통합 구결 환경을 위한 구결 아키텍처 구축 방안

구결학회 11월 월례발표회
고창수

1. 서론

이 연구는 구결자를 일반 텍스트 문서에서 입출력하는 방법과 입력 정보를 데이터베이스로 구축하여 구결 자료를 검색하고 정렬할 수 있는 방법(이하 이 두 방법을 통합하여 구결 정보화라 하겠다)에 대한 논의이다. 구결은 자토 구결과 점토 구결로 나누어지는데 이 연구에서는 자토 구결, 그 중 특히 석독 구결의 정보화 문제에 한정하여 논의를 진행하겠다. 기술적으로는 점토 구결을 정보화하는 문제가 그리 어려운 것이 아니지만, 구결 연구자들 사이에 점토 구결의 정확한 해독이 정립되지 않았기 때문에 이번 논의에서 제외하기로 한다.

자토 구결은 석독 구결과 음독 구결(순독 구결)로 나눌 수 있지만 두 구결의 정보화 방법에 있어서는 차이점을 가지지 않기 때문에, 이 연구에서는 석독 구결의 정보화에 초점을 맞추어 설명할 것이다. 석독 구결은 보통 한자의 약체자를 이용하여 우리말의 기능적 부분을 표기함으로써 한문 원전의 완전한 우리말 번역이 가능하도록 하는 우리 고유의 차자 표기이다. 그것은 구결자를 왼쪽과 오른쪽에 기입하고 한문을 우리말 어순으로 읽게 만드는 역독점을 이용한 것이므로, 지시하는 대로 구결자를 읽으면 그것은 거의 완전한 수준의 번역문과 대응한다. 따라서 석독 구결을 정보화한다면 고대 한국어의 다양한 특성을 이른바 원거리 독서(distance reading)로 조망할 수 있다.

인공지능 시대에서 구결자의 정보화는 구결이 입력된 문헌의 지능적 학습을 위해 꼭 필요하지만, 우리는 구결자의 정보화를 위한 입력 시스템마저 제대로 갖추지 못한 상태이다. 따라서 아래아 한글을 제외한 다른 저작 도구에서 구결자를 입력하지 못하는 불편함은 물론 국제회의에서 구결자 연구 결과를 공유하는 일마저 어려운 일이 되고 있다. 가장 큰 이유는 구결자의 코드가 사용자 정의 영역(PUA, Private User Area)에 배치되어 있기 때문이다. 이 영역은 말 그대로 사용자가 편의대로 쓸 수 있는 코드 영역으로 이 영역의 기호들은 알파벳이나 한자, 가나, 한글처럼 정식 코드를 배정받은 것이 아니어서 일반 텍스트 환경이나 다른 저작 도구에서 글자 표현이 불가능하며, 따라서 구결을 정보화하여 자유롭게 이용하기 어렵다.

이 연구는 구결 정보화의 문제를 해결하며, 텍스트 환경이나 디지털 기기에 상관없이 자유롭게 구결 문헌을 입력하고 입력 정보를 데이터베이스로 구축하여, 구결 문헌에 나오는 모든 구결자를 검색, 정렬할 수 있는 구결 정보화의 기술적 문제를 통합적으로 처리하는 방안을 제안한 것이다.

2. 유니코드 등재가 어려운 이유

구결 정보화를 위한 선결 과제는 PUA 영역이 아닌 유니코드의 문자 코드 영역에 구결자를 등재

하는 것이다. 그러나 구결자의 유니코드 등재를 위한 기본 작업은 그동안 다음의 이유로 난항을 겪어왔다.

1) 구결이 독립적 문자 체계가 아니라는 점
2) 코드와 음가의 일대일 대응 문제를 제대로 처리하지 못한 점
3) 코드가 할 일, 글리프(폰트)가 할 일, 데이터베이스가 할 일을 정확하게 구분하지 못한 점

1)의 문제는 구결로만 글쓰기가 불가능하다는 점에서 시작된다. 구결은 한자 원문을 제대로 이해하기 위한 보조적 수단이며, 한문의 문장 사이에서 우리말의 기능 형태들을 부가하기 위한 표기이다. 이런 점에서 독립적 글쓰기가 가능한 다른 문자들과 대등한 위치에 놓이지 않는다는 사실을 먼저 받아들여야 한다. 구결은 한문에 우리말의 기능 요소들을 첨기한 것이다. 대개 약자로 사용되지만 정자를 그대로 쓰는 경우도 있다. 정자로 사용된 구결은 한자의 코드 포인트와 별개로 유니코드에 등재해야 할 필요성을 상세히 설명해야 한다. 禾, 禾, 禾, 禾, 利은 모두 한자 利에서 온 '리, 이'를 표기하는 글리프로 각 변이형은 모두 동일한 기능에 대응하는데, 마치 초서와 해서의 차이 정도로 처리하는 시선으로는 정확한 대응이 어렵다.

위의 내용은 2)의 문제 제기와 바로 연결된다. 하나의 코드 영역을 배당받으려면, 완비된 목록과 용례 및 성격에 대한 합의가 필요한데, 이에 대한 국내 연구자들의 합의된 목록이 마련되지 못한 것이 유니코드 위원회의 코드 배정 거절 사유가 되었다.[1] 이를 위해 전체 구결자 목록의 테이블이 완비되어야 하는데, 여기서 가장 큰 문제로 대두되는 것은 바로 하나의 기능 범주 혹은 그 음가에 대응하는 구결자가 여러 자인 경우가 있다는 사실이다. 게다가 약체자 ㅅ의 경우처럼 只/果라는 2개의 원 한자와 대응하고 그 음가 역시 'ㄱ/기/과'와 같이 3개로 읽게 되는 용례의 양상은 더 복잡하다. 이 문제는 하나의 문자 코드가 하나의 발음 혹은 한자처럼 고정된 하나의 표상에 대응하는 유니코드 등재 원칙 위반을 의미한다. 한국 대표단은 구결자를 Ext. B 등 한자 확장 영역에 포함하길 원했지만, "구결은 한자가 아니므로 해당 안건은 부적절"하다는 이유로 받아들여지지 않았다(표 1 참조). 결국 이 회의에서 구결자는 한자와 성격이 다르므로 기존 한자 배열에 넣을 수 없다는 점이 확인된 셈이다.[2]

