자연어 문장을 이용한 자동 프로그래밍 시스템의 명세서에 관한 연구

김태용°, 황인환, 이정현
인하대학교 전자계산공학과

A Study on the Specification of an Automatic Programming System using Natural Language Sentence

Tae-Yong Kim , In-Hwan Hwang, Jung-Hyun Lee
Dept. of Computer Science & Engineering, Inha University

요약

자동 프로그램이란 인간이 프로그램 언어를 습득하는데 드는 노력과 시간을 감소시키고 프로그램하는 과정의 일부나 전부를 컴퓨터가 대신하도록 하여 프로그램 환경을 개선하고 유지, 보수의 비용을 줄이는데 그 목적이 있으며, 자동 프로그램 4대 구성 요소중 프로그램 명세서를 초고급언어나 예제에 의한 방법 또는 트레이스(Trace)로 기술하는 것이 일반적이다. 그러나 이 방법은 전문가가 아니면 이해하기 어렵고, 분량이 많은 경우 작성하기 어려운 문제점이 있다. 본 논문에서는 이런 단점을 개선하기 위해서 일반 사용자가 접하기 쉽고, 이해하기 용이한 자연어 문장으로 명세서를 작성한다. 그러나 자연어에는 많은 애매성이 존재하는데 이것을 방지하기 위해 사용자에게 미리 자연어 프로그램 틀을 제시한다. 자연어 문장으로 작성된 명세서는 형태소 분석과 구문 분석에 의해 처리되며, 구문 분석시 복합문과 내포문은 단문으로 분리한 다음, 동사를 중심으로한 격 프레임(case frame)를 만들며, 이것을 바탕으로 중간언어를 생성하는 방법을 제안한다.

1. 서론

자동 프로그래밍 시스템이란 용어는 1958년 ACM의 편집자에 의해서 처음으로 사용되었으며 최초의 시스템은 FORTRAN컴파일러를 들 수 있다[3]. 자동 프로그래밍 시스템은 인간이 행하고 있는 프로그래밍의 일부나 전부를 자동 프로그래밍 시스템이 대신하도록 하여 인간의 노력과 시간을 줄이고 편리한 프로그래밍 환경을 제공한다고 정의할 수 있다[9]. 더구나 요즘과 같이 컴퓨터의 보급이 확산되고 있는 시점에서 누구나 쉽게 자기가 원하는 프로그램을 작성할 수 있는 도구가 필요하다. 따라서 이 요구를 만족시킬 수 있는 방법은 자동 프로그래밍 시스템이다. 특히 사용자가 입력하는 프로그램 명세서의 작성 기법에 대한 보다 편리하고 이해하기 용이한 방법이 필요하다.

기존의 명세서 작성 방법은 초기에 많이 연구된 예제에 의한 방법, 제한된 영역에서 사용된 Special-purpose language, Logical-formalism, 최근에 가장 활발히 연구되고 있는 초고급 언어등을 사용한다[3]. 하지만 예제에 의한 방법은 사용자에게 친숙하며 이해하고 수정하기는 쉽지만 비형식 명세의 한계점이 있어서 실용화되기는 어렵다.

Special-purpose language 방법은 특별한 영역을 벗어나면 사용할 수 없고, 초고급 언어는 다시 배워야 하는 불편한 점이 있다. Logical-formalism는 계산적인 문제를 다루기 힘들고 사용자가 이해하고 작성하는데 어려운 점이 있다[1] [2] [3] [8].

본 논문에서는 이러한 단점을 개선하고자 사용자와 친숙한 자연어로 명세서를 작성하는 방법을 제안한다. 자연어는 사용자와 친숙하고 사용하기 편리하며 특별한 훈련이 필요없다는 장점을 가진다. 하지만 자연어란 많은 애매성이 존재하므로 이것을 개선하기 위해 자연어 프로그램 틀을 단들어 사용자에게 미리 제시하여, 이것을 바탕으로 명세서를 작성하므로 많은 애매성을 줄일 수 있다. 이렇게 작성된 명세서는 형태소 분석과 구문 분석 처리에 의해 각각 용언을 중심으로 격 프레임으로 표현하고, 목적언어를 생성하기 위한 전 단계인 중간언어로 변환한다.

