G-machine과 ZG-machine의 그래프 합동*

(*The Graph Congruence Between the G-machine and the ZG-machine*)

우 균, 한태숙

동아대학교 전기전자컴퓨터공학부, 한국과학기술원 전자전산학과

woogyun@mail.donga.ac.kr, han@cs.kaist.ac.kr

요 약

이 논문은 지연 함수형 언어를 위한 추상기계 G-machine과 ZG-machine의 동작이 동등함에 대하여 논한다. ZG-machine은 공간 효율을 높이기 위해 태그 옮김이라는 기법을 통해 G-machine을 변형한 추상기계이다. ZG-machine에서는 그래프 생성 방법을 변경하였으므로 해석 방법도 G-machine과 다르게 되는데, 그럼에도 불구하고 G-machine과 ZG-machine은 같은 의미의 동작을 수행한다는 것을 보이고자 한다. 이를 위해 이 논문에서는 두 추상기계 수행시 메모리에 저장되는 그래프 형태를 의미 도메인상의 원소로 기술하고, 같은 프로그램에 대하여 두 추상기계에서 상태 변환 과정 중의 그래프 형태가 같은 의미라는 것을, 즉 그래프 합동이라는 것을 보인다.

1. 서론

현재 널리 사용되고 있는 폰 노이만 방식의 컴퓨터 구조와 의미 격차가 큰 언어는, 이러한 의미 격차를 해소하기 위해 다양한 추상기계를 제시하고 있다. 함수형 언어도 이런 범주의 언어에 속한다. 그러나 정작 제시된 추상기계가 언어의 의미를 제대로 보존하는지에 대해서 연구한 결과는 많지 않다.

이 논문은 지연 함수형 언어를 위한 추상기계 ZG-machine[1, 2, 3]의 올바름에 대하여 논한다. ZG-machine은 태그 옮김이라는 부호화 기법을 통해 공간 효율을 개선한 G-machine[4, 5]이다. ZG-machine의 올바름은 G-machine의 올바름에 의해 기술될 수 있는데 G-machine의 올바름에 대해서는 일찌기 Lester에 의해 연구된 바[6, 7] 있다. 이 논문에서는 Lester가 제안한 G-machine의 올바름에 의거하여 ZG-machine의 올바름을 증명하고자 한다.

Lester는 의미 도메인 상에서 G-machine의 스택 의미정의(stack semantics)를 정의함으로써 G-machine이 올바르다는 것을 증명하였다. 스택 의미정의란 스택이 명시적으로 보이도록 한 의미정의를 말하는데, Lester가 정의한 G-machine 스택 의미정의는 스택 뿐만 아니라 G-machine의 상태를 구성하는 그래프 저장소, 출력값 등도 명시적으로 표현하고 있다. Lester는 주어진 프로그램의 결과가 G-machine의 스택 의미정의에서 최종 상태의 출력값과 같음을 보임으로써 G-machine이 올바르다는 것을 증명하였다.

ZG-machine은 G-machine의 변형이므로 ZG-machine이 올바르다는 것은 G-machine의 스택

*본 논문은 2002년도 정보통신부 IT관련학과 장비지원 사업의 동아대학교 대응자금에 의해 연구되었음.

의미정의에 의거하여 증명할 수 있다. 이 논문에서 논하고자 하는 부분을 도시하면 그림 1에서와 같다.

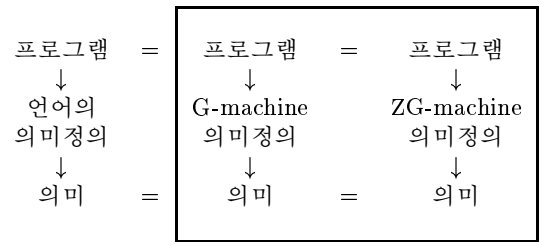

그림 1: 올바름 증명 범위

본 논문의 구조는 다음과 같다. 먼저 2절에서는 두 추상기계의 입력 언어 구문을 정의한다. 3절에서는 G-machine의 스택 의미정의를 기술하고, 4절에서는 ZG-machine의 스택 의미정의를 기술한다. 마지막으로 5절에서 이들 스택 의미정의에 따른 프로그램의 의미가 같다는 것을 그래프 합동(graph congruence) 관계에 입각하여 논한다. 끝으로 6절에서 결론을 맺는다.

2. 언어 정의

이 절에서는 G-machine과 ZG-machine의 입력으로 주어질 프로그램을 정의하는 입력 언어 구문을 정의한다. 여기서 기술할 언어는 함수형 언어의 틀을 갖춘, 아주 간단한 형태이다. 입력 언어의 구문 정의는 그림 2에 나타나 있다.

구문 정의에 따르면, 프로그램 P는 하나 이상의 완전조합자(supercombinator) 정의 D로 이루어져 있고, 각 완전조합자 정의 D는 완전조합자 C에 이름 I를 붙임으로써 이루어진다. 각 완전조합자 C는 필요한 만큼의 형식 인수 I를 가질 수 있는데, 형식 인수는 람다 기호(λ)를 앞에 붙여서 나타낸다. 완전조합자 몸체에 해당하는 표현식 E는 이름 I나, 정수값 Z, 또는 임의의 표현식 E_0를 임의의 표현식 E_1에 적용한 결과 등을 가질 수 있다. 여기서 이름 I는 형식 인수를 지칭할 수도 있고, 다른 완전조합자를 지칭할 수도 있다.

3. G-machine의 스택 의미정의

G-machine의 스택 의미정의에 필요한 의미 도메인은 그림 3에 나타나있다. G-machine은 일련의 상태 변환을 통해 프로그램을 수행하는 추상기계이므로 따라서 상태에 대한 의미 도메인이 필요한데, 이를 **S**로 정의하였다. 도메인 **S**는 도메인 **O**, $\mathbf{L}^*$, **G**, **U**, **D** 등 다섯 도메인의 곱 도메인으로 각 부분 도메인은 각각 출력값, 스택, 그래프 함수, 환경 함수, 덤프 스택을 나타낸다. 출력값 도메인 **O**는 정수 도메인의 리스트 도메인 $\mathbf{Z}^*$이다.

그래프 함수 도메인 **G**는 레이블 도메인 **L**에서 그래프 노드 도메인 **N**으로의 함수 도메인이다. 그래프 노드 도메인 **N**은 **A**, **I**, `Id`, **Z** 등 네 도메인의 합 도메인으로, 각 부분 도메인은 각각 적용(application) 노드, 우회(indirection) 노드, 이름 노드, 정수값을 나타낸다. 적용 노드 도메인 **A**는 두 개의 레이블 도메인의 곱 도메인인데, 각 레이블은 적용할 함수와 그의 인수를 가리킨다. 우회 노드 도메인 **I**는 다른 노드를 가리키는 우회 노드를 나타내는데, 레이블 도메인 **L**로 정의한다. 이름 노드 도메인 `Id`는 그림 2(a)의 구문 도메인 `Id`를 그대로 사용하였다.

환경 도메인 **U**는 완전조합자 이름에 대하여 대응하는 상태 변환 함수를 반환하는 함수 도메인이고, 상태 변환 함수를 포함하는 도메인 **K**는 상태 도메인 **S**에서 같은 도메인 **S**로의 함수 도메인이다. 스택 도메인은 레이블 도메인의 리스트 도메인 $\mathbf{L}^*$로 정의되고, 덤프 스택 도메인은 스택 도메인의 리스트, 즉 레이블의 리스트의 리스트 도메인 $\mathbf{L}^{**}$로 정의된다.

스택 의미정의를 기술할 때, 완전조합자의 인수에 대하여 스택 최상위로부터의 위치를 나타내기 위해 바인딩이 사용되는데, 바인딩은 이름 도메인 `Id`로부터 자연수 도메인 $\mathbf{Z}_+$으로의 함수 도메인으로 정의된다. 자연수 도메인 $\mathbf{Z}_+$은 자연수 집합이 들어 올려진 도메인(lifted domain)이다. 이상에서 설명한 의미 도메인을 기초로 G-machine의 스택 의미정의를 기술하면 그림 4에서와 같다.

G-machine의 스택 의미정의에서 프로그램의

P	$\in$	`Prog`	(프로그램)
D	$\in$	`Defs`	(완전조합자 정의)
C	$\in$	`Comb`	(완전조합자)
E	$\in$	`Expr`	(표현식)
Z	$\in$	`Int`	(정수값)
I	$\in$	`Id`	(이름)

(a) 구문 도메인

P	::=	D
D	::=	$I = C$
	|	$I = C; D$
C	::=	$\lambda I\ .\ C$
	|	E
E	::=	$I \mid Z \mid E_0\ E_1$

(b) 구문 정의

그림 2: 언어 구문 정의

σ	$\in$	$\mathbf{S}$	$=$	$\mathbf{O} \times \mathbf{L}^* \times \mathbf{G} \times \mathbf{U} \times \mathbf{D}$	(상태)
o	$\in$	$\mathbf{O}$	$=$	$\mathbf{Z}^*$	(출력값)
γ	$\in$	$\mathbf{G}$	$=$	$[\mathbf{L} \rightarrow \mathbf{N}]$	(그래프 함수)
ν	$\in$	$\mathbf{N}$	$=$	$\mathbf{A} + \mathbf{I} + \mathtt{Id} + \mathbf{Z}$	(그래프 노드)
		$\mathbf{A}$	$=$	$\mathbf{L} \times \mathbf{L}$	(적용 노드)
		$\mathbf{I}$	$=$	$\mathbf{L}$	(우회 노드)
		$\mathbf{Z}$	$=$	$\{\ldots, -2, -1, 0, 1, 2, \ldots\}_\perp$	(정수값)
ρ	$\in$	$\mathbf{U}$	$=$	$[\mathtt{Id} \rightarrow \mathbf{K}]$	(환경)
δ	$\in$	$\mathbf{D}$	$=$	$\mathbf{L}^{**}$	(덤프)
ϕ	$\in$	$\mathbf{L}^*$			(스택)
κ	$\in$	$\mathbf{K}$	$=$	$[\mathbf{S} \rightarrow \mathbf{S}]$	(상태 변환 함수)
β	$\in$	$\mathbf{B}$	$=$	$[\mathtt{Id} \rightarrow \mathbf{Z}_+]$	(바인딩)
		$\mathbf{Z}_+$	$=$	$\{0, 1, 2, \ldots\}_\perp$	(0을 포함한 자연수)
ℓ	$\in$	$\mathbf{L}$			(노드 주소 레이블)