1) 1990년대 한국 측의 제안 절차가 완결되지 못한 '절차적 교착 상태'와, 제안의 내용 자체가 유니코드의 근본 원칙인 '한자 통합'과 충돌하면서, 사실상 통합되어 버린 '원칙적 흡수'가 동시에 작용했다. 즉 학문적으로 국내 학자들의 완전한 합의 자체가 있기는 어려우나, 그보다 구결이 하나의 코드 체계에 대응하기 어려운 문서로 제출된 것이 문제의 핵심이다. 1990년대 초중반 이후 구결자의 국제 표준 문자 코드 등록을 위하여, 구결자의 전산화, 구결자 목록의 확정, 구결자의 사용 전거 마련 등을 위한 꾸준한 노력이 있어 왔다. 그러나 합의된 구결자 목록이 마련되지 못하였으며, 자음 중심의 구결자 목록이 제안되어 왔던 까닭에 현재까지 구결자가 국제 표준 문자 코드로 등록되지 못하였다(김준수, 배진솔, 엄상혁 2019, 구결자의 국제 표준 문자 코드 등록 방안, 口訣硏究 第42輯, p111)

2) 1990년대의 구결 제안이 절차적 난관을 모두 통과했다 하더라도, CJK(중국, 일본, 한국) 한자에 대한 유니코드의 핵심 원칙이라는 장벽에 부딪혔을 것이다. 이것이 바로 구결 문자를 위한 별도의 블록 할당이 기술적으로 그리고 철학적으로 불가능했던 근본적인 이유이다. 유니코드의 가장 근본적인 설계 원칙은 추상적인 문자(character)와 그것의 시각적 표현인 글리프(glyph)를 구분하는 것이다. 문자는 의미나 기능을 가진 최소 단위이며, 글리프는 해당 문자가 화면이나 종이에 표현되는 구체적인 모양이다. 유니코드 컨소시엄의 입장에서 이 문자들은 이미 그들의 원형에 해당하는 CJK 통합 한자 코드 포인트로 '인코딩이 완료된' 상태였다. 이들을 새로운 블록에 다시 추가하는 것은 유니코드 표준이 적극적으로 회피하는 중복 인코딩에 해당했다. 결국 구결 문자의 정체성을 어떻게 규정할 것인가 하는 문제가 핵심 쟁점이 되었다. 이는 문자의 정체성이 어원적 기원(의

문서 번호	날짜	출처/ 저자	회의 참조	내용 요약 및 조치 항목	상태
N936	1993 -10-29	김경석 (한국)	-	"UCS-2에 구결 문자를 추가하기 위한 제안서 초안" 제출	제출됨
N1952	1999 -02-14	WG2 등록부	-	구결(Gugyeol)이 검토 대상 스크립트로 등재됨. 원 제안 문서로 N811, N874, N1606 등이 언급됨	검토 중
N1903	1999 -02-15	WG2 회의록	M33, M34, M35	한국 국가표준기구는 구결 문자(원 제안 N936)에 대한 제안 요약서를 제출하도록 요청받음	진행 중
N3352	1999 -03-15	WG2 회의록	M36, M37	구결 문자 제안 요약서 제출 요청 조치 항목이 여전히 '진행 중' 상태임을 재확인	진행 중

<표 1> ISO/IEC JTC1/SC2/WG2 구결 제안 진행 과정 (1993–1999)

3)의 문제는 두 가지 범주, 즉 구결자의 문자적 성격과 이를 이용하여 구결자를 검색, 정렬하는 데이터베이스의 제작을 혼동함으로써 구결자의 완전한 테이블을 만드는 데 여러 방해 요인으로 작용했다. 이는 구결자를 구결 글리프, 즉 구결자 기호의 표상으로 이해하는 것과, 하나의 구결자를 우리말로 해석하거나 독음하는 등의 구결 음가로 이해하는 것 사이의 변별이 이루어지지 못했기 때문에 생긴 문제이다. 위의 예에서 보는 바와 같이 원 한자인 利로부터 파생된 약체자는 원 한자의 표상 '利'를 포함하고 있으며, 그 변체 또한  ,  ,  ,  의 4가지나 된다. 원 한자와 같은 표상 는 원 한자의 코드에 대응하지 못하고 그 약체 또한 일대일 대응이 아닌 일대다 대응 관계에 있기 때문에 구결자를 독립적인 문자 체계로 받아들이기 어려운 점이 발생한다. 게다가  처럼 2개의 원 한자를 가지는 경우 하나의 코드에 여러 발음 혹은 기능 해석이 존재하게 되는 문제도 발생한다.

, , , , 를 각기 다른 코드에 배정할 경우 기본적으로 이들은 다른 데이터로 인식된다. 반면 는 하나의 데이터로 인식된다. 물론 이들을 테이블 통합 데이터와 구분 데이터로 저장하는 규칙을 만들 수 있지만, 코드 포인트라는 측면에서 보면 많지도 않은 구결자 코드 포인트들은 표상과 데이터가 혼란된 상태라는 점에서 독립된 코드 체계를 구성하기 어렵게 된다. 이러한 이유로 유니코드 위원회에서는 목록 및 증빙의 미비, 일부 문자들이 지니는 기존 한자와의 중복 문제, 활용상의 필요성 입증 부족 등을 이유로 구결 코드 문제를 반려한 것이다.