2. 명세서의 문장 고찰

자연어의 모든 문장은 현재로는 처리할 수 없고 애매성이 많이 존재한다. 이런 단점을 개선하기 위한 방법으로 가장 일반적으로 사용되는 문장을 정의해 논 자연어 프로그램 틀을 기반으로 명세서를 작성한다.

따라서, 자동 프로그래밍 시스템의 명세서에서 사용되는 문장은 일반 문장보다는 정형화된 문장을 이용하여 프로그램을 작성한다.

명세서에 기술된 문장의 특성을 살펴보면 다음과 같이 네 가지로 나눌 수 있다.

(1) 변수 X와 Y를 정수형으로 선언한다.
(2) I의 값을 인쇄한다.
(3) 변수 A가 9보다 작거나 같을 때까지 반복한다.
(4) 만약 A가 B보다 크면 A를 출력한다.

(1)는 공동격 조사가 접속 기능으로 쓰인 문장이며 (2)는 명사구, (3)과 (4)는 내포문과 접속문을 나타낸다.

2.1. 명사구

명사구는 두 개 이상의 명사가 속격조사에 의해 된 것으로 조사의 여부와 결합 형태에 따라 구분할 수 있다.

(1) 명사 + 속격조사 + 명사 (예 : x의 값을 출력한다)
(2) 명사 + 명사 + 속격조사 + 명사 (예 : 변수 x의 값을 저장한다)
(3) 명사 + 명사 (예 : 정수형 변수. 변수 x)

(1)에서의 두 명사는 동격 관계로 이러한 형태의 명사는 첫번째 명사가 중요하다. 따라서 '출력한다'의 격 프레임 구조에서 목적격의 값은 X가 된다. 마찬가지로 (2)에서는 두번째 명사가 중요하다. (3)은 복합명사의 성격을 가지는 것으로 변수, 배열, 함수등은 예약어 형태로 인식하여 이 명사를 제외한 나머지 명사를 가지고 처리한다.

2.2. 접속문

접속문은 크게 대등 접속문과 종속 접속문으로 나눌 수 있다. 그러나 동일한 접속의 기능을 수행하는 형태소가 경우에 따라 대등 접속 어미가 되기도 하고 종속 접속 어미가 되기도 하므로 어미만으로는 구분하기가 힘들다. 따라서 접속문의 종류를 명사구 접속과 동사구 접속으로 나눈다[5].

2.2.1. 명사구 접속

명사구 접속 어미로 쓰이는 접속조사는 '와/과', '하고', '이며', '이나', '이거'등이 있다. 그러나 자동 프로그램에서 사용되는 것은 '와/과'가 대표적으로 쓰이며, 공격조사로 쓰이기보다는 단순히 명사를 접속시켜 주는 접속조사로 더 많이 사용된다.

● 변수 X와 Y를 정수형으로 선언한다.

2.2.2. 동사구 접속

자동 프로그램의 명세서를 기술할 때 자주 사용되는 형태이며, 가장 대표적인 어미로는 '고', '며', '거든', '거나', '든지', '(아)서', '해서'를 들 수 있다.

접속문에서는 문장 성분이 생략될 수 있는데 명사구인 경우는 순방향으로 뒤에 오는 동일 명사구가 생략되며, 동사구인 경우는 반대로 앞에 있는 동일 동사구가 생략된다[7].

예를 들어, 어미'고'는 동시성이나 계기성, 반복성의 의미자질로 사용된다. 따라서 자동 프로그램에서는 대등절로 인식하여 각각 완전한 한 문장으로 분리한다. 분리된 문장은 동사를 중심으로 격구조로 표현된다.

● 변수 c를 0으로 초기화하고, i를 1씩 증가시킨다.

위 문장은 '변수 c를 0으로 초기화한다', '그리고 i를 1씩 증가시킨다'로 2개의 단문으로 분리된다. 각 문장은 동사를 중심으로 격구조를 만들어 중간언어를 생성한다.

어미'(아)서'는 선행문과 후행문이 인과 관계로 맺어져 있는데 선행문이 원인과 이유를 후행문이 결과를 나타낸다. 이 어미는 사칙 연산에서 많이 나타나는 특성이 있다.

● 변수 x를 y로 나누어서 몫을 변수 c에 저장한다.

'(으)면', '거든'은 가정이나 조건을 의미하는 접속어로서 if문으로 인식하고, '거나', '든지'는 접속 기능을 하면서 선택의 의미를 가지므로 논리 연산자 'OR'로 '고'는 'AND'로 변환된다.