그림 3: G-machine 스택 의미정의를 위한 의미 도메인

$$
\begin{array}{lcl}
\mathcal{P} : \texttt{Prog} \rightarrow \mathbf{S} & & \\
\mathcal{P}\ [\![P]\!] & = & (\underline{print} \circ \underline{eval} \circ (\rho\ [\![\texttt{main}]\!]))\ ([],\ [],\ \{\},\ \rho,\ []) \\
 & & \text{where } \rho = \rho_{init} \oplus \mathcal{D}\ [\![P]\!] \\
 & & \\
\mathcal{D} : \texttt{Defs} \rightarrow \mathbf{U} & & \\
\mathcal{D}\ [\![D_0\ ;\ D_1]\!] & = & \mathcal{D}\ [\![D_0]\!] \oplus \mathcal{D}\ [\![D_1]\!] \\
\mathcal{D}\ [\![I = C]\!] & = & \{I \mapsto \mathcal{F}\ [\![C]\!]\ \{\}\ 0\} \\
 & & \\
\mathcal{F} : \texttt{Comb} \rightarrow \mathbf{B} \rightarrow \mathbf{Z}_+ \rightarrow \mathbf{K} & & \\
\mathcal{F}\ [\![\lambda I\ .\ C]\!]\ \beta\ n & = & \mathcal{F}\ [\![C]\!]\ (\beta \oplus \{I \mapsto n\})\ (n+1) \\
\mathcal{F}\ [\![E]\!]\ \beta\ n & = & \underline{entry}\ n\ (\mathcal{R}\ [\![E]\!]\ \beta\ n) \\
 & & \\
\mathcal{R} : \texttt{Expr} \rightarrow \mathbf{B} \rightarrow \mathbf{Z}_+ \rightarrow \mathbf{K} & & \\
\mathcal{R}\ [\![E]\!]\ \beta\ n & = & \underline{exit}\ n \circ \mathcal{C}\ [\![E]\!]\ \beta \\
 & & \\
\mathcal{E} : \texttt{Expr} \rightarrow \mathbf{B} \rightarrow \mathbf{K} & & \\
\mathcal{E}\ [\![E]\!]\ \beta & = & \underline{eval} \circ \mathcal{C}\ [\![E]\!]\ \beta \\
 & & \\
\mathcal{C} : \texttt{Expr} \rightarrow \mathbf{B} \rightarrow \mathbf{K} & & \\
\mathcal{C}\ [\![I]\!]\ \beta & = & I \in dom(\beta) \longrightarrow \underline{push}\ (\beta\ [\![I]\!]),\ \underline{pushval}\ ([\![I]\!]\ \mathbf{in}\ \mathbf{N}) \\
\mathcal{C}\ [\![Z]\!]\ \beta & = & \underline{pushval}\ (\mathcal{Z}\ [\![Z]\!]\ \mathbf{in}\ \mathbf{N}) \\
\mathcal{C}\ [\![E_0\ E_1]\!]\ \beta & = & \underline{mkap} \circ \mathcal{C}\ [\![E_0]\!]\ (inc \circ \beta) \circ \mathcal{C}\ [\![E_1]\!]\ \beta \\
 & & \\
\mathcal{Z} : \texttt{Int} \rightarrow \mathbf{Z} & & \text{(자세한 정의 생략)}
\end{array}
$$

그림 4: G-machine의 스택 의미정의

의미는 의미 함수 $\mathcal{P}$로 정의되는데, 의미 함수 $\mathcal{P}$는 프로그램 구문 P를 받아서 이를 G-machine에서 수행한 결과의 상태를 반환한다. 의미 함수 $\mathcal{F}$는 완전조합자 구문 C와 인수 바인딩 β, 인수 개수 n을 받아서 완전조합자 C에 해당하는 상태 변환 함수를 반환한다. 의미 함수 $\mathcal{R}$은 완전조합자 몸체에 대한 표현식 구문 E와 인수 바인딩 β, 인수 개수 n을 받아서 상태 변환 함수를 반환한다. 의미 함수 $\mathcal{C}$는 표현식 구문 E와 이에 대한 자유 변수 바인딩 β를 받아서 E에 해당하는 그래프 형태를 만드는 상태 변환 함수를 반환한다.

의미 함수 $\mathcal{C}$의 인수가 이름 I이고 I가 $dom(\beta)$에 속해 있는 경우, 이 이름은 형식 인수를 나타내므로 이에 대한 스택의 위치를 스택에 넣는 상태 변환 함수 $\underline{push}\ (\beta\ [\![I]\!])$를 반환한다. 이름 I가 $dom(\beta)$에 속해 있지 않은 경우, 이 이름은 다른 완전조합자 이름으로 취급되어 그 이름에 해당하는 노드를 바로 그래프 함수에 생성하도록 상태 변환 함수 $\underline{pushval}\ ([\![I]\!]\ \mathbf{in}\ \mathbf{N})$을 반환한다. 의미 함수 $\mathcal{C}$의 인수가 적용 표현식 $(E_0\ E_1)$인 경우, E_1에 대한 그래프를 생성하는 상태 변환 함수와 E_0에 대한 그래프를 생성하는 상태 변환 함수, 이 둘을 이용하여 그래프 함수에 적용 노드를 생성하는 상태 변환 함수 $\underline{mkap}$를 합성하여 그 결과를 반환한다.

이상에서 기술한 스택 의미정의에서 상태 변환 함수를 정의하기 위해 $\underline{push}$, $\underline{pushval}$, $\underline{mkap}$와 같은 함수를 사용하였는데, 이런 함수를 기본 함수(base functions)라고 한다. 기본 함수는 추상 기계 G-machine의 명령어에 대응되는 함수로, 그 정의는 그림 5에 정리되어 있다.

그림 5의 기본 함수들은 G-machine의 명령어 기능을 거의 그대로 따라하고 있다. 기본 함수 $\underline{pushval}$은 인수로 주어진 노드 ν를 그래프 함수 γ에 생성하고 그에 대한 레이블을 스택 최상위에 넣는다. 기본 함수 '$\underline{exit}\ n$'은 '$\underline{update}\ n$'과 '$\underline{pop}\ n$', $\underline{unwind}$의 합성 함수이다. 여기서 $\underline{unwind}$는 현재 스택이 가리키는 그래프를 축약시키기 위한 함수로 스택 최상위 레이블이 가리키는 대상 그래프에 따라 다르게 정의된다.

기본 함수 $\underline{eval}$은 스택 최상위 레이블이 가리키는 그래프에 대하여 축약된 형태인 최전방 정규형(WHNF: weak head normal form)을 얻어내기 위한 함수이다. 기본 함수 '$\underline{entry}\ n\ \kappa$'는 완전조합자 구문의 의미를 정의할 때 사용되는 함수로서 완전조합자의 인수 개수에 해당하는 n 만큼의 레이블이 스택 최상위를 제외한 나머지 부분에 있나 먼저 검사한 후, 있으면 그 부분의 레이블을 각 레이블이 가리키는 적용 노드의 인수로 치환한 후 상태 변환 함수 κ를 적용한다. 만약 n 만큼의 레이블이 스택에 없다면, 완전조합자의 인수가 모자라는 경우이므로 인수로 주어진 상태를 그대로 반환한다. 끝으로 함수 $\underline{print}$는 스택 최상위의 레이블이 가리키는 그래프가 정수 노드인 경우 해당 정수를 출력값의 뒷부분에 붙이는 함수이다.

이상에서 G-machine의 스택 의미정의에서 사용되는 기본 함수를 설명하였다. 기본 함수를 기술하는데 몇 가지 보조 함수가 사용되었는데, 각 보조 함수를 간단히 설명하면 다음과 같다. 보조 함수 New는 인수로 주어진 그래프 함수 γ로부터 새로운 레이블을 반환하는 함수이다. 보조 함수 Arg는 그래프 함수 γ에서 레이블 ℓ이 가리키는 노드 값을 찾아서 그 노드가 적용 노드를 나타낼 경우, 그 적용 노드의 두번째 레이블 즉 인수를 가리키는 레이블을 반환하는 함수이다. 보조 함수 $Elide$는 그래프 함수 γ에서 레이블 ℓ이 가리키는 노드 값을 찾아서 그 노드가 우회 노드를 나타내는 경우, 우회 노드가 아닐 때까지 우회 노드를 계속 따라가는 함수이다.

앞의 G-machine 스택 의미정의에서 초기 환경(initial environment)을 ρ_{init}으로 나타내었는데, ρ_{init}의 정의는 그림 6에 나타나 있다. 초기 환경 ρ_{init}는 일반 프로그래밍 언어의 라이브러리에 해당하는 것으로, 몇 가지 완전조합자에 대한 상태 변환 규칙을 지정하고 있다.

초기 환경 ρ_{init}을 정의하는데 사용된 내장 함수(built-in functions)의 정의는 그림 7에 나타나 있다. $\underline{if}$는 스택 최상위 노드 값에 따라 상태 변환 함수 κ_T 또는 κ_F를 선택하는 함수이다. $\underline{add}$는 스택 상의 두 노드의 정수 값을 합하여 결과의 정

$$
\begin{array}{lrcl}
\underline{\mathit{pushval}}\ \nu & (o,\,\phi,\,\gamma,\,\rho,\,\delta) & = & (o,\,\ell : \phi,\,\gamma \oplus \{\ell \mapsto \nu\},\,\rho,\,\delta) \\
 & & & \text{where } \ell = \mathit{New}\,\gamma \\
\underline{\mathit{push}}\ n & (o,\,\phi,\,\gamma,\,\rho,\,\delta) & = & (o,\,(\phi\,!\,n) : \phi,\,\gamma,\,\rho,\,\delta) \\
\underline{\mathit{mkap}} & (o,\,\ell_0 : \ell_1 : \phi,\,\gamma,\,\rho,\,\delta) & = & \underline{\mathit{pushval}}\ ((\ell_0,\,\ell_1)\ \mathbf{in}\,\mathbf{N})\ (o,\,\phi,\,\gamma,\,\rho,\,\delta) \\
\underline{\mathit{exit}}\ n & & = & \underline{\mathit{unwind}} \circ \underline{\mathit{pop}}\ n \circ \underline{\mathit{update}}\ n \\
\underline{\mathit{pop}}\ n & (o,\,\phi,\,\gamma,\,\rho,\,\delta) & = & (o,\,\mathit{drop}\,n\,\phi,\,\gamma,\,\rho,\,\delta) \\
\underline{\mathit{update}}\ n & (o,\,\ell : \phi,\,\gamma,\,\rho,\,\delta) & = & (o,\,\phi,\,\gamma \oplus \{(\phi\,!\,n) \mapsto \gamma\ \ell\},\,\rho,\,\delta) \\
\underline{\mathit{eval}} & (o,\,\ell : \phi,\,\gamma,\,\rho,\,\delta) & = & (\underline{\mathit{restore}} \circ \underline{\mathit{unwind}})\ (o,\,[\ell],\,\gamma,\,\rho,\,\phi : \delta) \\
\underline{\mathit{restore}} & (o,\,\phi,\,\gamma,\,\rho,\,\phi' : \delta) & = & (o,\,(\mathit{Elide}\,\gamma\ (\mathit{last}\,\phi)) : \phi',\,\gamma,\,\rho,\,\delta) \\
\underline{\mathit{entry}}\ n\ \kappa & (o,\,\ell : \phi,\,\gamma,\,\rho,\,\delta) & = & \#\phi \geq n \longrightarrow \kappa\ (o,\,\phi_a +\!\!+ \phi_r,\,\gamma,\,\rho,\,\delta), \\
 & & & \qquad\qquad (o,\,\ell : \phi,\,\gamma,\,\rho,\,\delta) \\
 & & & \text{where}\ \ \phi_a = \mathit{map}\,(\mathit{Arg}\,\gamma)\,(\mathit{take}\,n\,\phi) \\
 & & & \qquad\ \ \phi_r = n < 1 \longrightarrow \phi,\,\mathit{drop}\,(n-1)\,\phi \\
\underline{\mathit{unwind}} & (o,\,\ell : \phi,\,\gamma,\,\rho,\,\delta) & = & (\nu\ \mathtt{E}\ \mathtt{Id}) \longrightarrow \rho\ (\nu \mid \mathtt{Id})\ (o,\,\ell : \phi,\,\gamma,\,\rho,\,\delta), \\
 & & & (\nu\ \mathtt{E}\ \mathbf{I}) \longrightarrow \\
 & & & \quad \underline{\mathit{unwind}}\ (o,\,(\nu \mid \mathbf{I}) : \phi,\,\gamma,\,\rho,\,\delta), \\
 & & & (\nu\ \mathtt{E}\ \mathbf{A}) \longrightarrow \\
 & & & \quad \underline{\mathit{unwind}}\ (o,\,\mathit{fst}\,(\nu \mid \mathbf{A}) : \ell : \phi,\,\gamma,\,\rho,\,\delta), \\
 & & & \quad (o,\,\ell : \phi,\,\gamma,\,\rho,\,\delta) \\
 & & & \text{where } \nu = \gamma\ \ell \\
\underline{\mathit{print}} & (o,\,\ell : \phi,\,\gamma,\,\rho,\,\delta) & = & (\nu\ \mathtt{E}\ \mathbf{Z}) \longrightarrow (o +\!\!+ (\nu \mid \mathbf{Z}),\,\phi,\,\gamma,\,\rho,\,\delta), \\
 & & & \qquad\qquad \bot \\
 & & & \text{where } \nu = \gamma\ \ell
\end{array}
$$