결론적으로 유니코드 컨소시엄(UTC)의 기본 입장은 "구결자 등재의 잠재적 필요성은 인정하되,

미)에 있는지, 아니면 특정 문자 체계 내에서의 기능적 역할에 있는지에 대한 문자학적 질문이다. 유니코드의 CJK 정책은 전자를 우선시했지만, 구결은 문자가 원래의 의미를 버리고 순전히 문법적, 음성적 기능으로 재탄생한 대표적인 사례이다. 한자 '爲'는 '하다'라는 구체적인 의미를 갖지만, 여기서 파생된 구결 문자는 한국어 파생동사의 어간 '-하'를 나타내는데 전용된다. 유니코드의 한자 통합 모델은 이 둘의 어원적 연결고리를 우선하여 U+7232라는 단일 코드 포인트로 통합했다. 이에 반해 별도 블록을 주장하는 의견에서는 이러한 기능적 변화가 너무나 커서 새로운 추상적 문자가 탄생했다고 볼 수 있다고 주장한다. 이는 가타카나가 별도의 블록을 부여받은 논리와 유사하다. 구결이 별도로 인코딩되지 못했다는 사실은, CJK 파생 문자에 대해 유니코드 기술 위원회가 기능적 재목적화보다는 어원적 기원을 더 중요하게 평가했음을 보여준다. 단 그 기능적 재목적화가 가타카나처럼 완전하고 체계적이며 시각적으로 뚜렷이 구별되는 문자 체계를 형성한 경우는 예외로 두었다는 점도 주목할 만하다.

한국 측의 구체적 제안이 성숙하기를 기다린다"는 것이다.[3] 한국이 성숙하고 구체적인 제안을 마련하려면 위에 언급한 1) 구결의 독립 문자적 지위 입증 2) 코드와 음가 혹은 기능의 일대일 대응 문제 해결 3) 코드가 할 일과 글리프가 할 일, 그리고 데이터베이스가 할 일을 정확하게 구분하는 문제를 해결하고 정확한 구결 코드 목록을 제안하는 것이다.

3. 구결 정보화의 현실적 해결책

구결이 유니코드의 정식 코드로 배정되기 위해 우리는 3가지 문제를 해결해야 함을 보았는데, 코드 배정 없이 구결 정보화를 해결하는 방법으로 다음의 처리 방안을 생각할 수 있다. 먼저 구결 정보화가 의미하는 바를 목록으로 정리해 보겠다.

1) 어떠한 디지털 환경, 즉 OS, 디지털 기기, 워드와 같은 저작 도구 등 어떤 환경에서든 자유로운 입출력을 보장하는 IME(Input Method Editor)의 개발

현재 구결을 입력하는 유일한 방법은 한컴이 개발한 아래아 한글 중 아래와 같은 특수 문자표에서 구결자를 골라 입력하는 것이다. (이 외에 편법으로 '상용구' 기능을 활용할 수 있다.)

<그림 1> 아래아 한글의 구결 유니코드 PUA 영역표

3) 유니코드 컨소시엄(UTC) 및 관련 표준화 기구(ISO/IEC JTC1/SC2/WG2)의 입장은 <표1>에서 볼 수 있듯, 1990년대 후반에 작성된 여러 회의록과 공식 문서에서 일관되게 확인된다. 특정 문서 하나에 명시된 단일 문장이라기보다는, 여러 회의에 걸쳐 반복적으로 기록된 '조치 항목(action item)'의 내용을 통해 그 입장을 확인할 수 있다. 1. 공식 제안서 제출 요청: 1993년 구결 문자 제안서(문서 N936)가 처음 제출된 이후, 표준화 작업을 담당하는 실무 그룹(WG2)은 한국 측 국가 표준 기구(Korean national body)에 공식적인 '제안 요약서(proposal summary form)'를 제출해달라고 지속적으로 요청했다. 이는 제안을 거부한 것이 아니라, 공식적인 심의 절차 시작을 위해 필요한 서류를 기다리고 있음을 의미한다. 2. '진행 중(In progress)' 상태의 지속: 여러 회의록(M34, M35, M36, M37 등)에서 구결 제안 요약서 제출에 관한 조치 항목은 계속해서 '진행 중(In progress)'으로 기록되어 있다. 이 기록은 위원회가 제안을 기각하지 않고 한국 측의 후속 조치를 수년간 기다렸다는 증거이다. 즉 진행의 열쇠는 한국 측이 쥐고 있었으며, 위원회는 구체적이고 성숙한 제안이 제출되기를 기다리는 입장인 것이다. 이러한 기록을 종합해 볼 때, 구결에 대한 코드 배정의 "잠재적 필요성은 인정하되, 한국 측의 구체적 제안이 성숙되기를 기다리겠다"는 입장은 유니코드 컨소시엄과 관련 위원회의 공식적인 절차와 회의록에 나타난 사실을 요약한 표현이다.

<그림 1>에서 ᄉ(1D11)과 ᄉ(1D18)은 구결의 모양이 같으나 다른 코드 포인트에 배정되어 있다. 이러한 상태에서는 두 구결자가 정식 코드를 배정받는다 하더라도 데이터로서의 가치가 없다. 게다가 이와 같이 특수 문자표에서는 두 구결자의 또 다른 정보, 즉 원 한자와의 독음 차이를 알 수가 없으며 이는 구결을 이와 같이 특수 문자로 입력하는 방식 자체에 문제가 있음을 의미한다.

구결자는 언제나 원 한자와 그 독음이라는 대응쌍으로 구성되어 있다는 사실을 이해한다면, ᄉ(1D11)과 ᄉ(1D18)의 차이는 단지 독음 'ㄱ'과 '기'의 차이에 불과하며 원 한자는 只에 해당한다는 것을 알 수 있다. ㄱ/기를 다른 코드 포인트로 구분하는 것은 큰 의미가 없다. 만약 ㄱ/기를 구분하고 싶다면 변이음으로 다루어야 할 것이다.

① 信行ㄹ具足ソニケ復ソㄱ有ヒナケ五道ヒ一切衆生リ、
② 信行乙 具足爲示彌 復爲隱 有叱在彌 五道叱 一切衆生是
③ 신힝올 구족ᄒᆞ시며 또ᄒᆞᆫ 오도ㅅ 일체중생이 있으며
④ (十六大國王은 ...과 淸)信行을 具足하시며, 또 五道의 一切衆生이 있으며

<그림 2>『구역인왕경』 원본 이미지 파일과 그 해독

<그림 2>에서 원본 이미지를 해독하면 오른쪽 그림과 같다. 오른쪽 첫째 문장은 원본 이미지를 그대로 아래아 한글 파일로 옮겨적은 것이다. 여기에는 오른 구결과 왼 구결을 아래와 위에 배치하여 구분하였다. 오른 구결과 왼 구결은 해석의 순서에 의해 구별되는 것이지 구결자 자체 정보와는 관련이 없다. 구결을 원래의 한자로 복원하면 둘째 문장과 같다. 이를『구역인왕경』당시 독음이라 여겨지는 한글로 대응하면 셋째 문장이 되고 그것을 현대 한국어로 번역하면 넷째 문장이 된다.