3. 중간언어 생성기

사용자에게 미리 제시한 자연어 프로그램 틀을 기반으로 작성한 명세서는 형태소 분석과 구문 분석을 거친다. 중간언어 생성기는 분석된 정보를 가지고 각 용언을 중심으로 격 프레임 구조로 표현한다. 이것을 바탕으로 사용자가 원하는 목적언어를 생성하여 원하는 결과를 얻을 수 있다.

본 논문에서는 분석이 끝난 문장에서 정보를 추출하여 중간언어를 생성하는 과정까지 논한다. 입력된 문장은 한 문장 단위로 처리가 되며, 한 문장 안에 있는 복합문과 내포문은 모두 단문으로 분리하여 각 용언마다 프레임을 만들어 분석된 정보를 기억한다. 각 동사의 격 프레임의 구조는 다음과 같다.

(선언
  (SUBJ - (이 가)
  (OBJECT (을 를))
  (ADV (으로)))

이와 같은 구조로 용언을 표시하며 '-'은 선택격을 표시하는 것으로 선택격은 정보가 없어도 가능함을 의미한다. 하지만 필수격은 모두 정보가 충족되어야 하고, 만약 필수격 정보가 없는 경우는 그 문장을 에러 메시지와 함께 사용자에게 표시하여 올바른 문장을 입력하게 한다. 격 프레임이 완성되면 이것을 바탕으로 각 동사의 변형 틀을 완성하여 중간언어를 생성한다.

중간언어(meta level language)는 기존의 명세서 작성 방법으로 널리 사용된 초고급 언어와 유사한 표현으로 특정 목적언어의 문장 구조와 무관하게 작성된다.

3.1. 사칙연산과 관계 연산자

사칙연산에 사용된 자연어 문장은 다음과 같다.

3-1) 변수 X에 3을 더한다
3-2) 변수 X와 Y를 더하다.
3-3) 변수 X와 Y의 합을 구한다.
3-4) 변수 X에서 3을 뺀다.
3-5) 변수 A와 B를 곱한다
3-6) 변수 X에 3을 곱한다.
3-7) 변수 A를 5로 나눈다

3-1)에서 3-3)까지는 덧셈에 관한 것으로 덧셈을 수행하는 용언은 '더하다', '합하다'를 주로 많이 사용한다. '더하다'의 동사 패턴은 2개로 구분할 수 있다.

VT1-1 (더하다

(SUBJ - (이 가)
(OBJ (을 를))
(ADV (에 에다)))

VT1-2 (더하다

(SUBJ - (이 가)
(OBJ1 (을 를))
(OBJ2 (을 를)))

그리고 VT1-1 변형 틀은 (ADD (에) = (에) + (을 를)) 이며 예3-1)의 중간언어의 표현은 ADD X = X + 3이 된다.
하지만 3-1)은 L-value값이 없는 경우인데 다음 예문은 L-value값이 있는 경우이다.

● 변수 x에 3을 더해서 변수 z에 저장한다.

위 문장을 단문으로 분리하면, '변수 X에 3을 더한다', '그래서 변수 Z에 저장한다'. 그런데 각 문장을 프레임으로 표현할 때 두번째 문장의 목적격 성분이 없다. 이 목적격 성분은 앞 문장의 계산 결과이므로 프레임 표현시 목적격 값을 채우지 말고 중간언어 생성시 앞 문장의 결과 부분을 목적격 값으로 채워 중간언어를 생성한다.
3-3)은 '구하다'의 동사로는 변형을 시도할 수 없으므로 목적어인 '합'은 '하다'가 붙어서 동사로 쓰이므로 '합하다'의 동사 패턴을 사용하여 중간언어를 생성한다. 나머지 연산도 이와 같은 방법으로 행해진다.
다음으로 관계 연산자의 표현은 자연어의 형용사로 기술한다. 여기에 사용되는 형용사는'작다', '크다', '같다'가 있는데 각각 '<', '>', '=='로 변형될 수 있다. 그리고 '작거나 같다', '크거나 같다'는 단문 분리시 2개의 문장으로 분리하여 두 문장을 'OR'로 연결하여 중간언어로 표현한 다음 최종적으로 '<=', '>='의 의미로 변형한다.