그림 5: G-machine의 스택 의미정의에서 사용되는 기본 함수

$$
\begin{array}{lcl}
\rho_{init}[\![\mathtt{if}]\!] & = & \underline{\mathit{entry}}\ 3\ (\underline{\mathit{exit}}\ 3 \circ \underline{\mathit{if}}\ (\underline{\mathit{push}}\ 1)\ (\underline{\mathit{push}}\ 2) \circ \underline{\mathit{eval}} \circ \underline{\mathit{push}}\ 0) \\
\rho_{init}[\![\mathtt{add}]\!] & = & \underline{\mathit{entry}}\ 2\ (\underline{\mathit{exit}}\ 2 \circ \underline{\mathit{add}} \circ \underline{\mathit{eval}} \circ \underline{\mathit{push}}\ 1 \circ \underline{\mathit{eval}} \circ \underline{\mathit{push}}\ 0) \\
\rho_{init}[\![\mathtt{eq}]\!] & = & \underline{\mathit{entry}}\ 2\ (\underline{\mathit{exit}}\ 2 \circ \underline{\mathit{eq}} \circ \underline{\mathit{eval}} \circ \underline{\mathit{push}}\ 1 \circ \underline{\mathit{eval}} \circ \underline{\mathit{push}}\ 0)
\end{array}
$$

그림 6: G-machine의 스택 의미정의를 위한 초기 환경 (ρ_{init})

$$
\begin{array}{lll}
\underline{\mathit{if}}\ \kappa_T\ \kappa_F \quad (o,\ell:\phi,\gamma,\rho,\delta) & = & \nu \,\mathsf{E}\, \mathbf{Z} \longrightarrow \\
 & & (\nu \mid \mathbf{Z} = 1 \longrightarrow \kappa_T,\ \kappa_F)\ (o,\phi,\gamma,\rho,\delta),\ \bot \\
 & & \text{where}\ \ \nu = \gamma\ (\mathit{Elide}\ \gamma\ \ell) \\
 & & \\
\underline{\mathit{add}} \quad (o,\ell_0:\ell_1:\phi,\gamma,\rho,\delta) & = & (\nu_0 \,\mathsf{E}\, \mathbf{Z} \wedge \nu_1 \,\mathsf{E}\, \mathbf{Z}) \longrightarrow \\
 & & \underline{\mathit{pushval}}\ \nu\ (o,\phi,\gamma,\rho,\delta),\ \bot \\
 & & \text{where}\ \ \nu\ = (\nu_0 \mid \mathbf{Z} + \nu_1 \mid \mathbf{Z})\ \mathbf{in}\ \mathbf{N} \\
 & & \qquad\ \ \nu_i\ = \gamma\ (\mathit{Elide}\ \gamma\ \ell_i) \\
 & & \\
\underline{\mathit{eq}} \quad (o,\ell_0:\ell_1:\phi,\gamma,\rho,\delta) & = & (\nu_0 \,\mathsf{E}\, \mathbf{Z} \wedge \nu_1 \,\mathsf{E}\, \mathbf{Z}) \longrightarrow \\
 & & \underline{\mathit{pushval}}\ \nu\ (o,\phi,\gamma,\rho,\delta),\ \bot \\
 & & \text{where}\ \ \nu\ = (\nu_0 \mid \mathbf{Z}\ =\ \nu_1 \mid \mathbf{Z}) \longrightarrow \\
 & & \qquad\qquad 1\ \mathbf{in}\ \mathbf{N},\ 0\ \mathbf{in}\ \mathbf{N} \\
 & & \qquad\ \ \nu_i\ =\ \gamma\ (\mathit{Elide}\ \gamma\ \ell_i)
\end{array}
$$

그림 7: G-machine의 스택 의미정의에서 사용되는 내장 함수

수 노드를 스택에 넣는 함수이다. $\underline{\mathit{eq}}$는 스택 상의 두 노드의 정수 값을 비교한 후 결과의 정수 노드를 스택에 넣는 함수이다. 이들 내장 함수에서 논리 값 참은 정수 노드 1로, 거짓은 0으로 나타낸다. 만약 기대하고 있는 노드 형태와 다른 형태의 노드가 발견된다면 $\bot$ 상태를 반환한다.

이상에서 G-machine의 스택 의미정의를 기술하였다. G-machine의 스택 의미정의는 G-machine 상태 변환에 의해 기술되는데, G-machine 상태 변환은 G-machine 명령어에 해당하는 기본 함수를 이용하여 기술된다. 몇몇 완전 조합자는 초기 환경에 라이브러리 함수로 미리 등록되어 있다. 이러한 일련의 의미정의를 만들어 나가는 과정은 G-machine 추상 기계를 정의하는 과정과 유사한데, 이 모든 정의가 의미 도메인 안에서 이루어진다는 점에서 추상 기계 정의와 구분된다.

4. ZG-machine의 스택 의미정의

이 절에서는 ZG-machine의 스택 의미정의를 기술한다. ZG-machine에서는 태그 옮김으로 인해서 그래프 노드 구조가 워드 단위로 해체된다. 따라서, 그래프를 나타내는 의미 도메인이 G-machine과 다소 차이를 보이는데, ZG-machine 스택 의미정의를 위한 의미 도메인은 그림 8에 나타나 있다.

ZG-machine의 상태에 대한 의미 도메인 $\mathbf{S}$는 $\mathbf{O}$, $\mathbf{L}^*$, $\mathbf{G}$, $\mathbf{U}$, $\mathbf{D}$ 등 다섯 도메인의 곱 도메인으로 정의된다. 각 부분 도메인은 G-machine에서와 마찬가지로 각각 출력값, 스택, 그래프 함수, 환경 함수, 덤프 스택을 나타낸다. G-machine에서와 다른 점은 레이블 도메인 $\mathbf{L}$이 G-machine의 의미정의에서는 구체적으로 정해지지 않은데 비해 ZG-machine에서는 0을 포함한 자연수 도메인 $\mathbf{Z}_+$으로 정의되었다. 이렇게 레이블 도메인을 자연수 도메인으로 정한 이유는, 상대 주소를 통한 주소 연산, 즉 레이블에 대한 연산이 ZG-machine에서는 필요하기 때문이다.

그래프 함수 도메인 $\mathbf{G}$도 조금 다르게 정의되는데, G-machine에서는 노드 도메인 $\mathbf{N}$으로의 함수로 정의하였지만 ZG-machine에서는 워드 도메인 $\mathbf{W}$로의 함수로 정의한다. 그래프 워드 도메인 $\mathbf{W}$는 $\mathbf{P}$, $\mathbf{I}$, $\mathtt{Id}$, $\mathbf{Z}$ 등 네 도메인의 합 도메인으로 정의되는데, 우회 워드를 나타내는 $\mathbf{I}$와 이름을 나타내는 $\mathtt{Id}$, 정수를 나타내는 $\mathbf{Z}$는 G-machine의 스택 의미정의에서와 같은 역할을 하지만, $\mathbf{P}$는 ZG-machine에서 새로 정의된 도메인이다. 도메인 $\mathbf{P}$는 상대 주소와 함께 옮겨진 태그를 나타내기 위한 도메인이다.

도메인 $\mathbf{P}$는 태그를 나타내는 도메인 $\mathbf{T}$와 상

σ	$\in$	$\mathbf{S}$	$=$	$\mathbf{O} \times \mathbf{L}^* \times \mathbf{G} \times \mathbf{U} \times \mathbf{D}$	(상태)
o	$\in$	$\mathbf{O}$	$=$	$\mathbf{Z}^*$	(출력값)
γ	$\in$	$\mathbf{G}$	$=$	$[\mathbf{L} \rightarrow \mathbf{W}]$	(그래프 함수)
ω	$\in$	$\mathbf{W}$	$=$	$\mathbf{P} + \mathbf{I} + \mathtt{Id} + \mathbf{Z}$	(그래프 워드)
		$\mathbf{P}$	$=$	$\mathbf{T} \times \mathbf{Z}_+$	(태그, 상대 주소 워드)
		$\mathbf{I}$	$=$	$\mathbf{L}$	(우회 워드)
		$\mathbf{T}$	$=$	$\{\underline{\mathtt{ap}}, \underline{\mathtt{int}}\}_\perp$	(태그)
		$\mathbf{Z}$	$=$	$\{\ldots, -2, -1, 0, 1, 2, \ldots\}_\perp$	(정수)
ρ	$\in$	$\mathbf{U}$	$=$	$[\mathtt{Id} \rightarrow \mathbf{K}]$	(환경)
δ	$\in$	$\mathbf{D}$	$=$	$\mathbf{L}^{**}$	(덤프)
ϕ	$\in$	$\mathbf{L}^*$			(스택)
κ	$\in$	$\mathbf{K}$	$=$	$[\mathbf{S} \rightarrow \mathbf{S}]$	(상태 변환 함수)
β	$\in$	$\mathbf{B}$	$=$	$[\mathtt{Id} \rightarrow \mathbf{Z}_+]$	(바인딩)
		$\mathbf{Z}_+$	$=$	$\{0, 1, 2, \ldots\}_\perp$	(0을 포함한 자연수)
ℓ	$\in$	$\mathbf{L}$	$=$	$\mathbf{Z}_+$	(노드 주소)
χ	$\in$	$\mathbf{C}$	$=$	$\mathbf{P} + \mathbf{Z}_+ + \mathtt{Id} + \mathbf{Z}$	(그래프 워드 초기화 정보)
ζ	$\in$	$\mathbf{C}^*$			(초기화 정보 리스트)

그림 8: ZG-machine 스택 의미정의를 위한 의미 도메인

대 주소를 나타내는 도메인 $\mathbf{Z}_+$의 곱 도메인으로 정의된다. 상대 주소는 절대 주소와 함께 계산되어야 하므로 절대 주소를 나타내는 레이블 도메인과 같은 도메인 $\mathbf{Z}_+$으로 정의한다. 태그 도메인 $\mathbf{T}$는 그래프 노드의 태그를 포함하면 되는데, 여기서는 생성자 노드를 다루지 않고, 적용 노드와 정수 노드만을 구분하면 되므로 $\{\underline{\mathtt{ap}}, \underline{\mathtt{int}}\}_\perp$로 정의하였다. 물론 언어를 확장할 경우 필요에 따라 태그를 추가할 수 있다.

