프로그래밍 언어 nML
The Definition of nML

Momvo Niliry
프로그램 분석 시스템 연구단
한국과학기술원

Momvo Niliry
Research On Program Analysis System
National Creative Research Initiative Center
KAIST

2002년 3월 22일

목 차

제 1 장 동기 **2**

제 2 장 문법구조 **3**
제 1 절 문법 3
제 2 절 기본 타입, 연산자, 데이타구성자, 예외상황 9
제 3 절 우선순위와 방향성 9
제 4 절 문법적인 제약들 11
제 5 절 예약된 심볼들 12
제 6 절 설탕 구조 13

제 3 장 프로그램의 실행 **15**
제 1 절 프로그램 모듈의 실행 17
제 2 절 프로그램 식의 실행 21

제 4 장 프로그램의 기획 **26**
제 1 절 프로그램 모듈의 기획 31
제 2 절 프로그램 식의 기획 37

부록 A Standard ML및 OCaml과의 비교점 **41**

제 1 장

동기

- 프로그램 분석 실험의 일터를 실제 컴파일러 구축을 통해 실현할 필요가 있다.
- 상위의 타입구조를 갖춘 (higher-order and typed) 프로그래밍 언어의 컴파일러 시스템을 독자 개발할 필요가 있다.
- SML/NJ나 OCaml 컴파일러들은 프로그램 분석 기술을 충분히 응용하고 있지 못하거나, 그 가능성을 염두에 두고 만들어지지 않았다.
- SML/NJ나 OCaml 컴파일러들은 미래의 넷트웍 컴퓨팅 환경을 염두에 두고 만들어지지 않았다. nML 컴파일러는 우리가 목표로하는 넷트웍 프로그래밍 요소를 실험할 플랫폼이 될 것이다.
- nML 프로그래밍 언어 시스템은 ML의 한국 사투리이다.
- nML 프로그래밍 환경은 OCaml의 방식대로 Unix/C 의 전통을 보존한다. C 신도가 nML 교로 개종하는데 부담을 느끼지 않도록.

제 2 장

문법구조

제 1 절 문법

다음의 표기법을 따른다:

$\mid$	alternative	()	grouping
$\langle\rangle$	optional	$\bullet^*$	zero or more
$\bullet^+$	one or more	†	sugared alternative

$$
\begin{array}{rcl}
\textit{integer} & ::= & (\texttt{0}-\texttt{9})^+ \\
& \mid & (\texttt{0X}\mid\texttt{0x})(\texttt{0}-\texttt{9}\mid\texttt{A}-\texttt{F}\mid\texttt{a}-\texttt{f})^+ \\
& \mid & (\texttt{0O}\mid\texttt{0o})(\texttt{0}-\texttt{7})^+ \\
& \mid & (\texttt{0B}\mid\texttt{0b})(\texttt{0}-\texttt{1})^+ \\
\textit{real} & ::= & (\texttt{0}-\texttt{9})^+\langle\texttt{.}(\texttt{0}-\texttt{9})^+\rangle\langle(\texttt{E}\mid\texttt{e})\langle\texttt{-}\mid\texttt{+}\rangle(\texttt{0}-\texttt{9})^+\rangle \\
\textit{string} & ::= & \texttt{"}(\textit{printableChar}\mid\textit{escapecString}\mid\textit{ignore})^*\texttt{"} \\
\textit{character} & ::= & \texttt{'}(\textit{printableChar}\mid\textit{escapecChar})\texttt{'} \\
\textit{ignore} & ::= & \texttt{\textbackslash}(\textit{newline}\mid\textit{formfeed}\mid\textit{newline formfeed})(\textit{space}\mid\textit{tab})^+ \\
\textit{escapecString} & ::= & \texttt{\textbackslash}(\texttt{b}\mid\texttt{t}\mid\texttt{n}\mid\texttt{r}\mid\texttt{\textbackslash}\mid\texttt{"}\mid(\texttt{0}-\texttt{9})(\texttt{0}-\texttt{9})(\texttt{0}-\texttt{9})) \\
\textit{escapecChar} & ::= & \texttt{\textbackslash}(\texttt{b}\mid\texttt{t}\mid\texttt{n}\mid\texttt{r}\mid\texttt{\textbackslash}\mid\texttt{'}\mid(\texttt{0}-\texttt{9})(\texttt{0}-\texttt{9})(\texttt{0}-\texttt{9})) \\
\textit{comment} & ::= & \text{balanced } \texttt{(* *)}\text{, between which any character can appear.}
\end{array}
$$

