.

3. INPRAN의 실현

INPRAN은 그 필요성에 따라 인터프리터(interpreter)또는 컴파일러(compiler) 형식으로 실현할 수 있다. 그러나 software tool로서의 기능을 높이고 interactive한 처리를 하기 위해 본 논문에서는 인터프리터형식으로 하였다. INPRON은 자연언어시스템의 연구개발에 능동적으로 적응할 수 있도록 trace와 backtrancking등을 갖는 incremental형 시스템이 되도록 하였다.

3－1 INPRAN의 구조

INPRAN은 시스템의 관리를 목적으로 하는 TOP-LEVEL, arc의 〈test〉처리를 위한 EVAL-TEST와 〈action〉을 수행하는 EVEL-ACT의 3부분으로 구성된다.

TOP-LEVEL은 〈subsystem-unit〉을 정의하고 이들을 계층적 구조로 결합한다. 또한 INPRAN에서 필요한 global-variable의 초기치를 설정하고 디버깅을 위한 trace방법과 backtracking방법을 지정한다.

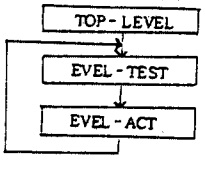

그림 2 : INPRAN의 구조

Fig 2: Structure of INPRAN

EVAL - TEST는 정의된 9개의 〈test〉를 eualuation한다. 각〈test〉는 독립된 형태의 module로 존재한다. EVAL-TEST는 〈test〉의 evaluation을 총괄하고 있기때문에 부가적인 〈test〉가 필요하다면, 시스템의 구조변경없이 해당 〈test〉의 정의만으로 실현되는 확장성을 갖는다.

EVAL-ACT는 정의된 〈action〉을 evaluation하며, 각 〈action〉은 EVAL-TEST와 같이 독립 module로 존재한다. 부가적인 〈action〉은 용의하게 적용할 수 있을 뿐만아니라 〈action〉에서는 임의의 LISP함수를 처리할 수 있다. 즉 LISP과의 인터페이스가 이루어지므로 다양한 〈action〉을 모두 수용할 수 있다.

3－2 EVAL-TEST의 실현

〈test〉는 EVAL-TEST에 의해 독립적인 단위로 실현된다. 이때 각 〈test〉의 실현에 있어서는 다음을 고려하여야 한다. 첫째, INPRAN과 자연언어처리시스템이 결합되었을때 변수충돌(variable conflict)을 방지해야한다. 둘째, 〈test〉에 의해 야기될 수 있는 side-effect를 없애야 하며, 셋째로 확장이 용이해야 한다.

INPRAN은 이러한 점들과 컴파일러 개발을 위해 각 〈test〉을 MACRO형식으로 실현하였다. 즉 각 〈test〉는 EVAL-TEST하의 독립적인 MACRO함수로 존재한다.

LISP 1.5에는 MACRO기능이 없기때문에 처

리에 알맞도록 MACRO함수 DM을 정의하였고, 변수의 binding environment를 국소화하여 변수충돌이 발생하지 않도록 함수 LET를 정의하였다. side-effect는 〈test〉자체가 form형식을 갖고 있으므로 발생하지 않는다.

```
(DM 〈test-name〉(〈test〉)
  (LIST 'LET〈internal-variables〉
    '(COND ((〈test〉)(EVAL-ACT AC-
      TION  (T NIL))))
```

그림3 : 〈test〉의 실현

Fig 3: Realization of 〈test〉

따라서 각 〈test〉는 그림3과 같은 형식으로 정의된다.

3－3 EVAL－ACT의 실현

〈action〉은 〈test〉와 마찬가지로 MACRO형식으로 실현하였다. 〈action〉은 stack 기능을 위주로 구성되고 있으므로 PUSHT, POPT가 기본 〈action〉으로 사용된다. 그러나 영어등의 서구어는 right-branch형식을 갖으므로 PUSHB POPB등의 기능도 첨가하였다. 그러므로 실제적으로는 dequeue형식을 갖는다. 일반적으로 〈action〉은 그림4와 같은 형식으로 실현된다.

```
(DM 〈action-name〉(〈action〉)
  (LISH'(LAMBDA(〈internal-varia-
    bles〉)(〈action〉)))))
```

그림4 : 〈action〉의 실현

Fig 4: Implemenation of 〈action〉

LAMBDA형식을 사용하고 있기때문에 〈action〉처리에 사용되는 변수들은 국소화된다.

4. 보조기능

INPRAN은 자연언어시스템개발에 편리하도록 trace와 tracking의 기능을 갖고 있다.

① trace 기능

trace에는 처리상황을 추적하는 것으로 state trace, test trace와 stack trace의 기능이 있으며 각 trace는 사용자의 편의에 따라 T-OP-LEVEL에서 임의 지정이 가능하다. trace가 지정되면 각 trace의 상황과 그때의 값을 표시해 준다. 그림5에 trace의 예를 보였다.

② backtracking

자연언어처리에 있어 backtracking은 중요한 위치를 차지 한다. 그러나 LISP 1.5에는 CATCH와 THROW기능이 없기때문에 back-tracking을 실현하는데는 난점이 있다. INPRON에서는 〈test〉와 〈action〉처리시 backtracking이 발생하면 최근에 branch한 곳으로 이동하는 순차 backtracking방법을 취하였다. TOP-LEVEL에서 backtracking을 지정하면 이러한 backtracking은 자동적으로 수행된다.

5. 결 론

본 논문에서는 자연언어연구와 처리시스템개발에 적합한 프로그램언어인 INPRAN을 정의하여 실현하였다.

자연언어의 일반 구조를 분석하여 처리에 필요한 〈test〉와 〈action〉을 유도하여 INPRAN의 기본구조를 구성하였다. INPRAN은 다양한 데이타구조를 수용하여 처리할 수 있을 뿐만아니라

stack중심의 〈action〉은 자연언어처리에 효과적으로 응용될 수 있다. 실현에 있어서는 MACRO을 중심으로 〈test〉와 〈action〉에 강력한 확장성을 부여하였고 LISP과도 자연스럽게 인터페이스 되도록 하였다.

INPRAN은 자연언어의 기능별 처리단위를 계층적으로 결합할 수 있기 때문에 software tool로서 적합하다.

MACRO을 이용하기 때문에 expansion으로 인하여 처리시간이 길어지고 LISP 1.5로 인하여 효과적인 backtracking이 되지 않는 단점이 있다. 앞으로 editor 등의 보조적인 software 개발과 nondeterninistic 상태천이를 deferministic하게 변환하는 것이 기대된다.

참고문헌

1. Bates, M, "The theory and Dractice of Augmented Transition Network Grammars", In Bolc, L.(ed) Natural Language Communication with Computer, Springer Verlag. 1978
2. Hendrix. G, et al."Developing a natural language interfaces to complex data", ACM Trans Database Vol 3, No 2 1978
3. Kaplan, M. "Augumented Transition Networks as Psychological Models of Sentence Comprehension", Artificial Intelligence vol 3, 1972
4. Woods. "Transition Network Grammars for Natural Language ANalysis", CACM vol, 13 , 1970
5. Feldman, J.A et al. "Connection networks and tueir properties", Conitive Science , vol 6, 1982
6. Prott, V.R , "LINGOL-A progress Report", proceeding. 4th IJCHI, 1975
7. MaKoto Nagao et al," PLATON-A New Programming Languages for Natural Language Analysis", proceeding, 2nd USA-JAPAN Computer Conference, 1975