ZG-machine에서는 순차적인 주소를 갖는 공간을 할당하여 이를 초기화함으로써 그래프를 생성하는데, 이 초기화에 필요한 정보를 위해 도메인 $\mathbf{C}$를 정의한다. 초기화 정보 도메인 $\mathbf{C}$는 $\mathbf{P}$, $\mathbf{Z}_+$, $\mathtt{Id}$, $\mathbf{Z}$ 등 네 도메인의 합 도메인으로 정의된다. 도메인 $\mathbf{C}$의 원소는 $\mathbf{Z}_+$에 속하는 경우만 제외하고 그대로 그래프 워드에 복사된다. 도메인 $\mathbf{C}$의 원소가 $\mathbf{Z}_+$에 속하는 경우에는, 스택의 특정 위치의 주소가 그래프 워드에 복사되어 우회 워드를 나타내도록 한다. 초기화 정보는 통상 리스트로 묶여 사용되는데, 초기화 정보 리스트는 $\mathbf{C}^*$ 도메인의 원소가 된다.

그 외의 의미 도메인은 G-machine에서와 같다. 그림 8에서도 G-machine의 의미정의에서와 마찬가지로 각 의미 도메인에 속하는 대표 원소를 그리이스 문자의 소문자로 나타내었는데, 이들은 스택 의미정의를 기술하는데 사용된다. 정의된 의미 도메인을 기초로 ZG-machine의 스택 의미정의를 기술하면 그림 9에서와 같다.

ZG-machine은 그래프 생성 방법을 제외하고 G-machine과 유사한 형태를 이루고 있으므로, ZG-machine의 스택 의미정의도 예상대로 G-machine의 스택 의미정의와 거의 같게 정의된다. 다만 그래프 생성을 위한 의미 함수 $\mathcal{C}$가 차이를 보인다. 의미 함수 $\mathcal{C}$는 표현식의 종류에 상관 없이 의미 함수 $\mathcal{N}$을 호출하여 초기화 정보를 얻어낸 후, 이를 기본 함수 $\underline{alloc}$에 넘겨 줌으로써 그래프를 생성한다.

의미 함수 $\mathcal{N}$은 인수로 주어진 표현식 구문의 종류에 따라 다르게 정의된다. 의미 함수 $\mathcal{N}$의 인수가 이름 I이고 I가 $dom(\beta)$에 속해 있는 경우, 이 이름 I는 형식 인수 이름을 나타내므로 스택 상에서 해당 형식 인수가 놓여지게 될 위치를 초기화 정보로 반환한다. 이름 I가 $dom(\beta)$에 속해 있지 않은 경우, 이 이름 I는 다른 완전조합자 이름이므로 그 이름 자체를 초기화 정보로 반환한다.

$$
\begin{array}{lcl}
\mathcal{P} : \texttt{Prog} \to \mathbf{S} & & \\
\mathcal{P}\ [\![P]\!] & = & (\underline{\mathit{print}} \circ \underline{\mathit{eval}} \circ (\rho\ [\![\texttt{main}]\!]))\ ([],\ [],\ \{\},\ \rho,\ []) \\
 & & \text{where } \rho = \rho_{init} \oplus \mathcal{D}\ [\![P]\!] \\
 & & \\
\mathcal{D} : \texttt{Defs} \to \mathbf{U} & & \\
\mathcal{D}\ [\![D_0\ ;\ D_1]\!] & = & \mathcal{D}\ [\![D_0]\!] \oplus \mathcal{D}\ [\![D_1]\!] \\
\mathcal{D}\ [\![I = C]\!] & = & \{I \mapsto \mathcal{F}\ [\![C]\!]\ \{\}\ 0\} \\
 & & \\
\mathcal{F} : \texttt{Comb} \to \mathbf{B} \to \mathbf{Z}_+ \to \mathbf{K} & & \\
\mathcal{F}\ [\![\lambda I\ .\ C]\!]\ \beta\ n & = & \mathcal{F}\ [\![C]\!]\ (\beta \oplus \{I \mapsto n\})\ (n+1) \\
\mathcal{F}\ [\![E]\!]\ \beta\ n & = & \underline{\mathit{entry}}\ n\ (\mathcal{R}\ [\![E]\!]\ \beta\ n) \\
 & & \\
\mathcal{R} : \texttt{Expr} \to \mathbf{B} \to \mathbf{Z}_+ \to \mathbf{K} & & \\
\mathcal{R}\ [\![E]\!]\ \beta\ n & = & \underline{\mathit{exit}}\ n \circ \mathcal{C}\ [\![E]\!]\ \beta \\
 & & \\
\mathcal{E} : \texttt{Expr} \to \mathbf{B} \to \mathbf{K} & & \\
\mathcal{E}\ [\![E]\!]\ \beta & = & \underline{\mathit{eval}} \circ \mathcal{C}\ [\![E]\!]\ \beta \\
 & & \\
\mathcal{C} : \texttt{Expr} \to \mathbf{B} \to \mathbf{K} & & \\
\mathcal{C}\ [\![E]\!]\ \beta & = & \underline{\mathit{alloc}}\ (\mathcal{N}\ [\![E]\!]\ \beta\ 1) \\
 & & \\
\mathcal{N} : \texttt{Expr} \to \mathbf{B} \to \mathbf{Z}_+ \to \mathbf{C}^* & & \\
\mathcal{N}\ [\![I]\!]\ \beta\ r & = & I \in \mathit{dom}(\beta) \longrightarrow [(\beta\ [\![I]\!])\ \mathbf{in}\ \mathbf{C}],\ [[\![I]\!]\ \mathbf{in}\ \mathbf{C}] \\
\mathcal{N}\ [\![Z]\!]\ \beta\ r & = & [(\underline{\texttt{int}},\ r)\ \mathbf{in}\ \mathbf{C},\ (\mathcal{Z}\ [\![Z]\!]\ \mathbf{in}\ \mathbf{C})] \\
\mathcal{N}\ [\![E_0\ E_1]\!]\ \beta\ r & = & (\underline{\texttt{ap}},\ r)\ \mathbf{in}\ \mathbf{C} : h_0 : h_1 : (t_0 +\!\!+ t_1) \\
 & & \text{where } h_0 : t_0 \quad = \quad \mathcal{N}\ [\![E_0]\!]\ \beta\ 2 \\
 & & \qquad\quad h_1 : t_1 \quad = \quad \mathcal{N}\ [\![E_1]\!]\ \beta\ (1 + \#t_0) \\
 & & \\
\mathcal{Z} : \texttt{Int} \to \mathbf{Z} & & \text{(자세한 정의 생략)}
\end{array}
$$

그림 9: ZG-machine의 스택 의미정의

의미 함수 $\mathcal{N}$의 인수가 정수 Z인 경우, 정수를 나타내는 태그 int와 $\mathcal{N}$의 두번째 인수로 주어진 상대 주소 r을 순서쌍으로 만들어 초기화 정보에 기록하고, 해당 정수값 $\mathcal{Z}$ $[\![Z]\!]$를 추가로 기록하여 반환한다. 이렇게 기록된 초기화 정보 중에서 태그와 상대 주소 순서쌍에 대한 초기화 정보 '(int, r) **in C**'는 그 다음 초기화 정보 '$\mathcal{Z}$ $[\![Z]\!]$ **in C**'와 상대 주소 r 만큼 떨어져 놓이도록 나중에 $\mathcal{N}$에 의하여 위치가 재조정된다.

의미 함수 $\mathcal{N}$의 인수가 적용 표현식 $(E_0\ E_1)$인 경우, 각 표현식 E_i에 대하여 먼저 $\mathcal{N}$을 재귀적으로 적용하여 초기화 목록을 추출해 낸다. 이 때, 최종적으로 초기화 목록의 앞쪽에 놓여질 표현식 E_0에 대해서는 2를 상대 주소로 설정하도록 $\mathcal{N}$을 호출하고, 뒷쪽에 놓여질 표현식 E_1에 대해서는 E_0의 초기화 목록 길이를 상대 주소로 설정하도록 $\mathcal{N}$을 호출한다. 적용 노드임을 알리는 태그 ap와 인수로 주어진 상대 주소 순서쌍을 만들어 초기화 정보 목록의 맨 앞에 기록하고, 앞에서 생성된 E_i에 대한 초기화 목록을 조정, 연결하여 최종 초기화 목록을 만든다.

이상에서 ZG-machine 스택 의미정의의 전체 구조를 기술하였는데, 설명되지 않은 나머지 의미 함수($\mathcal{P}$와 $\mathcal{D}$, $\mathcal{F}$, $\mathcal{R}$, $\mathcal{E}$, $\mathcal{Z}$)에 대해서는 G-machine의 스택 의미정의에서와 같다. ZG-machine의 스택 의미정의에서도 몇 가지 기본 함수를 사용하였는데, 이들 기본 함수에 대한 정의가 그림 10에 정리되어 있다.

그림 10의 기본 함수들은 대부분 그림 5에 정리한 G-machine에서의 기본 함수와 같다. ZG-machine에서 그래프를 생성하기 위해 *alloc*과 *copy*가 추가되었고, 대신 G-machine에서 사용되던 *pushval*과 *mkap*는 제거되었다. G-machine에서의 기본 함수 *push*는 내장 함수로 바뀌어 그림 11에 정의되어 있다. 그 외 차이점으로, G-machine과 다른 방법으로 생성된 그래프를 올바로 참조하기 위해 함수 *unwind*와 *print*가 변경되었다.