$$
\begin{array}{rcl}
\textit{alphanum} & ::= & \texttt{a}-\texttt{z}\mid\texttt{A}-\texttt{Z}\mid\textit{hangul}\mid\texttt{0}-\texttt{9}\mid\texttt{\_}\mid\texttt{'} \\
\textit{upper} & ::= & \texttt{A}-\texttt{Z}\mid\texttt{\_} \\
\textit{lower} & ::= & \texttt{a}-\texttt{z}\mid\textit{hangul} \\
\textit{hangul} & ::= & \text{syllables of KSX1001 (a.k.a. KSC5601 or eur-kr)} \\
& \mid & \text{syllables of KSX1005-1 (a.k.a. KSC5700, unicode, or ISO/IEC10646-1)} \\
\textit{sym} & ::= & \texttt{!}\mid\texttt{\%}\mid\texttt{\&}\mid\texttt{\$}\mid\texttt{\#}\mid\texttt{+}\mid\texttt{-}\mid\texttt{/}\mid\texttt{:}\mid\texttt{<}\mid\texttt{=}\mid\texttt{>}\mid\texttt{?}\mid\texttt{@}\mid\texttt{\textbackslash}\mid\texttt{\~{}}\mid\texttt{`}\mid\texttt{\^{}}\mid\texttt{|}\mid\texttt{*}
\end{array}
$$

$$
\begin{array}{rcl}
\mathit{lid} & ::= & \mathit{lower}\,(\mathit{alphanum})^* \\
\mathit{uid} & ::= & \mathit{upper}\,(\mathit{alphanum})^* \\
\mathit{varid} & ::= & \mathit{lid} \\
\mathit{prefixid} & ::= & (\texttt{!} \mid \texttt{?} \mid \texttt{\~{}})\mathit{sym}^* \\
\mathit{infixid} & ::= & (\texttt{\%} \mid \texttt{\&} \mid \texttt{\$} \mid \texttt{\#} \mid \texttt{+} \mid \texttt{-} \mid \texttt{/} \mid \texttt{:} \mid \texttt{<} \mid \texttt{=} \mid \texttt{>} \mid \texttt{@} \mid \texttt{\textbackslash} \mid \texttt{`} \mid \texttt{\^{}} \mid \texttt{|} \mid \texttt{*})\mathit{sym}^* \\
\mathit{opid} & ::= & \mathit{prefixid} \mid \mathit{infixid} \\
\mathit{tyid} & ::= & \mathit{lid} \\
\mathit{conid} & ::= & \mathit{uid} \\
\mathit{strid} & ::= & \mathit{uid} \\
\mathit{sigid} & ::= & \mathit{uid} \\
\mathit{fctid} & ::= & \mathit{uid} \\
\mathit{tyvar} & ::= & \texttt{'}\mathit{lid} \\
\mathit{lab} & ::= & (\texttt{0}-\texttt{9})(\texttt{0}-\texttt{9})^* \mid \mathit{lid} \\
\mathit{varlongid} & ::= & \mathit{varid} \mid \mathit{strid}\,\texttt{.}\,\mathit{varlongid} \\
\mathit{oplongid} & ::= & \mathit{opid} \mid \mathit{strid}\,\texttt{.}\,\mathit{oplongid} \\
\mathit{tylongid} & ::= & \mathit{tyid} \mid \mathit{strid}\,\texttt{.}\,\mathit{tylongid} \\
\mathit{conlongid} & ::= & \mathit{conid} \mid \mathit{strid}\,\texttt{.}\,\mathit{conlongid} \\
\mathit{strlongid} & ::= & \mathit{strid} \mid \mathit{strid}\,\texttt{.}\,\mathit{strlongid}
\end{array}
$$