3.2. 제어문

자연어 문장 중에서 제어문을 인식할 수 있는 방법은 부사어와 접속 어미로 판단할 수 있다.
대표적인 것이 부사어 '가령', '만약'이며, 접속 어미로는 가정이나 조건의 의미를 부여하는 '(으)면', '거든'으로 인식한다

● 만약에 A가 B보다 크거나 같으면, A를 출력하고 아니면 B를 출력한다.

위 문장에서 '만약'이라는 부사어가 문장 앞에 나왔으므로 쉽게 판단할 수 있다. 따라서 이 문장이 단문으로 분리되면 4개의 단문이 생성된다. 즉 '만약에 A가 B보다 크다', '그렇거나 같다', '그러면 A를 출력한다', '그리고 아니면 B를 출력한다'. 그러므로 if문으로 변환할 때 조건식은 첫 번째, 두번째 문장이 되고 THEN이하 문장은 세 번째 문장이고 ELSE에 해당하는 부사어는 '아니면', '그렇지 않으면'이다. 위 문장에서 '아니면'이 있으므로 ELSE문장에 해당하는 것은 마지막 문장이 된다. 따라서 최종적인 중간언어는

IF (LARGE(A) (>) (B)) OR (EQUL(A) (==) (B))
THEN PRINT A ELSE PRINT B로 표현된다.

3.3. 순환문

순환문을 기술하는 방법은 2가지로 나눌 수 있다.우선 '반복한다', '수행하다'라는 동사를 중심으로 하는 한 문장으로 기술하는 방법인데, 이것은 반복되는 문장이 작은 경우에 편리하나 문장이 많아지는 경우에는 효율이 떨어진다. 나머지 하나는 반복되는 구간을 예약어 형태로 '반복 시작', '반복 끝'이라는 단어를 사용하여 반복되는 구간을 정확하게 구별하여 기술한다. 자연어 문장으로 프로그램을 기술할 때, 특히 순환문인 경우 반복되는 구간을 명확히 구별하기가 힘들다[1]. 이 단점을 개선하기 위해 이 방법을 사용한다. 또한 중첩 순환문인 경우에도 편리하고 이해하기 쉽게 기술할 수 있다. 각각 예를 들면 다음과 같다.

● 변수 s에다 N을 더하고, N를 1씩 증가시키고, s가 1000보다 작을 때까지 반복한다.

위 문장은 단문 분리시 4개의 문장으로 분리가 된다. 또한 반복 범위는 한 문장 단위로 이루어지기 때문에 위 문장의 반복 부분은 첫번째, 두번째 문장이 되고, 세번째 문장은 이 순환문의 조건식이 된다. 따라서 중간언어로 표현하면,

(ADD (S) (=) (S) (+)N))
(INCR (N))
LOOP (s<1000)

으로 생성한다. 이 표현은 다음 단계에서 목적언어가 C인 경우에는 while, do-while문으로 변형될 수 있다. 그리고 '반복하다'라는 동사의 패턴이 '~부터 ~까지 반복한다'고 하면 이것은 for문으로 생성된다.

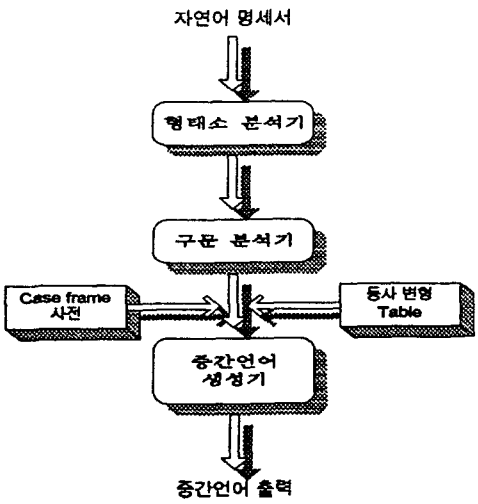
반복 시작:
변수 x를 입력받는다.
만약 x가 MAX보다 크면 MAX에 x를 대입하고,
아니면 x가 MIN보다 작으면 MIN에 x를 대입한다.
반복 끝:10번을 반복한다.

위 문장과 같이 기술하면 반복 구간을 명확히 할 수 있고 for, while, do-while으로 정확하게 변형할 수 있는데 '반복 시작' 옆에 제어문을 기술하면 while문으로, '반복 끝' 옆에 기술하면 do-while문으로 바꾸어 준다.