기본 함수 정의에 사용된 보조 함수를 간단히 설명하면 다음과 같다. 보조 함수 *NewBlock*은 그래프 함수 γ로부터 n개의 새로운 일련의 레이블을 찾아 그 시작 레이블을 반환한다. 보조 함수 *Arg*는 레이블 ℓ이 가리키는 워드 ω가 ap 태그를 가진 순서쌍 워드인 경우, ω의 상대 주소 '*snd* $(\omega \mid \mathbf{P})$'를 이용하여 적용 노드의 인수에 해당하는 주소를 반환하는 함수이다. 보조 함수 *Elide*는 주소 ℓ이 가리키는 노드가 우회 노드가 아닐 때까지 우회 노드를 따라가는 함수이다.

ZG-machine의 스택 의미정의에서도 초기 환경 ρ_{init}을 사용하였는데, ρ_{init}의 정의는 G-machine의 경우(그림 6)와 똑같다. 다만 내장 함수는, 다른 방법으로 생성된 그래프를 해석해야 하므로 G-machine과 다르게 되는데, 내장 함수 정의는 그림 11에 나타나 있다.

그림 11의 내장 함수 정의에서 G-machine과 다른 점은 스택의 레이블이 원하는 노드 형태인지 알기 위해서 복잡한 검사가 필요하다는 점이다. 레이블이 가리키는 워드의 종류를 알기 위해서는 먼저 도메인 검사 연산으로 해당 워드가 우회 워드, 이름 워드, 순서쌍 워드 중 어느 것인가 판단해야 하고, 순서쌍 워드의 경우에는 순서쌍 워드의 태그를 다시 참조해야 한다.

이상에서 ZG-machine의 스택 의미정의를 기술하였다. ZG-machine의 의미정의가 G-machine의 의미정의와 다른 점은 옮겨진 태그를 표현하기 위해 의미 도메인이 워드 단위로 바뀐 점과 워드 단위에서 그래프를 생성하기 위해 의미 함수 $\mathcal{C}$가 변경되고 $\mathcal{N}$이 추가된 점, 또 태그가 옮겨진 형태의 그래프를 참조하기 위해 기본 함수와 내장 함수가 바뀐 점 등을 들 수 있다.

5. 그래프 합동

ZG-machine은 그래프 생성 방법을 제외하고 G-machine과 유사한 상태 변환을 하므로, 이 상태 변환에 초점을 맞추면 두 기계가 같다는 것을 보일 수 있다. 그러나, G-machine 의미정의에서의 상태 도메인과 ZG-machine 의미정의에서의 상태 도메인은 다르게 정의되어 있으므로 비교하기 어렵다. 이 논문에서는 두 기계의 상태를 비교하기 위해서 각 기계의 상태로부터 특정 표준 그래프 함수로의 변환을 정의한다. 이 논문에

$$
\begin{array}{lrcl}
\underline{alloc}\ \zeta & (o,\ \phi,\ \gamma,\ \rho,\ \delta) & = & \underline{copy}\ 0\ \zeta\ (o,\ \ell : \phi,\ \gamma,\ \rho,\ \delta) \\
 & & & \text{where}\ \ell\ =\ \mathit{NewBlock}\ \gamma\ (\#\zeta) \\
 & & & \\
\underline{copy}\ i\ \zeta & (o,\ \ell : \phi,\ \gamma,\ \rho,\ \delta) & = & \#\zeta = 0 \longrightarrow (o,\ \ell : \phi,\ \gamma,\ \rho,\ \delta), \\
 & & & \quad \underline{copy}\ (i+1)\ (\mathit{tl}\ \zeta)\ (o,\ \ell : \phi,\ \gamma',\ \rho,\ \delta) \\
 & & & \text{where} \\
 & & & \quad \gamma' = \gamma \oplus \{\ell + i \mapsto \omega\} \\
 & & & \quad \omega = \chi \in \mathbf{Z}_+ \longrightarrow (\phi\ !\ (\chi \mid \mathbf{Z}_+))\ \mathbf{in}\ \mathbf{W}, \\
 & & & \qquad\quad \chi\,\mathsf{E}\,\mathtt{Id} \longrightarrow (\chi \mid \mathtt{Id})\ \mathbf{in}\ \mathbf{W}, \\
 & & & \qquad\quad \chi\,\mathsf{E}\,\mathbf{P} \longrightarrow (\chi \mid \mathbf{P})\ \mathbf{in}\ \mathbf{W}, \\
 & & & \qquad\quad \chi\,\mathsf{E}\,\mathbf{Z} \longrightarrow (\chi \mid \mathbf{Z})\ \mathbf{in}\ \mathbf{W},\ \bot \\
 & & & \quad \chi = \mathit{hd}\ \zeta \\
 & & & \\
\underline{exit}\ n & & = & \underline{unwind} \circ \underline{pop}\ n \circ \underline{update}\ n \\
 & & & \\
\underline{pop}\ n & (o,\ \phi,\ \gamma,\ \rho,\ \delta) & = & (o,\ \mathit{drop}\ n\ \phi,\ \gamma,\ \rho,\ \delta) \\
 & & & \\
\underline{update}\ n & (o,\ \ell : \phi,\ \gamma,\ \rho,\ \delta) & = & (o,\ \phi,\ \gamma \oplus \{(\phi\ !\ n) \mapsto \gamma\ \ell\},\ \rho,\ \delta) \\
 & & & \\
\underline{eval} & (o,\ \ell : \phi,\ \gamma,\ \rho,\ \delta) & = & (\underline{restore} \circ \underline{unwind})\ (o,\ [\ell],\ \gamma,\ \rho,\ \phi : \delta) \\
 & & & \\
\underline{restore} & (o,\ \phi,\ \gamma,\ \rho,\ \phi' : \delta) & = & (o,\ (\mathit{Elide}\ \gamma\ (\mathit{last}\ \phi)) : \phi',\ \gamma,\ \rho,\ \delta) \\
 & & & \\
\underline{entry}\ n\ \kappa & (o,\ \ell : \phi,\ \gamma,\ \rho,\ \delta) & = & \#\phi \geq n \longrightarrow \kappa\ (o,\ \phi_a +\!\!+ \phi_r,\ \gamma,\ \rho,\ \delta), \\
 & & & \qquad\qquad (o,\ \ell : \phi,\ \gamma,\ \rho,\ \delta) \\
 & & & \text{where}\ \phi_a\ =\ \mathit{map}\ (\mathit{Arg}\ \gamma)\ (\mathit{take}\ n\ \phi) \\
 & & & \qquad \phi_r\ =\ n < 1 \longrightarrow \phi, \mathit{drop}\ (n-1)\ \phi \\
 & & & \\
\underline{unwind} & (o,\ \ell : \phi,\ \gamma,\ \rho,\ \delta) & = & (\omega\,\mathsf{E}\,\mathtt{Id}) \qquad\qquad \longrightarrow \\
 & & & \qquad \rho\ (\omega \mid \mathtt{Id})\ (o,\ \ell : \phi,\ \gamma,\ \rho,\ \delta), \\
 & & & (\omega\,\mathsf{E}\,\mathbf{I}) \qquad\qquad \longrightarrow \\
 & & & \qquad \underline{unwind}\ (o,\ (\omega \mid \mathbf{I}) : \phi,\ \gamma,\ \rho,\ \delta), \\
 & & & (\omega\,\mathsf{E}\,\mathbf{P} \wedge t = \underline{\mathtt{ap}}) \longrightarrow \\
 & & & \qquad \underline{unwind}\ (o,\ (\ell + r) : \ell : \phi,\ \gamma,\ \rho,\ \delta), \\
 & & & (\omega\,\mathsf{E}\,\mathbf{P} \wedge t = \underline{\mathtt{int}}) \longrightarrow \\
 & & & \qquad (o,\ \ell : \phi,\ \gamma,\ \rho,\ \delta),\ \bot \\
 & & & \text{where}\ \omega\ =\ \gamma\ \ell \\
 & & & \qquad t\ =\ \mathit{fst}\ (\omega \mid \mathbf{P}) \\
 & & & \qquad r\ =\ \mathit{snd}\ (\omega \mid \mathbf{P}) \\
 & & & \\
\underline{print} & (o,\ \ell : \phi,\ \gamma,\ \rho,\ \delta) & = & (\omega\,\mathsf{E}\,\mathbf{P} \wedge t = \underline{\mathtt{int}}) \longrightarrow \\
 & & & \qquad (o +\!\!+ (\omega' \mid \mathbf{Z}),\ \phi,\ \gamma,\ \rho,\ \delta),\ \bot \\
 & & & \text{where}\ \omega\ =\ \gamma\ \ell \\
 & & & \qquad t\ =\ \mathit{fst}\ (\omega \mid \mathbf{P}) \\
 & & & \qquad \omega'\ =\ \gamma\ (\ell + \mathit{snd}\ (\omega \mid \mathbf{P}))
\end{array}
$$

그림 10: ZG-machine의 스택 의미정의에서 사용되는 기본 함수

$$
\begin{array}{lrcl}
\underline{\mathit{if}}\ \kappa_T\ \kappa_F & (o,\ \ell : \phi,\ \gamma,\ \rho,\ \delta) & = & (\omega \in \mathbf{P}\ \wedge\ t = \underline{\texttt{int}}\ \wedge\ \omega' \in \mathbf{Z}) \\
 & & & \longrightarrow\ (\omega' \mid \mathbf{Z} = 1\ \longrightarrow\ \kappa_T,\ \kappa_F) \\
 & & & \quad (o,\ \phi,\ \gamma,\ \rho,\ \delta),\ \bot \\
 & & & \text{where } \omega\ =\ \gamma\ \ell \\
 & & & \quad t\ =\ \mathit{fst}\ (\omega \mid \mathbf{P}) \\
 & & & \quad \omega'\ =\ \gamma\ (\ell + \mathit{snd}\ (\omega \mid \mathbf{P})) \\
 & & & \\
\underline{\mathit{add}} & (o,\ \ell_0 : \ell_1 : \phi,\ \gamma,\ \rho,\ \delta) & = & \bigwedge_{i=0}^{1}(\omega_i \in \mathbf{P}\ \wedge\ t_i = \underline{\texttt{int}}\ \wedge\ \omega_i' \in \mathbf{Z}) \\
 & & & \longrightarrow\ \underline{\mathit{alloc}}\ [(\underline{\texttt{int}},\ 1)\ \mathbf{in}\ \mathbf{C},\ n\ \mathbf{in}\ \mathbf{C}] \\
 & & & \quad (o,\ \phi,\ \gamma,\ \rho,\ \delta),\ \bot \\
 & & & \text{where}\ \ \omega_i\ =\ \gamma\ \ell_i \\
 & & & \quad t_i\ =\ \mathit{fst}\ (\omega_i \mid \mathbf{P}) \\
 & & & \quad \omega_i'\ =\ \gamma\ (\ell_i + \mathit{snd}\ (\omega_i \mid \mathbf{P})) \\
 & & & \quad\quad \text{for}\ \ i \in \{0,\ 1\} \\
 & & & \quad n\ =\ \omega_0' \mid \mathbf{Z} + \omega_1' \mid \mathbf{Z} \\
 & & & \\
\underline{\mathit{eq}} & (o,\ \ell_0 : \ell_1 : \phi,\ \gamma,\ \rho,\ \delta) & = & \bigwedge_{i=0}^{1}(\omega_i \in \mathbf{P}\ \wedge\ t_i = \underline{\texttt{int}}\ \wedge\ \omega_i' \in \mathbf{Z}) \\
 & & & \longrightarrow\ \underline{\mathit{alloc}}\ [(\underline{\texttt{int}},\ 1)\ \mathbf{in}\ \mathbf{C},\ n\ \mathbf{in}\ \mathbf{C}] \\
 & & & \quad (o,\ \phi,\ \gamma,\ \rho,\ \delta),\ \bot \\
 & & & \text{where}\ \ \omega_i\ =\ \gamma\ \ell_i \\
 & & & \quad t_i\ =\ \mathit{fst}\ (\omega_i \mid \mathbf{P}) \\
 & & & \quad \omega_i'\ =\ \gamma\ (\ell_i + \mathit{snd}\ (\omega_i \mid \mathbf{P})) \\
 & & & \quad\quad \text{for}\ \ i \in \{0,\ 1\} \\
 & & & \quad n\ =\ (\omega_0' \mid \mathbf{Z} = \omega_1' \mid \mathbf{Z})\ \longrightarrow\ 1,\ 0 \\
 & & & \\
\underline{\mathit{push}}\ n & (o,\ \phi,\ \gamma,\ \rho,\ \delta) & = & (o,\ (\phi\ !\ n) : \phi,\ \gamma,\ \rho,\ \delta)
\end{array}
$$