$$
\begin{array}{rcll}
\mathit{ty} & ::= & \mathit{tyvar} & \\
 & \mid & \texttt{\{}\mathit{tyrow}\texttt{\}} & \\
 & \mid & \mathit{tyseq}\ \mathit{tylongid} & \\
 & \mid & \mathit{ty}_1\ \texttt{->}\ \mathit{ty}_2 & \\
 & \mid & \texttt{(}\mathit{ty}\texttt{)} & \\
 & \dagger & \mathit{ty}_1\ \texttt{*}\ \mathit{ty}_2 & \\
 & & & \\
\mathit{tyrow} & ::= & \mathit{lab}\ \texttt{:}\ \mathit{ty}\ \langle\texttt{,}\mathit{tyrow}\rangle & \\
\mathit{tyseq} & ::= & \mathit{ty} & \\
 & \mid & & \text{empty sequence} \\
 & \mid & (\mathit{ty}_1, \cdots, \mathit{ty}_k) & k \geq 1 \\
\mathit{tyvarseq} & ::= & \mathit{tyvar} & \\
 & \mid & & \text{empty sequence} \\
 & \mid & (\mathit{tyvar}_1, \cdots, \mathit{tyvar}_k) & k \geq 1
\end{array}
$$

pat	::=	`_`	wild pattern
	|	*integer* | *string* | *character*	constant pattern
	|	`{` *patrow* `}` | `{}`	record pattern
	|	`[|` *pat* (`,` *pat*)* `|]` | `[||]`	array pattern
	|	`(` *pat* `)`	
	|	`ref` *pat*	
	|	*conlongid* ⟨*pat*⟩	datacon pattern
	|	*varid*	pattern variable
	|	`(` *opid* `)`	operator pattern variable
	|	*pat* `:` *ty*	pattern with type
	|	(*varid* | `(` *opid* `)`) ⟨`:` *ty*⟩ `as` *pat*	as pattern
	|	pat_1 `|` pat_2	or pattern
	|	*pat* `::` *pat*	cons pattern
	†	(*pat* (`,` *pat*)$^+$`)`	tuple pattern
	†	`[`*pat* (`,` *pat*)*`]` † `[]`	list pattern
	†	`()`	unit pattern

patrow	::=	`...`	
	|	*lab* `=` *pat* ⟨`,` *patrow*⟩	
	†	*lid*⟨`,` *patrow*⟩	

e	::=	*const*	
	|	*varlongid*	
	|	*conlongid*	
	|	`(` *oplongid* `)`	
	|	`{` *labrow* `}` | `{}`	record
	|	e_1 `.` *lab*	record field
	|	`[|` ⟨*elmtrow*⟩ `|]` | `[||]`	array
	|	e_1 `.[` e_2 `]`	array field
	|	e_1 `.[` e_2 `]` `<-` e_3	array update
	|	`let` *valdec* `in` e `end`	binding
	|	`(` e `)`	
	|	e_1 e_2	application
	|	e `:` *ty*	
	|	e `handle` *match*	
	|	`raise` e	
	|	`fn` *match*	function
	†	`fn` *curmatch*	curried function
	|	e_1 `:=` e_2	assignment
	|	`!` e	dereference
	|	`ref` e	reference
	†	*prefixid* e	prefix application
	†	e_1 *infixid* e_2	infix application
	†	e_1 `{` *lab* `<-` e_2 `}`	new record
	†	e_1 `;` e_2	sequence
	†	`case` e `of` *match*	
	†	`if` e_1 `then` e_2 `else` e_3	
	†	`if` e_1 `then` e_2	
	†	`(` e `,` *elmtrow* `)`	tuple
	†	`[` ⟨*elmtrow*⟩ `]`	list
	†	`while` e_1 `do` e_2 `end`	
	†	`for` *varid* `=` e_1`;` e_2`;` e_3 `do` e_4 `end`	
	†	`()`	