4. 시스템 구성 및 실험

본 시스템은 자연어로 작성된 명세서를 인하대학교 자연어연구실에서 개발한 형태소 분석기[6]와 구문 분석시 단

문 분리기[4]를 사용하여 분석한다. 각 분석 정보를 격 프레임 사전에 저장하고 이것을 바탕으로 중간언어 생성기를 거쳐 목적언어를 생성하기 위한 중간언어를 출력한다.



[그림 1] 시스템 구성도

4.1 자연어 명세서 작성 예

다음은 10개의 값을 읽어들여 그 중 최대값과 최소값을 구하는 프로그램을 자연어 문장으로 작성한 명세서의 예이다.

```
1) 변수 MAX와 MIN를 각각 실수형으로 선언한다.
2) 변수 X를 실수형으로 선언한다.
3) MAX를 -99999로 초기화한다.
4) MIN를 99999로 초기화한다.
5) 반복 시작 :
                X를 입력받는다.
                만약 X가 MAX보다 크면,
                MAX에 X를 대입하고,
                아니면 X가 MIN보다 작으면,
                MIN에 X를 대입한다.
6) 반복 끝  : 10 번을 반복한다.
7) MAX와 MIN를 출력한다.
```

[그림 2] 자연어 명세서

[그림 2]의 명세서를 중간언어 생성기로 처리하면 다음과 같은 중간언어를 출력한다.

```
(1) (DECL (float ) (MAX MIN))
(2) (DECL (float ) (X)
(3) (INIT (MAX -99999))
(4) (INIT (MIN 99999 ))
    (
     (INPUT (X) )
     (IF(LARGE(X) (>) (MAX))THEN (STORE (MAX) (=) (X))
          ELSE (SMAL(X) (<) (MIN)) THEN
               (STORE(MIN) (=) (X))
          )LOOP ( FOR 1 TO 10)
(5) (PRINT (MAX MIN))
```

[그림 3] 중간 언어

5. 결론

본 논문에서는 기존에 사용하던 명세서 작성의 단점을 개선하고자 자연어 문장으로 프로그램을 기술하는 방법을 제안하였다. 자연어란 사용자와 친숙하고 이해하기가 쉽고, 특별한 훈련이 필요하지 않다는 장점이 있는 반면 많은 애매성이 존재한다. 따라서 이 단점을 개선하기 위해 일반적으로 가장 많이 사용되는 문장을 모아둔 자연어 프로그램 틀을 참조하여 명세서를 작성하게 한다. 작성된 명세서는 형태소 분석과 구문 분석을 거쳐 각 동사를 중심으로 격 프레임를 생성하고, 생성된 프레임을 바탕으로 중간언어를 생성한다.

추후 연구로는 자연어로 작성된 명세서를 입력받아 사용자가 원하는 프로그램 언어로 결과를 얻을 수 있는 자동 프로그래밍 시스템을 만드는 것이며 이를 위해서는 자연어의 지식 표현 방법과 프로그램 변형시, 목적언어에 대한 지식 베이스 구축에 관한 연구가 필요하다.

참고 문헌

[1] Ballard, B. W., *Semantic processing for a natural language programming system*, Ph.D. Dissertation, Duke University, 1979.

[2] Alan W. Biermann, Bruce W. ballard and Anne H. Sigmon, "An experimental study of natural language programming," *Int. J. Man -Machine Studies* Vol.18. pp. 71-87, 1983.

[3] Charles Rich and Richard C. waters, "Automatic Programming : Myths and Prospects," *IEEE on Computer, August*, pp.40-51, 1988.

[4] 김광진, 송영훈, 이정현, "한국어 내포문을 분리하는 시스템의 구현," *한글 및 한국어정보처리 학술 발표 논문집*, pp. 24-34, 1993.

[5] 김진수, *국어 접속조사와 어미연구*, 탑출판사, 1987.

[6] 여상화, *다단계 필터링을 이용한 형태소 분석기의 설계 및 구현*, 인하대 공학석사 학위논문, 1992.

[7] 이익섭, 임홍빈, *국어문법론*, 학연사, 1983.

[8] 정목동, *비절차적 명세를 이용한 수치 제산 자동 프로그래밍*, 서울대 공학박사 학위, 1990.

[9] 정목동, 김영택, "지식 기반 자동 프로그래밍 시스템의 KAPS의 설계 및 구현," *정보과학회 논문지*, 15권, 4호, pp. 293-301, 1988.