그림 11: ZG-machine의 스택 의미정의에서 사용되는 내장 함수

서 사용할 표준 그래프 도메인은 그림 12에 정의되어 있다.

$$\begin{array}{lcll} \overline{\mathbf{G}} & = & \mathbf{Z} + \mathtt{Id} + \mathbf{A} & \text{(표준 그래프 노드)} \\ \mathbf{A} & = & \overline{\mathbf{G}} \times \overline{\mathbf{G}} & \text{(적용 노드)} \end{array}$$

그림 12: 표준 그래프 도메인

그림 12의 표준 그래프 도메인 $\overline{\mathbf{G}}$는 정수 도메인 $\mathbf{Z}$와 이름 도메인 Id, 적용 노드 도메인 $\mathbf{A}$의 합 도메인이다. 도메인 $\mathbf{Z}$와 Id는 앞의 스택 의미정의에서와 같지만 도메인 $\mathbf{A}$는 재귀적으로 표준 그래프 도메인 $\overline{\mathbf{G}}$ 두 개의 곱으로 정의된다.

이렇게 표준 그래프 도메인이 결정되면 각 추상 기계의 상태로부터 표준 그래프 함수로의 변환 함수를 정의할 수 있다. 표준 그래프 함수란 스택 최상위로부터의 위치를 자연수로 주었을 때, 그 위치에 해당하는 표준 그래프를 반환하는 함수를 말한다. 표준 그래프 함수는 주어진 상태에서 레이블을 통해 접근 가능한 모든 그래프를 표준 그래프로 표현한 것으로, 그 상태의 모든 그래프 정보를 추출해 낸 것으로 볼 수 있다. G-machine, ZG-machine 각각의 상태로부터 표준 그래프 함수로의 변환 함수를 정의하면 그림 13, 14에서와 같다.

표준 그래프 함수로의 변환 함수는 모두 세 개의 함수로 구성되는데, 이 중에서 함수 $\mathcal{G}$는 실제 변환을 위해 호출되는 함수이고, 함수 $\mathcal{G}_1$와 $\mathcal{G}_2$는 보조 함수 역할을 한다. 함수 $\mathcal{G}$는 각 추상기계 상태의 스택과 덤프 스택의 레이블을 모두 모아서 그래프 함수 γ와 처음 생성될 표준 그래프의 위치 0과 함께 $\mathcal{G}_1$에 넘겨준다. 여기서 표준 그래프의 위치란 그 상태의 스택과 덤프 스택을 망라한 레이블 목록($\underline{concat}\ (\phi : \delta)$)에서 그 그래프가 어느 위치에 있느냐 하는 것을 나타낸다. 즉 위치가 0이면 스택 최상위의 그래프를 나타내고, 최상위에서 두 번째 레이블이 가리키는 그래프는 1, 이런 식으로 번호가 매겨진다. 함수 $\mathcal{G}_1$는 각 레이블마다 $\mathcal{G}_2$를 호출하여 표준 그래프를 생성하고, 생성된 그래프를 그래프의 위치 정보와 연관시켜 표준 그래프 함수를 생성한다. 함수 $\mathcal{G}_2$는 하나의 레이블에 대하여 주어진 그래프 함수 γ를 참조하여 표준 그래프를 생성한다.

앞에서 정의한 표준 그래프 변환 함수를 통해 두 추상 기계의 상태에 대하여 그래프 합동 관계를 정의할 수 있다. 그래프 합동 관계는 각 기계의 상태가 나타내는 실질적인 그래프의 내용이 같다는 것을, 즉 두 추상기계 상태의 그래프가 동등함을 나타내는 관계이다. 그래프 합동 관계의 정의는 다음과 같다.

정의 1 *각 추상 기계의 상태 $\sigma_{\mathfrak{g}} \in \mathbf{S}_{\mathfrak{g}}$, $\sigma_{\mathfrak{z}} \in \mathbf{S}_{\mathfrak{z}}$에 대하여 $\mathcal{G}_{\mathfrak{g}}\ \sigma_{\mathfrak{g}} = \mathcal{G}_{\mathfrak{z}}\ \sigma_{\mathfrak{z}}$ 일 때, 두 상태는 그래프 합동 관계에 있다고 하고, $\sigma_{\mathfrak{g}} \cong \sigma_{\mathfrak{z}}$으로 나타낸다. 같은 추상 기계의 두 상태 $\sigma_1 \in \mathbf{S}$, $\sigma_2 \in \mathbf{S}$에 대해서도 $\mathcal{G}\ \sigma_1 = \mathcal{G}\ \sigma_2$ 이면, 두 상태는 그래프 합동 관계에 있다고 하고, $\sigma_1 \cong \sigma_2$로 나타낸다.*

두 추상 기계의 상태에 대한 그래프 합동 관계는 자연스럽게 상태 변환 함수로 확장될 수 있다. 그래프 합동 관계의 상태 변환 함수는 그래프 합동 관계를 보존하도록 상태 변환을 수행하는 함수를 말한다. 상태 변환 함수의 그래프 합동 관계를 정의하면 다음과 같다.

정의 2 *각 추상 기계에서 정의된 상태 변환 함수 $f_{\mathfrak{g}} \in \mathbf{S}_{\mathfrak{g}} \to \mathbf{S}_{\mathfrak{g}}$, $f_{\mathfrak{z}} \in \mathbf{S}_{\mathfrak{z}} \to \mathbf{S}_{\mathfrak{z}}$에 대하여*

$$\forall\ \sigma_{\mathfrak{g}} \in \mathbf{S}_{\mathfrak{g}},\ \sigma_{\mathfrak{z}} \in \mathbf{S}_{\mathfrak{z}}\ .\ \sigma_{\mathfrak{g}} \cong \sigma_{\mathfrak{z}}\ \Rightarrow$$
$$(f_{\mathfrak{g}}\ \sigma_{\mathfrak{g}} \cong f_{\mathfrak{z}}\ \sigma_{\mathfrak{z}}\ \vee\ f_{\mathfrak{g}}\ \sigma_{\mathfrak{g}}, f_{\mathfrak{z}}\ \sigma_{\mathfrak{z}}\text{의 상태}$$
변환이 모두 끝나지 않는다)

일 때, 두 함수는 그래프 합동 관계에 있다고 하고, $f_{\mathfrak{g}} \cong f_{\mathfrak{z}}$으로 나타낸다. 같은 추상 기계의 두 상태 변환 함수 $f_1 \in \mathbf{S} \to \mathbf{S}$, $f_2 \in \mathbf{S} \to \mathbf{S}$에 대해서도 $\forall\ \sigma_1 \in \mathbf{S},\ \sigma_2 \in \mathbf{S}.\ \sigma_1 \cong \sigma_2\ \Rightarrow\ f_1\ \sigma_1 \cong f_2\ \sigma_2$ 이면, 두 함수는 그래프 합동 관계에 있다고 하고, $f_1 \cong f_2$로 나타낸다.

이렇게 두 기계의 상태와 상태 변환 함수에 대하여 그래프 합동 관계를 정의하면, 각 추상 기계의 초기 상태가 그래프 합동이라는 것과, 각 추상 기계의 스택 의미정의에서 제시하는 상태 변환 함수들 간에 그래프 합동 관계의 대응 관계가

$$
\begin{array}{l}
\mathcal{G} \;:\; \mathbf{S}_{\mathfrak{g}} \to\; (\mathbf{Z}_+ \;\to\; \overline{\mathbf{G}}) \\
\mathcal{G}\ (o,\ \phi,\ \gamma,\ \rho,\ \delta) \;=\; \mathcal{G}_1\ (\mathit{concat}\ (\phi : \delta))\ \gamma\ 0
\end{array}
$$

$$
\begin{array}{l}
\mathcal{G}_1 \;:\; \mathbf{L}^* \;\to\; \mathbf{G} \;\to\; \mathbf{Z}_+ \;\to\; (\mathbf{Z}_+ \;\to\; \overline{\mathbf{G}}) \\
\mathcal{G}_1\ \phi\ \gamma\ p \;=\; (\#\phi = 0) \longrightarrow \{\},\ \mathcal{G}_1\ (\mathit{tl}\ \phi)\ \gamma\ (p+1)\ \oplus\ \{p \;\mapsto\; \mathcal{G}_2\ (\mathit{hd}\ \phi)\ \gamma\}
\end{array}
$$

$$
\begin{array}{l}
\mathcal{G}_2 \;:\; \mathbf{L} \;\to\; \mathbf{G} \;\to\; \overline{\mathbf{G}} \\
\mathcal{G}_2\ \ell\ \gamma \;=\; \nu\,\mathtt{E}\,\mathbf{Z} \quad\longrightarrow\quad (\nu \mid \mathbf{Z})\ \mathbf{in}\ \overline{\mathbf{G}}, \\
\qquad\qquad \nu\,\mathtt{E}\,\mathtt{Id} \quad\longrightarrow\quad (\nu \mid \mathtt{Id})\ \mathbf{in}\ \overline{\mathbf{G}}, \\
\qquad\qquad \nu\,\mathtt{E}\,\mathbf{A} \quad\longrightarrow\quad (\mathcal{G}_2\ \ell_0\ \gamma,\ \mathcal{G}_2\ \ell_1\ \gamma)\ \mathbf{in}\ \overline{\mathbf{G}} \\
\qquad\qquad\qquad\qquad \text{where}\ \ (\ell_0,\ \ell_1) = \nu \mid \mathbf{A}, \\
\qquad\qquad \nu\,\mathtt{E}\,\mathbf{I} \quad\longrightarrow\quad \mathcal{G}_2\ (\nu \mid \mathbf{I})\ \gamma,\ \bot \\
\qquad\qquad \text{where}\ \ \nu = \gamma\ l
\end{array}
$$