const	::=	*integer* | *real* | *string* | *character*
curmatch	::=	pat^+ `=>` e ⟨`|` *curmatch*⟩
match	::=	*pat* `=>` e ⟨`|` *match*⟩
labrow	::=	*lab* `=` e ⟨`,` *labrow*⟩
elmtrow	::=	e ⟨`,` *elmtrow*⟩

$$
\begin{array}{rcl}
\mathit{dec} & ::= & \mathit{valdec} \\
& | & \texttt{type}\ \mathit{typbind} \\
& | & \texttt{exception}\ \mathit{exnbind} \\
& | & \texttt{local}\ \mathit{dec}_1\ \texttt{in}\ \mathit{dec}_2\ \texttt{end} \\
& | & \mathit{dec}_1\ \langle ; \rangle\ \mathit{dec}_2 \\
& \dagger & \texttt{open}\ \mathit{strlongid} \\
\\
\mathit{valdec} & ::= & \texttt{val}\ \mathit{tyvarseq}\ \mathit{valbind} \\
& \dagger & \texttt{fun}\ \mathit{funbind} \\
& | & \mathit{valdec}\ \langle ; \rangle\ \mathit{valdec} \\
\\
\mathit{strdec} & ::= & \mathit{dec} \\
& | & \texttt{structure}\ \mathit{strbind} \\
& | & \mathit{strdec}\ \langle ; \rangle\ \mathit{strdec} \\
\\
\mathit{fntordec} & ::= & \texttt{functor}\ \mathit{fctid}\ \texttt{(}\ \mathit{strid}\ \texttt{:}\ \mathit{sig}_1\ \texttt{)}\ \texttt{=}\ \mathit{strexp} \\
& \dagger & \texttt{functor}\ \mathit{fctid}\ \texttt{(}\ \mathit{strid}\ \texttt{:}\ \mathit{sig}_1\ \texttt{)}\ \texttt{:}\ \mathit{sig}_2\ \texttt{=}\ \mathit{strexp} \\
& \dagger & \texttt{functor}\ \mathit{fctid}\ \texttt{(}\ \mathit{strid}_1\ \texttt{:}\ \mathit{sig}_1\ (\texttt{,}\ \mathit{strid}_2\ \texttt{:}\ \mathit{sig}_2)^+\ \texttt{)}\ \langle \texttt{:}\ \mathit{sig}_3 \rangle\ \texttt{=}\ \mathit{strexp} \\
\\
\mathit{sigdec} & ::= & \texttt{signature}\ \mathit{sigbind} \\
\\
\mathit{strexp} & ::= & \mathit{strlongid} \\
& | & \texttt{struct}\ \langle \mathit{strdec} \rangle\ \texttt{end} \\
& | & \mathit{strexp}\ \texttt{:}\ \mathit{sig} \\
& | & \mathit{fctid}\ \texttt{(}\mathit{strexp}\texttt{)} \\
& \dagger & \mathit{fctid}\ \texttt{(}\mathit{strexp}\ (\texttt{,}\ \mathit{strexp})^+\texttt{)} \\
\\
\mathit{valbind} & ::= & \mathit{pat}\ \texttt{=}\ e\ \langle \texttt{and}\ \mathit{valbind} \rangle \\
& | & \texttt{rec}\ \mathit{valbind} \\
\mathit{funrule} & ::= & (\mathit{varid}\ |\ \texttt{(}\ \mathit{opid}\ \texttt{)})\ \mathit{pat}^+ \langle \texttt{:}\ \mathit{ty} \rangle\ \texttt{=}\ e \\
\mathit{funbind} & ::= & \mathit{funrule}\ (|\ \mathit{funrule})^*\ \langle \texttt{and}\ \mathit{funbind} \rangle \\
\mathit{typbind} & ::= & \mathit{tyvarseq}\ \mathit{tyid}\ \texttt{=}\ \mathit{ty}\ \langle \texttt{and}\ \mathit{typbind} \rangle \\
& | & \mathit{tyvarseq}\ \mathit{tyid}\ \texttt{=}\ \mathit{conbind}\ \langle \texttt{and}\ \mathit{typbind} \rangle \\
\mathit{conbind} & ::= & \mathit{conid}\ \langle \texttt{of}\ \mathit{ty} \rangle\ \langle |\ \mathit{conbind} \rangle \\
\mathit{exnbind} & ::= & \mathit{conid}\ \langle \texttt{of}\ \mathit{ty} \rangle\ \langle \texttt{and}\ \mathit{exnbind} \rangle \\
\mathit{sigbind} & ::= & \mathit{sigid}\ \texttt{=}\ \mathit{sig} \\
\mathit{strbind} & ::= & \mathit{strid}\ \texttt{=}\ \mathit{strexp}\ \langle \texttt{and}\ \mathit{strbind} \rangle \\
& \dagger & \mathit{strid}\ \texttt{:}\ \mathit{sig}\ \texttt{=}\ \mathit{strexp}
\end{array}
$$