여기서 주목해야 할 것은 구결자는 언제나 원 한자와 특정한 한글 음소나 음절에 대응한다는 점이다. 따라서 구결을 입력하는 방법은 두 가지가 존재한다. 하나는 한자를 호출하여 그것의 구결자를 테이블에서 찾아 입력하는 방법이다. 예를 들어 爲를 입력하여 그것에 대응하는 ソ를 입력하는 것이다. 이 방법은 한자를 입력하기 위해 '위'를 입력하여 고르고 다시 여기서 ソ를 고르는 두 개의 단계를 거치게 된다. 물론 'ソ'의 원자가 '爲'임을 모르면 입력할 수 없다는 것도 또 다른 약점이며 두 단계 모두 글자를 골라서 입력하는 방식이라는 점도 문제이다.

다른 하나는 'ᄒᆞ'를 입력하여 여기에서 구결자를 선택하는 방식이다. 이 방식은 한자를 이용한 입력보다 더 직관적이고 구결의 해석을 이미 아는 상태에서 입력하기 쉬운 방식이다. 이 방법의 단점은 구결에 대한 기본적 이해가 없는 사람은 입력하기 어렵다는 점이다. 그렇지만 셋째 문장을 입력하면서 바로바로 구결자로 변환하는 방식이기 때문에 조금만 숙달하면 구결 정보의 입력이 매우 쉬워지며, 이런 방식의 입력을 통해 사용자는 구결 정보를 더 직관적으로 이해하는 장점이 있다. 한자를 통해 입력하는 방식을 부가적으로 두면 구결 IME의 자판은 다음과 같이 설계할 수 있다.

	ㅂ/ㅸ	ㅈ/ㅹ	ㄷ	ㄱ/ㆁ	ㅅ/ㅿ	ㅐ/ㅒ	ㅔ/ㅖ	ᄼ/ᄾ	ᅎ/ᅐ	ᅔ/ᅕ
	ㅁ/ㅱ	ㄴ	ㅇ	ㄹ/ㅀ	ㅎ/ㆆ	ㅗ/ㅛ	ㅏ/ㅑ	ㅜ/ㅠ	ㅓ/ㅕ	˙/: (방점)
shift	ㅋ	ㅌ	ㅊ	ㅍ	·	ㅡ	ㅣ	종성	구결	한자

<그림 3> 구결 입력을 위한 옛한글 자판

<그림 3>은 옛한글을 입력하기 위한 변형 3벌식 자판으로 이 자판을 통해 옛한글을 자유롭게 입력할 수 있다. 구결 변환을 위한 구결 토글(위 붉은색 글쇠)을 누르면 구결 후보들이 별도의 창에 나타나고 이를 선택하면 원하는 구결자가 입력되는 방식이다.[4] 예를 들어 '며'를 입력하고 구결 토글을 누르면 화면에는 다음과 같은 후보 구결자가 나타난다.

ケ	尔	旀	弥	彌

<표 2> '며'에 해당하는 구결 글리프

여기서 필요한 구결자를 고르면 입력이 가능하다. 같은 방법으로 '具足ᄒᆞ시며'를 치고 구결 토글을 누르면 'ᄒᆞ시며'를 'ソニケ'로 입력할 수 있다. 물론 한자를 통해 구결자로 바꾸어도 마찬가지 결과를 얻으며 어떤 과정으로 입력하더라도 전체 문자표에서 구결자를 하나씩 입력하는 것보다 직관적이며 경제적이다.

이 IME에서 별도로 존재하는 구결자는 정보가 아니라 구결자의 표상 기호 즉, 글리프이다. 이 글리프는 코드에 종속되지 않고 별도의 구결 표상 테이블에 위치하며 키보드로 입력된 정보로부터 호출된다. 구결자의 표상과 정보를 분리하는 이 방식을 사용하면 구결자를 유니코드에 등재하지 않고 정보의 단위로 다룰 수 있게 된다.

2) 구결 정보는 구결의 표상 기호와 분리된 "한자+한글"의 대응쌍으로 정의한다. 구결자는 언제나 원 한자와 그 독음의 쌍을 가지고 있다. 구결 문자에 대한 정보 구조는 해당 구결자의 원 한자

4) 이 입력기는 모바일에서도 사용할 수 있도록 고안한 것이다. 아래아 한글에서 사용하려면 아래아 한글이 이 입력기를 수용하면 되고 자판은 일반적으로 사용되는 두벌식 옛글에 구글 토글을 달면 된다. 마찬가지 방법으로 워드나 파워포인트에서도 구글 입력기를 선택하여 입력할 수 있다.

와 이에 대한 독음이 하나의 쌍으로 묶인 정보 상태를 해당 구결자에 대응하는 형태로 이루어져 있다. 구결 정보 테이블은 다음과 같다.

일련번호	구결 글리프	원 한자	한글 독음	기능
001	乙	乙	을	목적격조사(NObjE)

<표 3> 구결 테이블의 정보 구조

<표 3>에서 구결 정보는 일련번호와 함께 구결 글리프 정보, 원 한자 정보, 한글 독음 정보, 그리고 기능 정보를 가진다. 기능 정보는 잉여적으로 보일 수 있으나 추후 정보 검색과 정보 정렬 과정에 필요한 정보이므로 하나의 일련번호에 이 정보를 쌍 정보로 표시하는 것이 효율적이다.

일련번호	구결 글리프	원 한자	한글 독음	기능
X	ゝ	爲	ᄒᆞ	동사어간(Vstem)
Y1	ゟ	彌	며	연결어미(VlinkE)
Y2	ホ	彌	며	연결어미(VlinkE)
Y3	𣏌	彌	며	연결어미(VlinkE)
Y4	弥	彌	며	연결어미(VlinkE)
Y5	彌	彌	며	연결어미(VlinkE)
Z1	ハ	只	기	강세어미(VEmE)
Z2	ハ	果	과	공동격(NcomE)

<표 4> 구결 정보 테이블의 예시

<표 4>에서 기능 정보에 사용된 영문 약자는 이해를 돕기 위한 것이고, 테이블 정보를 채우는 여러 약어는 학계가 의견을 모아야 할 부분이다. <표 4>에서 보듯 일련번호를 포함하여 모든 정보가 완전히 일치하는 정보 행은 존재하지 않기 때문에 이 테이블 정보는 관계형 데이터베이스를 구축하는 기본 자료가 된다.