그림 13: G-machine 상태에서 표준 그래프 함수로의 변환 함수

$$
\begin{array}{l}
\mathcal{G} \;:\; \mathbf{S}_{\mathfrak{z}} \to\; (\mathbf{Z}_+ \;\to\; \overline{\mathbf{G}}) \\
\mathcal{G}\ (o,\ \phi,\ \gamma,\ \rho,\ \delta) \;=\; \mathcal{G}_1\ (\mathit{concat}\ (\phi : \delta))\ \gamma\ 0
\end{array}
$$

$$
\begin{array}{l}
\mathcal{G}_1 \;:\; \mathbf{L}^* \;\to\; \mathbf{G} \;\to\; \mathbf{Z}_+ \;\to\; (\mathbf{Z}_+ \;\to\; \overline{\mathbf{G}}) \\
\mathcal{G}_1\ \phi\ \gamma\ p \;=\; (\#\phi = 0) \longrightarrow \{\},\ \mathcal{G}_1\ (\mathit{tl}\ \phi)\ \gamma\ (p+1)\ \oplus\ \{p \;\mapsto\; \mathcal{G}_2\ (\mathit{hd}\ \phi)\ \gamma\}
\end{array}
$$

$$
\begin{array}{l}
\mathcal{G}_2 \;:\; \mathbf{L} \;\to\; \mathbf{G} \;\to\; \overline{\mathbf{G}} \\
\mathcal{G}_2\ \ell\ \gamma \;=\; \omega\,\mathtt{E}\,\mathtt{Id} \quad\longrightarrow\quad (\omega \mid \mathtt{Id})\ \mathbf{in}\ \overline{\mathbf{G}}, \\
\qquad\qquad \omega\,\mathtt{E}\,\mathbf{P} \quad\longrightarrow\quad \left(\begin{array}{llcl}
t = \underline{\mathtt{int}} & \longrightarrow & ((\gamma\ \ell_0) \mid \mathbf{Z})\ \mathbf{in}\ \overline{\mathbf{G}}, & \\
t = \underline{\mathtt{ap}} & \longrightarrow & (\mathcal{G}_2\ \ell_0\ \gamma,\ \mathcal{G}_2\ \ell_1\ \gamma)\ \mathbf{in}\ \overline{\mathbf{G}},\ \bot & \\
\text{where} & (t,\ r) & = & \omega \mid \mathbf{P} \\
 & \ell_0 & = & \ell + r \\
 & \ell_1 & = & \ell + r + 1
\end{array}\right), \\
\qquad\qquad \omega\,\mathtt{E}\,\mathbf{I} \quad\longrightarrow\quad \mathcal{G}_2\ (\omega \mid \mathbf{I})\ \gamma,\ \bot \\
\qquad\qquad \text{where}\ \ \omega = \gamma\ l
\end{array}
$$

그림 14: ZG-machine 상태에서 표준 그래프 함수로의 변환 함수

있다는 것을 보임으로써, 두 추상 기계의 의미가 같다는 것을 보일 수 있다.

그 전에, ZG-machine의 경우에 발생하는 특별한 성질을 정리해 보겠다. ZG-machine에서는 태그가 상대 주소와 함께 앞으로 옮겨짐에 따라 그래프 구조가 복잡해 지는데, 상태 변환 중에 실제로 그래프 함수에 기록될 상대 주소는 의미 함수 $\mathcal{N}$에 의하여 생성되는 초기화 정보 리스트에 의해 결정된다. 그러나, 의미 함수 $\mathcal{N}$의 첫번째 인수인 표현식만 같다면, $\mathcal{N}$의 세번째 인수로 주어지는 특정한 상대 주소 값은 근본적인 그래프 정보를 바꾸지 않는데, 이를 정리하면 다음과 같다.

명제 1 *임의의 표현식 $E \in$ Expr가 다른 상대 주소 값으로 ZG-machine 스택 의미정의의 의미 함수 $\mathcal{N}$에 의하여 리스트 ζ_1, ζ_2로 변환되었다고 하자. 이들 리스트에 상대 주소 값을 반영한 리스트(아래 정의된 의미 함수 $\underline{fill}$을 적용한 결과)에 대하여 $\underline{alloc}$은 그래프 합동 관계의 상태 변환 함수를 이룬다. 구체적으로*

$$\begin{aligned} \zeta_1 &= \mathcal{N}\ [\![E]\!]\ \beta\ n \\ \zeta_2 &= \mathcal{N}\ [\![E]\!]\ \beta\ (n+1) \\ &\quad \textit{where } n \in \mathbf{Z}_+, n \geq 1 \end{aligned}$$

일 때,

$$\begin{aligned} &\underline{fill}\ \zeta\ n{=}(\#\zeta = 1) \longrightarrow \zeta, \\ &\qquad [hd\ \zeta] +\!\!+ [d_1,\ d_2, \ldots, d_{n-1}] +\!\!+ (tl\ \zeta) \\ &\qquad \textit{여기서 } d_i \textit{는 임의의 원소} \end{aligned}$$

라고 하면,

$$\underline{alloc}\ (\underline{fill}\ \zeta_1\ n) \cong \underline{alloc}\ (\underline{fill}\ \zeta_2\ (n+1))$$

이다. 결과적으로 임의의 자연수 n에 대하여

$$\underline{alloc}\ (\underline{fill}\ (\mathcal{N}\ [\![E]\!]\ \beta\ n)\ n) \cong \underline{alloc}\ (\mathcal{N}\ [\![E]\!]\ \beta\ 1)$$

이다.

위 명제는 ZG-machine의 기본 함수 $\underline{alloc}$의 특징에 대해서 언급하고 있다. ZG-machine에서 같은 표현식에 대해 다른 상대 주소로 초기화 정보를 생성하고(ζ_i), 필요한 만큼 임의의 초기화 정보를 추가시켜 상대 주소가 맞도록 태그 옮김을 한다면($\underline{fill}\ \zeta_i\ n_i$, 여기서 n_i는 각 ζ_i를 생성할 때 사용된 상대 주소), 각 초기화 정보에 대해 기본 함수 $\underline{alloc}$을 적용한 결과의 상태 변환 함수는 그래프 합동 관계를 유지한다는 것을 말해주고 있다. 이 명제를 이용하여 다음 보조정리를 증명할 수 있다.

보조정리 1 *주어진 표현식과 바인딩이 같으면 각 추상 기계 의미정의의 $\mathcal{C}$ 함수는 그래프 합동 관계의 상태 변환 함수를 생성한다.*

$$\forall\ E \in \texttt{Expr},\ \beta \in \mathbf{B}\ .\ \mathcal{C}_{\mathfrak{g}}\ [\![E]\!]\ \beta \cong \mathcal{C}_{\mathfrak{z}}\ [\![E]\!]\ \beta$$

(증명) $E = I \in$ Id, $E = Z \in \mathbf{G}$인 경우를 귀납법 기초(induction basis)로, $E = E_0\ E_1$인 경우를 귀납법 단계(induction step)로 하여 구조적 귀납법(structural induction)을 이용. □

위 보조정리는 각 추상 기계 의미정의의 의미 함수 $\mathcal{C}_{\mathfrak{g}}$와 $\mathcal{C}_{\mathfrak{z}}$가 같은 표현식 E와 같은 바인딩 β에 대하여 그래프 합동 관계의 상태 변환 함수를 생성한다는 것을 말해 주고 있다. 이를 기초로 의미 함수 $\mathcal{F}$도 유사한 성질을 가지고 있음을 말할 수 있는데, 이를 정리하면 다음과 같다.

보조정리 2 *주어진 완전조합자와 바인딩이 같으면 각 추상 기계 의미정의의 의미 $\mathcal{F}$ 함수는 그래프 합동 관계의 상태 변환 함수를 생성한다.*

$$\forall\ C \in \texttt{Comb}\ .\ \mathcal{F}_{\mathfrak{g}}\ [\![C]\!]\ \{\}\ 0 \cong \mathcal{F}_{\mathfrak{z}}\ [\![C]\!]\ \{\}\ 0$$

(증명) 완전조합자의 호출 횟수에 대한 귀납법으로 증명 □

위 보조정리에 의하면, 임의의 완전조합자 C는 각 추상 기계 의미정의의 의미 함수 $\mathcal{F}$에 의해 그래프 합동 관계의 상태 변환 함수로 변환되는데, 그렇다면 의미 함수 $\mathcal{F}$는 프로그램의 최종 결과값을 나타내는 완전조합자 main에 대해서도 그래프 합동 관계가 성립하는 상태 변환 함수를

반환한다는 말이 된다. 두 추상 기계의 초기 상태는 그래프 합동 관계이고, 완전조합자 main에 대한 상태 변환 함수도 그래프 합동 관계를 유지하므로, 이제 그래프 합동 관계의 상태에 대하여 같은 출력값을 생성한다는 것만 증명하면 된다. 이는 다음의 두 보조정리에서 논하고 있다.