$$
\begin{array}{rcl}
\mathit{sig} & ::= & \mathit{sigid} \\
& | & \texttt{sig}\ \langle \mathit{spec} \rangle\ \texttt{end} \\
& | & \mathit{sig}\ \mathit{where} \\
\\
\mathit{spec} & ::= & \texttt{val}\ \mathit{valdesc} \\
& | & \texttt{type}\ \mathit{typdesc} \\
& | & \texttt{exception}\ \mathit{exndesc} \\
& | & \texttt{include}\ \mathit{sig} \\
& | & \texttt{structure}\ \mathit{strdesc} \\
& | & \mathit{spec}\ \langle \texttt{;} \rangle\ \mathit{spec} \\
\\
\mathit{valdesc} & ::= & (\mathit{varid} \mid \texttt{(}\ \mathit{opid}\ \texttt{)})\ \texttt{:}\ \mathit{ty}\ \langle \texttt{and}\ \mathit{valdesc} \rangle \\
\mathit{typdesc} & ::= & \mathit{tyvarseq}\ \mathit{tyid}\ \langle \texttt{and}\ \mathit{typdesc} \rangle \\
& | & \mathit{tyvarseq}\ \mathit{tyid}\ \texttt{=}\ \mathit{ty}\ \langle \texttt{and}\ \mathit{typdesc} \rangle \\
& | & \mathit{tyvarseq}\ \mathit{tyid}\ \texttt{=}\ \mathit{condesc}\ \langle \texttt{and}\ \mathit{typdesc} \rangle \\
\mathit{condesc} & ::= & \mathit{conid}\ \langle \texttt{of}\ \mathit{ty} \rangle\ \langle \texttt{|}\ \mathit{condesc} \rangle \\
\mathit{exndesc} & ::= & \mathit{conid}\ \langle \texttt{of}\ \mathit{ty} \rangle\ \langle \texttt{and}\ \mathit{exndesc} \rangle \\
\mathit{strdesc} & ::= & \mathit{strid}\ \texttt{:}\ \mathit{sig}\ \langle \texttt{and}\ \mathit{strdesc} \rangle \\
\\
\mathit{where} & ::= & \texttt{where type}\ \mathit{longtypbind} \\
\mathit{longtypbind} & ::= & \mathit{tyvarseq}\ \mathit{tylongid}\ \texttt{=}\ \mathit{ty}\ \langle \texttt{and}\ \mathit{longtypbind} \rangle \\
\\
\mathit{topdec} & ::= & \mathit{sigdec} \\
& | & \mathit{fntordec} \\
& | & \mathit{strdec} \\
& | & \mathit{topdec}_1\ \langle \texttt{;} \rangle\ \mathit{topdec}_2
\end{array}
$$

제 2 절 기본 타입, 연산자, 데이타구성자, 예외상황

- 기본 타입 :

  `unit`, `bool`, `int`, `real`, `string`, `char`, `list`, `ref`, `exn`, `array`

- 기본 연산자 :