유의해야 할 점은 구결 글리프는 코드에 대응하지 않는다는 사실이다. 이 테이블에서 코드에 대응하는 것은 오직 원 한자+한글 독음의 대응쌍뿐이다. 한자와 한글은 모두 유니코드의 정규 코드 포인트를 배정받고 있으므로, 이 두 쌍은 언제나 정형 데이터의 특정 값을 가지게 된다. Y1-Y5까지의 구결 글리프는 '彌 +며'에 대응하기 때문에 이것으로 표상되는 5개의 구결 글리프는 데이터베이스에서 하나의 기능 값을 갖게 된다. Z1과 Z2는 대응하는 원 한자와 한글 독음이 달라 다른 데이터로 분류할 수 있으므로, 같은 꼴의 글리프에 대응하더라도 데이터에서는 별개의 정보로 간주한다.

여기서 다시 생각해 볼 문제는 바로 왼 구결과 오른 구결의 표현상 문제이다. 제시한 구결 테이블에는 왼 구결과 오른 구결의 정보가 존재하지 않는다. 이것은 구결의 위치 정보 자체는 구결자의 정보와 무관하기 때문이다. 그것은 마치 구결 글리프가 정보 단위로 대응하지 않는 것과 같다. 왼

구결과 오른 구결은 저작 도구의 입력과 출력에서 구분할 수 있으며, ZWJ(Zero Width Joiner, 폭이 없는 결합자)를 사용하면 그 구분이 가능하다. ZWJ는 아랍 문자나 인도계 문자처럼 문자의 형태가 주변 문자와의 조합에 따라 달라지는 복잡한 문자에서, 원래는 합쳐지지 않는 문자들을 서로 결합하여 하나의 형태로 표시하는 기능을 담당한다. 예를 들어, 데바나가리에서 자음 र (라) 뒤에 할란트와 ZWJ, 그리고 다른 자음이 오면, '라'가 다음 자음 위에 작은 갈고리 모양(र)으로 표기되는 것을 막고 다음과 같이 다른 형태의 결합을 만든다. 일반적인 '라' 합자(레프)는 र (라) + ् (할란트) + य (야) → र्य와 같이 되나, ZWJ를 사용한 합자는 र (라) + ् (할란트) + ZWJ + य (야) → र् य와 같이 출력 글리프가 달라진다.

이것을 구결에 응용해 보자. 구결 IME에서 구결 토글을 눌러 구결 글리프를 선택하면 ZWJ가 ZWJ+ 爲+ ᄒᆞ로 입력되어 오른 구결을 표기하고 구결 토글을 다시 누르면 ZWJ가 爲+ ᄒᆞ+ ZWJ처럼 오른쪽에 삽입되어 왼 구결을 표시한다. 이와 같은 구별은 가로쓰기에서 오른 구결은 한자의 아래 첨자로 왼 구결은 위첨자로 표시하여 수행된다. 물론 이 정보는 출력에서만 사용되고 정보 저장에서는 오른 구결과 왼 구결을 구별하지 않는다. 결과 <그림2>의 원문은 다음과 같이 출력된다.

■ 信行ㄴ 具足ゝニゟ 復ゝ 有ヒナゟ 五道ヒ 一切衆生ㅣ

4. 통합 구결 환경의 아키텍처

이제까지 구결 IME를 통해 구결을 입출력하고 이를 정보화하는 이론적 방안에 대해 살펴보았다. 구결 IME를 사용하면 어떠한 OS나 디지털 기기 혹은 저작 도구를 사용하더라도 구결을 쉽게 입력할 수 있다. 물론 이를 위해 필요한 절차가 있다. 윈도우나 MacOS 등의 운영 체제, 그리고 한컴오피스나 MS 오피스에 구결 IME를 채택하도록 하는 것이다. 이것은 국립국어원과 구결학회가 힘을 합하면 어렵지 않게 진행할 수 있을 것으로 보인다.[5] 물론 그 전에 윈도우 환경에서 구동하는 구결 IME를 개발해야 한다.

구결 IME와 지능형 구결 글리프를 구결 IME를 설치할 때 자동으로 설치하도록 하면 전용 구결 IME 저작 도구를 통해 구결을 자유롭게 입출력하고 구결을 정보 단위로 보존할 수 있다.[6] 해외에서도 구결 IME를 설치하면 똑같은 환경에서 구결의 연구 결과를 공유할 수 있다. 구결 IME를 통해 입력한 정보는 언제나 '원 한자+ 한글'의 쌍으로 저장되기 때문에 정보의 검색과 정렬이 보장되며, 한글이나 한자를 통해 원하는 구결 글리프의 원본 디지털 정보를 데이터로 사용할 수 있다.

이러한 관점에서 제안하는 구결 IME는 단순한 입력기를 넘어 디지털 인문학의 원거리 독서가 가능한 자유로운 구결 입출력 및 정보화의 통합 환경을 지향한다. 여기서 구결 글리프와 구결 정보

5) 구결 아키텍처를 일반적인 저작 도구(아래아 한글, 워드, 파워포인트 등)에 통합하는 일은 개개의 기술 회사들이 수용 여부를 결정하면 되는 것이기 때문에 유니코드 위원회와 같은 국제기구에서 여러 나라의 대표들이 코드 수용 여부를 결정하는 것과 같은 까다로운 절차가 필요 없다. 특히 아래아 한글에는 옛한글과 구결자가 이미 존재하므로, 구결 아키텍처를 위한 API를 제공하고 시스템이 이를 수용하기만 하면 구결 IME를 쉽게 쓸 수 있다.

6) 지능형 폰트란 고급 오픈타입 기능을 활용하여 내부 규칙(GPOS, GSUB)을 통해 문맥을 해석하고 글자의 위치와 형태를 스스로 결정하는 동적 시스템을 의미한다. 즉 '원 한자+한글 독음' 쌍 정보를 하나의 구결 글리프로 치환하는 폰트 시스템이다.