보조정리 3 *두 추상 기계에서 어떤 상태에 상태 변환 함수 $\underline{eval}$을 적용한 결과 최종 상태에 도달했다면, 최종 상태의 스택 최상위가 최전방 정규형(본 논문의 경우 정수 노드나 인수가 불충분한 적용 노드)이다.*

(증명) 상태 변환 함수 $\underline{eval}$은 스택 최상위만 놓아두고 나머지는 덤프한 후, $\underline{unwind}$를 호출한다. 함수 $\underline{unwind}$는 재귀적으로 정의되어 있는데, $\underline{unwind}$가 재귀적으로 호출되지 않는 경우는 스택 최상위가 정수 노드이거나 다른 완전조합자를 호출할 경우이다. 이 두 경우만 $\underline{unwind}$가 끝날 수 있는 경우이고, 따라서 $\underline{eval}$이 끝날 수 있는 경우이다. 다른 완전조합자를 호출하였을 경우 $(\underline{entry}\ n\ \kappa)$가 적용되는데, 여기서 κ 에서도 마지막에 적용되는 $(\underline{exit}\ n)$에 의해 다시 $\underline{unwind}$ 가 호출될 수 있다. 그렇지 않은 경우는 $(\underline{entry}\ n\ \kappa)$가 적용될 때 스택 상의 인수 개수가 n보다 작아서 아예 κ가 적용되지 않는 경우이다. 즉 $\underline{unwind}$가 끝나는 경우는 맨 처음 $\underline{unwind}$가 호출될 때의 스택 최상위가 정수 노드이거나 인수가 불충분한 적용 노드일 경우이다. $\underline{unwind}$가 끝난 후 호출되는 $\underline{restore}$는 덤프된 스택을 다시 복귀시키는 역할을 하고, 결과적으로 $\underline{eval}$은 스택 최상위가 최전방 정규형이 되도록 한다. □

위 보조정리는 상태 변환 함수 $\underline{eval}$의 성질에 관하여 말하고 있다. 두 추상 기계에서 똑 같이 정의되고 있는 상태 변환 함수 $\underline{eval}$은, 주어진 상태에 대하여 스택 최상위가 가리키는 노드가 최전방 정규형이 되도록 상태 변환을 수행한다. 앞에서 언급한 바와 같이 최전방 정규형은 그래프 축약 과정에서 더 이상 축약될 수 없는 형태, 즉 '값'에 해당하는 그래프를 나타내는데, 이 장에서 정의한 언어의 경우 정수값만을 지원하고 있으므로, 정수 노드나 인수가 불충분한 적용 노드를 의미한다. 상태 변환 함수 $\underline{eval}$이 최전방 정규형을 만든다는 것이 증명되면, 그래프 합동 관계의 상태에 대한 출력값에 대해서도 논할 수 있는데 이를 정리하면 다음과 같다.

보조정리 4 *각 추상 기계의 상태가 그래프 합동이고 각 상태의 출력값이 같다면 $(\underline{print} \circ \underline{eval})$에 의해 생성되는 출력값은 같다.*

$$\begin{aligned}&\forall\ \sigma_{①} \in \mathbf{S}_{①},\ \sigma_{②} \in \mathbf{S}_{②}\ .\\ &\sigma_{①} \cong \sigma_{②}\ \wedge\ fst\ \circ\ \sigma_{①} = fst\ \circ\ \sigma_{②}\ \Rightarrow\\ &(fst \circ \underline{print} \circ \underline{eval})\ \sigma_{①}\\ &= (fst \circ \underline{print} \circ \underline{eval})\ \sigma_{②}\end{aligned}$$

(증명) 그래프 합동 관계인 두 상태를 $\sigma_{①}, \sigma_{②}$라 하고, 이들에 의해 정의되는 표준 그래프 함수를 Γ라 하자.

$$\Gamma = \mathcal{G}\ \sigma_{①} = \mathcal{G}\ \sigma_{②}$$

보조정리 3에 의해서 $\underline{eval}$은 각 상태의 스택 최상위가 최전방 정규형이 되도록 한다. $\underline{print}$는 스택 최상위가 정수 노드인 경우에 대하여 출력값 위치에 그 값을 연결하는 함수이므로, 스택 최상위를 나타내는 $(\Gamma\ 0)$의 경우에 따라 $\underline{print}$의 결과가 같음을 증명하면 된다.

i) $(\Gamma\ 0)$가 정수를 나타낼 경우:
$\underline{print}$는 같은 정수값을 각 $\sigma_{①}, \sigma_{②}$의 출력값에 연결시킨다.

ii) $(\Gamma\ 0)$가 인수가 불충분한 적용 노드를 나타내는 경우:
$\underline{print}$의 정의에 의해 양변은 모두 $\perp$이 되어 같다.

i), ii)에 의하여

$$(fst \circ \underline{print} \circ \underline{eval})\ \sigma_{①} = (fst \circ \underline{print} \circ \underline{eval})\ \sigma_{②}$$

□

위 보조정리에 의하여 출력값이 같은 그래프 합동 관계의 상태가 상태 변환 함수 ($\underline{print} \circ \underline{eval}$)에 의해 같은 출력값을 갖는 상태로 변환된다는 것을 알 수 있으므로, 앞서 언급한 두 추상 기계 의미정의의 의미 함수 $\mathcal{F}$가 그래프 합동 관계의 상태 변환 함수를 생성한다는 것과 함께 생각하면, 같은 프로그램에 대하여 두 추상 기계 의미정의에 의해 생성된 출력값이 같다는 것을 증명할 수 있다. 이를 정리하면 다음과 같다.

정리 1 *같은 입력 프로그램에 대해 두 추상 기계는 같은 출력을 생성한다. 즉, 두 기계의 의미는 같다.*

$$\forall\ P \in \texttt{Prog}\ .\ fst\ (\mathcal{P}_{\mathfrak{g}}\ [\![P]\!]) = fst\ (\mathcal{P}_{\mathfrak{z}}\ [\![P]\!])$$

(증명) $\mathcal{P}_{\mathfrak{g}}$, $\mathcal{P}_{\mathfrak{z}}$의 정의에 의해

$$\begin{aligned}
\mathcal{P}_{\mathfrak{g}}\ [\![P]\!] &= (\underline{print} \circ \underline{eval} \circ (\rho_{\mathfrak{g}}\ \texttt{main}))\ \sigma_{\mathfrak{g}} \\
&\quad \text{where } \sigma_{\mathfrak{g}} = ([],\ [],\ \{\},\ \rho_{\mathfrak{g}},\ []) \\
\mathcal{P}_{\mathfrak{z}}\ [\![P]\!] &= (\underline{print} \circ \underline{eval} \circ (\rho_{\mathfrak{z}}\ \texttt{main}))\ \sigma_{\mathfrak{z}} \\
&\quad \text{where } \sigma_{\mathfrak{z}} = ([],\ [],\ \{\},\ \rho_{\mathfrak{z}},\ [])
\end{aligned}$$

이다. 여기서

$$\mathcal{G}\ \sigma_{\mathfrak{g}} = \{\} = \mathcal{G}\ \sigma_{\mathfrak{z}}$$

이므로, $\sigma_{\mathfrak{g}}$와 $\sigma_{\mathfrak{z}}$는 그래프 합동 관계이다. main에 대한 완전조합자를 $C_{\texttt{main}}$ 이라고 하면, 보조정리 2에 의하여 각 추상 기계의 (ρ main)은 그래프 합동 관계의 상태 변환 함수

$$\mathcal{F}_{\mathfrak{g}}\ [\![C_{\texttt{main}}]\!]\ \{\}\ 0\ \cong\ \mathcal{F}_{\mathfrak{z}}\ [\![C_{\texttt{main}}]\!]\ \{\}\ 0$$

를 정의하므로,

$$(\rho_{\mathfrak{g}}\ \texttt{main})\ \sigma_{\mathfrak{g}}\ \cong\ (\rho_{\mathfrak{z}}\ \texttt{main})\ \sigma_{\mathfrak{z}}$$

가 성립하고, 여기서 각 (ρ main)은 상태 변환 함수 $\underline{print}$를 포함하지 않으므로 양 변의 출력값은

$$fst\ ((\rho_{\mathfrak{g}}\ \texttt{main})\ \sigma_{\mathfrak{g}}) = [] = fst\ ((\rho_{\mathfrak{z}}\ \texttt{main})\ \sigma_{\mathfrak{z}})$$

로 같다. 따라서 보조정리 4에 의해

$$\begin{aligned}
&(fst \circ \underline{print} \circ \underline{eval})\ ((\rho_{\mathfrak{g}}\ \texttt{main})\ \sigma_{\mathfrak{g}}) \\
=\ &(fst \circ \underline{print} \circ \underline{eval})\ ((\rho_{\mathfrak{z}}\ \texttt{main})\ \sigma_{\mathfrak{z}})
\end{aligned}$$

이고, 결과적으로 다음 식이 성립한다.

$$fst\ (\mathcal{P}_{\mathfrak{g}}\ [\![P]\!]) = fst\ (\mathcal{P}_{\mathfrak{z}}\ [\![P]\!])$$

□

6. 결론

이상에서 추상 기계 G-machine과 ZG-machine의 의미가 같음을 증명하였다. 이를 증명하기 위해, 두 기계의 스택 의미정의를 먼저 기술하였고, 스택 의미정의에 따른 각 기계의 상태와 상태 변환 함수에 대하여 표준 그래프를 통한 그래프 합동 관계를 제시하였다. 최종적으로 각 스택 의미정의의 대응되는 의미 함수가 그래프 합동 관계의 상태 변환 함수를 생성함을 보이고, 출력값이 같은 그래프 합동 관계의 상태가 출력값이 같은 최종 상태로 변환됨을 보임으로써, 두 기계가 같은 의미의 상태 변환을 수행함을 증명하였다.

지금까지 보인 증명에서는 상태 변환이 끝나지 않는 경우에 대하여, 두 추상 기계의 스택 의미정의에서 공히 상태 변환이 끝나지 않음을 보였다. 그러나 일반적으로 표기적 의미론의 관점에서는 이러한 경우 최소 고정점(least fixed point) 방법에 의해 두 추상 기계의 상태가 최소값 $\perp$으로 정의된다. 최소 고정점 방법에 입각하여 두 추상기계의 의미가 같다는 것을 보이는 것은 향후 연구로 남겨둔다.

참고문헌

[1] G. Woo and T. Han. "Compressing the Graphs in the G-machine by Tag-Forwarding". *Journal of Computing and Information*, 3(1):112–138, 1998.

[2] 우균, 한태숙. "태그 옮기기에 의한 G-machine 그래프의 압축". *정보과학회논문지(B)*, 26(5), May 1999.

[3] 우균. *태그 옮김 기법으로 공간 효율을 높인 G-machine*. PhD thesis, KAIST, February 2000.

[4] L. Augustsson. "A Compiler for Lazy ML". In *Proceedings of the ACM Symposium on Lisp and Functional Programming*, pages 218–227, 1984.

[5] T. Johnsson. "Efficient Compilation of Lazy Evaluation". In *Proceedings of the ACM SIGPLAN '84 Symposium on Compiler Construction*, pages 58–69, 1984.

[6] D. Lester. "The G-Machine as a Representation of Stack Semantics". In G. Kahn, editor, *Proceedings of the ACM Conference on Functional Programming Languages and Computer Architecture*, volume 274 of *Lecture Notes in Computer Science*, pages 47–59. Springer, 1987.

[7] D. Lester. *"Combinator Graph Reduction: A Congruence and its Applications"*. PhD thesis, Oxford University, Oxford OX1 3QD, England, July 1988.

우 균
1991년
KAIST 전산학과(학사).
1993년
KAIST 전산학과(석사).
2000년
KAIST 전산학과(박사).
현재 동아대학교 전기전자컴퓨터공학부 조교수.
⟨관심분야⟩ 함수형 언어 프로그래밍 환경, 지연 함수형 언어를 위한 추상기계, 프로그램의 정적 분석 방법.

한태숙
1976년
서울대학교 전자공학과.
1978년
KAIST 전산학과(석사).
1990년
Univ. of North Carolina at Chapel Hill(박사).
현재 한국과학기술원 전자전산학과 부교수.
⟨관심분야⟩ 프로그래밍 언어론, 함수형 언어.