`+,-,*,/`	(infix) `int` → `int` → `int`	int add/sub/mul/div
	(infix) `real` → `real` → `real`	real add/sub/mul/div
`+`	(prefix) `int` → `int`	identity
	(prefix) `real` → `real`	identity
`-`	(prefix) `int` → `int`	int negation
	(prefix) `real` → `real`	real negation
`%`	(infix) `int` → `int` → `int`	int modulus
`**`	(infix) `int` → `int` → `int`	int exponential
	(infix) `real` → `real` → `real`	real exponential
`++, --`	(postfix) `int ref` → `unit`	int inc/dec side effect
	(postfix) `real ref` → `unit`	real inc/dec side effect
`+=,-=,*=,/=`	(infix) `int ref` → `int` → `unit`	int add/sub/mul/div side effect
	(infix) `real ref` → `real` → `unit`	real add/sub/mul/div side effect
`<<, >>`	(infix) `int` → `int` → `int`	bitwise arithematic shift left/right
`andalso,&&`	(infix) `bool` → `bool` → `bool`	boolean and
`orelse,||`	(infix) `bool` → `bool` → `bool`	boolean or
`not`	(prefix) `bool` → `bool`	boolean negation
`=`	(infix) `'a` → `'a` → `bool`	equality
`<>`	(infix) `'a` → `'a` → `bool`	inequality
`<,<=,>,>=`	(infix) `'a` → `'a` → `bool`	int comparison
`@`	(infix) `'a list` → `'a list` → `'a list`	list concatenation
`^`	(infix) `string` → `string` → `string`	string concatenation

- 기본 데이타구성자와 예외상황

`True,true`	`bool`	true
`False,false`	`bool`	false
`Nil,nil`	`'a list`	empty list
`::`	(infix) `'a` × `'a list` → `'a list`	list cons
`Match`		pattern match exception
`Zero`		division exception
`Overflow`		overflow exception
`Bound`		array/record bound exception
`Equality`		equality check exception

제 3 절 우선순위와 방향성

"⋆···" 은 ⋆로 시작하는 이름들을 뜻한다.

- 프로그램식에서의 우선순위(내림차순)와 방향성

expression constructs	associativity
`ref`, *prefixop*, `++` (postfix), `--` (postfix)	right
`.`, `.[`	left
new record	left
data constructor application	left
function application	left
`+` (prefix), `-` (prefix)	right
`**`$\cdots$	right
`*`$\cdots$, `/`$\cdots$, `%`$\cdots$	left
`+`$\cdots$, `-`$\cdots$	left
`::`	right
`@`$\cdots$, `^`$\cdots$	right
other infix operators	left
`not`	right
`andalso`, `&&`	right
`orelse`, `||`	right
`:=`, `+=`, `-=`, `*=`, `/=`, `<-`	right
`,`	left
`:`	left
`raise`	right
`if`	right
`;`	right
`case`, `fn`, `handle`	right

prefix 프로그램 구성자 여러 개가 infix 구성자와 섞어 쓰일 때는 가장 안쪽의 prefix 구성자의 우선순위를 따른다.

예, `if` $\cdots$ `else fn x =>` e_3`;` e_4 는 `if` $\cdots$ `else (fn x =>` (e_3`;` e_4`))` 로 해석한다. 즉, 가장 안쪽의 `fn`이 `;` 보다 우선 순위가 낮으므로 e_3`;` e_4가 먼저 묶인다.

- 값 선언에서의 우선순위(내림차순)와 방향성

valbind constructs	associativity
`and`	–
`rec`	–

- 패턴에서의 우선순위(내림차순)와 방향성

pattern constructs	associativity
`ref`	–
data constructor application	left
curried patterns	–
`::`	right
`|`	–
`:`	–
`as`	–

- 타입식에서의 우선순위(내림차순)와 방향성

type constructs	associativity
type constructor application	–
`*` (tuple type)	–
`->` (function type)	right
`where` (signature constraint)	left

제 4 절 문법적인 제약들

- 하나의 타입 정의 뭉치 (*typbind*) 에서 선언되는 타입이름들은 모두 달라야 한다.

  어긴 예, "`type t = A | B and t = C`"

- 하나의 데이타 타입 정의 뭉치에서 선언되는 데이타 구성자 이름들은 모두 달라야 한다.

  어긴 예, "`type t = A | A`"혹은 "`type t = A | B and f = A`"

- 하나의 값 정의 뭉치 (*valbind*) 에서 선언되는 값의 이름들은 모두 달라야 한다.