가 이원화되고 이 둘은 하나의 아키텍처 안에서 별개의 모듈로 작동한다. 즉 통합 구결 환경(UGE, Unificated Gugyeol Environment)은 단일 소프트웨어나 개별 기술의 집합이 아니다. 이는 구결 데이터의 생성, 렌더링, 보존, 그리고 활용에 이르는 전 과정을 유기적으로 지원하는 총체적 디지털 생태계를 지칭한다. 따라서 통합 구결 환경은 디지털 인문학(Digital Humanities)이 지향하는 원거리 독서, 즉 텍스트를 단순한 읽기 대상이 아닌 계산과 분석이 가능한 구조화된 데이터로 전환하여 새로운 지식을 창출하기 위한 환경으로, 구결의 데이터 정형화가 궁극적인 목표가 된다. 이를 위하여 텍스트 인코딩 이니셔티브(TEI, Text Encoding Initiative)와 같은 국제 표준을 기반으로 구결 입력 및 출력, 정보의 교환 환경을 하나의 아키텍처로 통합하려는 것이다. 이 아키텍처는 아래 3부분의 모듈로 구성된다.

구성 요소	핵심 기술	주요 전략적 이점
1. 지능형 폰트	오픈타입 GPOS 'Mark-to-Base'[7]	비표준 문자 없이 완벽한 시각적 정확성을 구현하여 범용적 호환성 보장
2. 직관적 입력기	윈도우 TSF[8] 마스터 데이터베이스	기술 지식 없이도 누구나 쉽고 정확하게 구결을 입력할 수 있는 사용자 경험 제공
3. 영속적 데이터	TEI-XML 스키마[9]	데이터의 미래 보장, 검색 가능성, 상호운용성, 의미 정보 보존을 통한 학술적 가치 영구화

<표 5> 구결 통합 환경을 위한 아키텍처 구성 요소

이제까지 구결 입력은 정형 데이터로 이용할 수 없는 PUA 영역의 코드 포인트를 이용하였다. 물론 액셀과 같은 계산 시트를 이용하여 각각의 데이터를 분리할 수 있지만, 이러한 방식의 데이터 이용은 관계형 데이터베이스를 통한 검색 및 정렬과는 비교할 수 없는 수준이다. 3개의 모듈로 구성하는 데 대한 타당성은 구결 계층을 다음 3단계로 나누어 관찰하면 분명해진다.

7) 오픈타입(OpenType) GPOS는 오픈타입 글꼴에서 문자(글리프)의 위치를 조절하여 더욱 정교한 조판을 가능하게 하는 테이블(table)을 말한다. GPOS는 'Glyph Positioning'의 약자로, 글꼴의 모양을 바꾸지 않고도 글자 간 간격(kerning) 조정, 받침과 모음의 결합, 특정 언어에 맞는 서식 등 다양한 타이포그래피 기능을 수행하게 된다.

8) 윈도우 TSF는 텍스트 서비스 프레임워크(Text Services Framework)의 약자로, 고급 텍스트 입력 및 처리를 지원하는 Windows 시스템의 API를 말한다. 이 프레임워크는 텍스트 서비스를 위한 기반을 제공하며, 음성 인식, 필기 인식, 다양한 언어 입력 방식과 같은 기능을 지원한다. 다른 OS를 사용할 경우 그에 해당하는 다른 API가 적용된다. 이 논의에서는 일반적으로 사용하는 윈도우 환경에서 구동하는 아키텍처를 기본으로 설명하기 위해 이 API를 이용하는 것으로 설정하였다.

9) 텍스트 인코딩 이니셔티브(TEI, Text Encoding Initiative)는 디지털 인문학 분야의 텍스트 중심 실무 커뮤니티를 말한다. 여기서 규정한 XML 스키마는 기계 판독 가능한 텍스트를 위한 표준 인코딩 방법을 정의하고 있다. 이는 XML의 확장성을 활용하여 인문학, 사회과학, 언어학 분야에서 텍스트를 구조적으로 표현하고 교환하기 위한 지침과 규칙을 제공한다. 따라서 이 스키마를 따르는 모든 정보는 정형 데이터로 생산하고 유통할 수 있다.

- 의미론 계층 (Semantic Layer)
  TEI-XML 구조인 <g ref="...">爲ᄒᆞ</g>는 'ᄒᆞ'가 '爲'에 대한 특정 유형의 구결 주석이라는 의미 관계를 명시적으로 기술한다. 이것이 데이터의 본질, 즉 '무엇인가(what it is)'에 해당한다.
- 문자 정체성 계층 (Character Identity Layer)
  XML 태그 내의 '爲'와 'ᄒᆞ'는 각각의 문자가 가진 고유한 정체성을 표준 유니코드 코드 포인트를 통해 표현한다. 이는 데이터의 가장 기본적인 구성 요소를 정의한다.
- 표현 계층 (Presentation Layer)
  오픈타입 폰트 파일 내의 GPOS 규칙은 텍스트 렌더링 엔진에 'ᄒᆞ'의 글리프를 '爲'의 글리프에 대해 시각적으로 어떻게 배치해야 하는지를 지시한다. 이것은 데이터가 '어떻게 보이는가(what it looks like)'에 해당한다.

UGE 아키텍처의 설계 목적은 PUA 모델에서 파괴적으로 융합되었던 정보 계층을 체계적으로 분리(decoupling)하는 데 있다. PUA 방식에서는 하나의 비표준 코드 포인트(U+E001)가 원본 한자, 구결의 음가, 그리고 시각적 위치 정보까지 모든 것을 암시적으로 포함하고 있어 어느 하나도 분리할 수 없었다. UGE는 이 융합된 덩어리를 세 개의 독립적이고 표준화된 계층으로 해체한다. 이러한 체계적인 분리는 데이터 아키텍처의 유연성과 견고성을 극대화한다. 예를 들어 미래에 더 나은 시각적 표현을 위해 폰트 파일(표현 계층, 즉 글리프)을 교체하더라도, 데이터의 핵심인 TEI-XML 파일(의미론 및 문자 정체성 계층)은 전혀 영향을 받지 않는다. 또한, TEI-XML 데이터를 웹페이지용 HTML이나 데이터 분석용 JSON으로 변환하는 작업 역시 표현 계층과는 무관하게 이루어질 수 있다. 이처럼 각 계층이 독립적으로 발전하고 교체될 수 있도록 설계함으로써 데이터의 미래 가치를 보장하는 데 도움을 줄 수 있다.