  어긴 예, "`val x = 1 and x = 2`"

- 재귀적인 값 정의에서 (`rec` *valbind*) `=`의 오른쪽에는 오직 "`fn` *match*"만 올 수 있다.

  어긴 예, "`val rec f = 2`"

- 재귀적인 값 정의에서 (`rec` *valbind*) 에서 `=`의 왼편 패턴은 패턴변수만 가능하다.

  어긴 예, "`val rec f as g = fn x => f g x`"

- 무더기 패턴(or-pattern)을 제외하고, 하나의 패턴에서 같은 패턴 변수가 여러번 올 수 없다.

  어긴 예, "`case` e `of (x,x) =>` e'"

- 하나의 접속방안 (*spec*)에서 같은 이름의 선언이 여러번 나타날 수 없다.

  어긴 예, "`sig type t type t end`" 혹은 "`sig val t val t end`"

- 예약된 심볼들은 프로그램에서 재 정의될 수 없다.
  어긴 예, “`val int = 10`” 혹은 “`type 'for char = True of 'for list`”

제 5 절 예약된 심볼들

```
and andalso case do else end exception fn for fun functor
handle if in include let local of open orelse raise rec ref
sig signature struct structure then type val where while
:= ! -> <- => | : ; { } [ ] [| |] , ( ) .

array bool char exn int list real ref string unit

nil not true false ++ -- && ||
```

제 6 절 설탕 구조

자유로운 이름이 설탕이 녹으면서 묶이지 않도록한다.

$(pat_0, \cdots, pat_n)$	$\equiv$	{0 = pat_0, $\cdots$, n = pat_n}	$n \geq 1$
$[pat_0, \cdots, pat_n]$	$\equiv$	pat_0::$\cdots$::pat_n::nil	$n \geq 0$
$ty_0 * \cdots * ty_n$	$\equiv$	{0:ty_0, $\cdots$, n:ty_n}	
()	$\equiv$	{}	
$(e_0, \cdots, e_n)$	$\equiv$	{0 = e_0, $\cdots$, n = e_n}	$n \geq 1$
$[e_0, \cdots, e_n]$	$\equiv$	e_0::$\cdots$::e_n::nil	$n \geq 0$
[]	$\equiv$	nil	
{*lid*⟨, *patrow*⟩}	$\equiv$	{*lid* = *lid*⟨, *patrow*⟩}	
e_1 *infixid* e_2	$\equiv$	(*infixid*) (e_1,e_2)	
prefixid e	$\equiv$	(*prefixid*) e	
e_1 { *lab* <- e_2 }	$\equiv$	let val x = e_1 in {*lab*=e_2,l_1=x.l_1,$\cdots$,l_n=x.l_n}end (단, e_2의 타입이 τ일 때, e_1의 타입은 $\{lab \mapsto \tau, l_1 \mapsto \tau_1, \cdots, l_n \mapsto \tau_n\}$)	
fn $pat_{00} \cdots pat_{0m}$ => e_0 $\cdots$ | $pat_{n0} \cdots pat_{nm}$ => e_n	$\equiv$	fn x_0 => $\cdots$ => fn x_m => case (x_0,$\cdots$,x_m) of (pat_{00},$\cdots$,pat_{0m}) => e_0 $\cdots$ | (pat_{n0},$\cdots$,pat_{nm}) => e_n (단, $m > 0$일 경우만)	
fun (*varid* | (*opid*)) $pat_{00} \cdots pat_{0m}$: *ty* = e_0 $\cdots$ | (*varid* | (*opid*)) $pat_{n0} \cdots pat_{nm}$: *ty* = e_n	$\equiv$	val rec (*varid* | (*opid*)) = fn x_0 => $\cdots$ => fn x_m => (case (x_0,$\cdots$,x_m) of (pat_{00},$\cdots$,pat_{0m}) => e_0 $\cdots$ | (pat_{n0},$\cdots$,pat_{nm}) => e_n) : *ty*	
e_1 ; e_2	$\equiv$	(fn _ => e_2) e_1	
case e of *match*	$\equiv$	(fn *match*) e	
if e_1 then e_2 else e_3	$\equiv$	case e_1 of true => e_2 | false => e_3	
if e_1 then e_2	$\equiv$	if e_1 then e_2;() else ()	
while e_1 do e_2 end	$\equiv$	let val rec f = fn _=> if e_1 then (e_2; f()) else () in f() end	
for *varid* = e_1; e_2; e_3 do e_4 end	$\equiv$	let val rec f = fn *varid* => if e_2 then (e_4; f(e_3)) else () in f e_1 end	
e_1 andalso e_2 or e_1 && e_2	$\equiv$	if not e_1 then false else e_2	
e_1 orelse e_2 or e_1 || e_2	$\equiv$	if e_1 then true else e_2	