UGE 폰트의 핵심 기술은 오픈타입 사양의 GPOS(Glyph Positioning) 테이블에 정의된 'Mark-to-Base Attachment'(GPOS Lookup Type 4) 기능이다. 이 기능은 아랍어의 모음 부호, 베트남어의 성조 부호, 히브리어의 니쿠드 등 전 세계의 다양한 문자 체계에서 베이스 문자에 결합 표시를 정확하게 부착하기 위해 설계된 표준 기술이다. 작동 원리는 다음과 같다.

- 독립된 글리프 설계
  기존 방식처럼 '爲+ᄒᆞ'와 같이 한자와 구결을 미리 합쳐놓은 수천 개의 합자 글리프를 만드는 대신, UGE 폰트는 모든 구결자(예: 'ᄒᆞ', '이', '고' 등)를 원본 한자 크기의 약 1/4 크기로 디자인된 독립적인 '마크(mark)' 글리프로 포함한다.[10] 이 마크 글리프들은 모두 표준 유니코드 옛한글 코드 포인트에 할당된다.
- 앵커 포인트(Anchor Point) 정의
  폰트 디자이너는 각 베이스 글리프(모든 한자)와 마크 글리프(모든 구결자)에 '앵커 포인트'라는 보이지 않는 연결점을 정의한다. 예를 들어, '爲'라는 한자 글리프의 우측 하단에 right_bottom이라는 이름의 앵커를 설정한다. 그리고 'ᄒᆞ'라는 구결자 글리프에는 이에 대응하는 _right_bottom이라는 이름의 앵커를 설정한다.[11]

10) 한문 원본의 글자 크기를 10포인트라고 할 때 구결 글리프는 5.5 포인트에 대응한다. 이로써 구결자를 한문 옆에 붙였을 때 약 1/4 크기의 아래첨자나 위첨자처럼 보이게 한다.

11) 폰트 구현에서 '앵커(anchor)'는 주로 특정 글자(기본 글리프)에 다른 기호나 글리프(합자, 분음부호 등)를 정확하게 연결하고 배치하기 위해 사용되는 특별한 지점을 의미한다. 이를 통해 다양

- 동적 정렬

  사용자가 '爲ᄒᆞ'라는 텍스트를 입력하면, 운영 체제의 텍스트 렌더링 엔진은 UGE 폰트의 GPOS 규칙을 읽어 들인다. 이 규칙은 "'爲' 다음에 'ᄒᆞ'가 오면, '爲'의 right_bottom 앵커와 'ᄒᆞ'의 _right_bottom 앵커를 서로 일치하라"고 지시한다. 이 지시에 따라 렌더링 엔진은 'ᄒᆞ' 글리프를 '爲' 글리프의 우측 하단에 정확하게 동적으로 배치한다.[12]

UGE 아키텍처의 주요한 목표는 '데이터의 영속성'을 확보하는 것이다. 디지털 기술은 끊임없이 변하지만, 학술 데이터의 가치는 영원해야 하기 때문이다. 따라서 디지털 인문학 분야의 국제 표준으로 확고히 자리 잡은 '텍스트 인코딩 이니셔티브(Text Encoding Initiative, TEI)'의 XML 스키마를 따르는 데이터 표준은 어떤 디지털 환경에서도 데이터 정보를 생산, 유통할 수 있는 최선의 방안이라고 할 수 있다.[13]

UGE 입력기(구결 IME)가 생성하는 최종 데이터는 단순 텍스트가 아니라, 다음과 같은 TEI-XML 형식의 구조화된 데이터 조각이다: <g ref="#g-ha-01">爲ᄒᆞ</g>. 이 구조는 TEI 가이드라인을 준수하며 다음과 같은 장점을 보여준다.

- 시각과 의미의 완벽한 분리: TEI는 '추상적 문자(abstract character, 의미와 음가를 가진 단위)'와 '글리프(glyph, 문자의 시각적 형태)'를 엄격하게 구분하는 것을 핵심 원칙으로 삼는다. <g> 태그는 화면에 표시되는 '爲+ ᄒᆞ'의 조합을 하나의 특수한 '글리프'로 정의한다. 태그 내부의 텍스트 '爲ᄒᆞ'는 이 글리프를 구성하는 '추상적 문자'들의 나열이다. 이 덕분에 UGE 전용 폰트가 없는 환경에서 이 데이터를 열면, 화면에는 그냥 '爲ᄒᆞ'라는 일반 텍스트로 보일 뿐 데이터가 소실되지 않는다. XML 태그 <g>와 ref 속성이 '이것은 爲에 대한 ᄒᆞ라는 구결 주석이다'라는 핵심적인 의미 정보를 영구적으로 보존하기 때문이다.
- 궁극의 검색 가능성: 이 구조는 "한글 입력으로 데이터를 검색하고 싶다"는 구결 연구자들의 핵심 요구사항을 완벽하게 해결한다. 태그 <g> 안에 구결의 한글 음가인 'ᄒᆞ'가 일반 텍스트로 명시적으로 포함되어 있기 때문에, 사용자는 문서 전체에서 'ᄒᆞ'라는 한글 키워드로 검색하여 해당 구결이 사용된 모든 용례를 정확하고 완벽하게 찾아낼 수 있다. 이는 PUA 방식에서는 원천적으로 불가능했던 기능으로, 대규모 말뭉치 분석을 위한 필수 전제 조건이다.
- 데이터의 청결성과 상호운용성: 최종 저장되는 XML 데이터에는 렌더링을 위해 임시로 사용되었던 ZWJ와 같은 제어 문자가 전혀 포함되지 않는다. 따라서 저장된 데이터는 불필요한 코드 없

한 언어의 복잡한 글자 조합을 일관성 있고 정확하게 표현할 수 있다. right_bottom은 아래첨자처럼 보이는 오른 구결을 위한 표기 규약이다.

12) GPOS는 OpenType 레이아웃 테이블 중 GSUB(Glyph Substitution) 테이블과 함께 사용된다. GSUB가 특정 상황에서 글리프를 다른 글리프로 대체(예: 합자 생성)하는 역할을 한다면, GPOS는 글리프의 위치를 조정하는 역할을 하게 된다.

13) TEI는 1987년부터 인문학 텍스트를 전자적으로 표현하고 교환하기 위한 표준 가이드라인을 개발해 온 국제 컨소시엄이다. TEI가 제공하는 XML 기반의 마크업 언어는 텍스트의 구조, 의미, 외형적 특징 등을 풍부하고 정밀하게 기술할 수 있어, 전 세계의 수많은 디지털 아카이브와 학술 편집 프로젝트의 표준으로 사용되고 있다. 과거 서구를 중심으로 발전해 온 TEI는 동아시아 고유의 복잡한 텍스트적 특징(예: 세로쓰기, 루비 주석)을 다루는 데 한계가 있다는 지적이 있었다. 최근 '동아시아/일본 특별 관심 그룹(SIG East Asian/Japanese)'의 활동 등을 통해 이러한 한계를 극복하려는 노력이 활발히 이루어지고 있다. 일본의 와카(和歌) 데이터 구축이나 고전 텍스트의 구조화에 TEI를 적용하는 사례가 늘고 있으며, 이는 TEI가 진정한 글로벌 표준으로 발전하고 있음을 보여준다.

이 깨끗한 상태로 보존되어, 향후 데이터 처리 및 분석 과정에서 발생할 수 있는 오류 가능성을 최소화한다. 또한, TEI-XML은 XSLT(Extensible Stylesheet Language Transformations)와 같은 표준 기술을 통해 다른 형식으로 매우 쉽게 변환할 수 있다. 예를 들어, 동일한 TEI-XML 소스 데이터를 웹페이지에 표시하기 위한 HTML(爲ᄒᆞ)이나, 데이터 분석 파이프라인에서 사용하기 위한 JSON({"base": "爲", "reading": "ᄒᆞ", "ref": "#g-ha-01"})으로 손쉽게 변환하여 재활용할 수 있다.

결론적으로, TEI-XML을 데이터의 영속적 형식으로 채택한 것은 구결 데이터를 일회성 시각적 결과물이 아닌, 영구적이고 학술적인 가치를 지니는 디지털 자산으로 만드는 가장 확실한 방법이다.

5. 결론

그동안 구결 연구는 아래아 한글이 제공하는 PUA(사용자 정의 영역)에 의존함으로써 데이터 정형화 및 구결 입력의 편의성 등에서 디지털 시대에 맞지 않는 문제를 안고 진행되었다. 이 문제의 핵심은 당연히 PUA 코드를 버리고 유니코드 다국어 기본 평면이나 보충 평면에 구결 글리프를 독립적 코드 포인트를 배정받는 일로 귀결될 수 있다. 그러나 현 상황에서 이 문제의 해결은 그리 간단하지 않다. 기존의 구결 디지털화 방식이 단기적인 시각적 편의, 즉 아래아 한글이 제공하는 PUA를 통한 접근에 의존하여 데이터의 검색, 상호운용, 그리고 영속적 데이터 저장을 외면하였다. 이러한 접근은 구결이라는 지식 유산 혹은 한글 이전의 언어 상황을 재구하기 위한 문자 정보를 미래 세대가 활용할 수 없는 '디지털 박제'로 전락시키는 결과를 초래했다.

통합 구결 환경을 위한 아키텍처는 구결 데이터를 살아있는 지식 자원으로 재탄생시키기 위한 명확한 청사진을 제시하기 위한 것이다. UGE의 핵심은 지능형 오픈타입 폰트, 직관적인 TSF 입력기, 그리고 영속적인 TEI-XML 데이터 스키마라는 세 가지 구성 요소의 유기적인 결합에 있다. 이 통합 솔루션은 시각적 표현과 의미론적 내용을 완벽하게 분리하는 현대 데이터 아키텍처의 원칙을 충실히 따르며, 유니코드, 오픈타입, TEI-XML 등 검증된 국제 표준 기술만을 사용하여 기술적 종속성을 배제하고, 데이터의 미래 가치를 극대화한다. 또한 이러한 아키텍처를 구동하기 위한 기술적 복잡성은 사용자에게 아무런 영향을 끼치지 않고 단순히 옛한글 입력+구결 토글의 조합으로 구결자를 입력할 수 있게 한다. 따라서 구결 연구자 혹은 학부의 학생들도 누구나 손쉽게 구결 데이터를 생산할 수 있는 길을 열어준다.

또한 통합 구결 환경은 '근접 독해'를 넘어서, 방대한 말뭉치의 거시적 패턴을 분석하는 디지털 인문학이 지향하는 '원거리 독해'가 가능하도록 한다. 일반적인 검색은 물론, 데이터 정렬이 다층적이고 유기적으로 손쉬운 사용자 화면(UI)에서 기능하는 구결 통계 시스템을 제작, 배포할 수 있다. 이 시스템이 세종 이후 생산된 옛한글 문헌의 통계 분석과 접목된다면, 중세 및 고대 한국어의 어휘, 문법, 어휘의 분포를 여러 각도에서 재조명할 수 있다.

통합 구결 환경을 위한 아키텍처를 구축하기 위해서는 구결학회 회원들이 3장에서 제기한 구결 테이블을 완비하고, 연구 재단의 공동 연구 예산을 확보하는 것이 필요하다. 구결 아키텍처 개발을 위한 구결 테이블을 제대로 구축한다면, 이를 통해 유니코드 다국어 평면에 구결자를 등록하는 것

도 수월해진다. 그런데 유니코드 등록이 되었다고 구결자를 제대로 입력하고 출력하며, 데이터를 정보화하는 일이 해결되는 것이 아니다. 이 발표에서 제안하는 구결 아키텍처를 성공적으로 구축하는 것이 구결 정보화와 관련된 모든 문제를 해결하는 방안임을 강조하겠다.