`functor` *fctid* `(` *strid* `:` sig_1 `)` `:` sig_2 `=` *strexp*

$\equiv$ `functor` *fctid* `(` *strid* `:` sig_1 `)` `=` *strexp* `:` sig_2

`functor` *fctid* `(` $strid_1$ `:` sig_1 `,` $\cdots$ `,` $strid_n$ `:` sig_n `)` $\langle$`:` *sig*$\rangle$ `=` *strexp*

$\equiv$ `functor`
fctid `(S: sig`
`structure` $strid_1$ `:` sig_1
$\vdots$
`and` $strid_n$ `:` sig_n
`end)`
`=` $\{$`S.`$strid_1/strid_1, \cdots,$ `S.`$strid_n/strid_n\}$*strexp* $\langle$`:` *sig*$\rangle$
(*strexp*에 있는 $strid_i$를 `S.`$strid_i$로 바꾼다)

fctid `(`$strexp_1$ `,` $\cdots$ `,`$strexp_n$`)` $\equiv$ *fctid* `(struct`
`structure` $strid_1$ `=` $strexp_1$
$\vdots$
`and` $strid_n$ `=` $strexp_n$
`end)`
($strid_i$들은 모듈함수 *fctid*의 인자 이름들)

`structure` *strid* `:` *sig* `=` *strexp* $\equiv$ `structure` *strid* `=` *strexp* `:` *sig*

sig `where type` $longtypbind_1$ `and` $\cdots$ `and` $longtypbind_n$

$\equiv$ *sig* `where type` $longtypbind_1 \cdots$ `where type` $longtypbind_n$

`open` *strlongid* $\equiv$
`open`의 유효범위에 있는 자유변수 x가 *strlongid*에서 정의된 이름이라면, *strlongid* `.`x로 바꾼다.

- `open`의 유효범위는 컴파일 단위내에서 그 `open`식을 포함하는 가장 큰 *strdec*이 다음과 같을때 $strdec_2$ 로 정의된다:

$$strdec_1 \ \texttt{open}\ strlongid \ \ strdec_2.$$

제 3 장

프로그램의 실행

$$
\begin{array}{rcrcl}
v & \in & \mathit{Val} & = & \mathit{Constant} \cup \mathit{Data} \cup \mathit{Exn} \cup \mathit{Record} \cup \mathit{Array} \cup \\
 & & & & \mathit{Addr} \cup \mathit{Closure} \cup \mathit{PrimOp} \\
c & \in & \mathit{Constant} & = & \mathit{Integer} \cup \mathit{Real} \cup \mathit{String} \cup \mathit{Character} \cup \mathit{Bool} \\
d & \in & \mathit{Data} & = & \mathit{ConId} \cup (\mathit{ConId} \times \mathit{Val}) \\
x, [x] & \in & \mathit{Exn} & = & \mathit{ConId} \cup (\mathit{ConId} \times \mathit{Val}) \\
\mathit{vid} & \in & \mathit{VId} & = & \mathit{VarId} \cup \mathit{OpId} \cup \mathit{ConId} \\
r & \in & \mathit{Record} & = & \mathit{Lab} \xrightarrow{\text{fin}} \mathit{Val} \\
y & \in & \mathit{Array} & = & \mathit{Integer} \xrightarrow{\text{fin}} \mathit{Addr} \\
 & & \mathit{Closure} & = & \mathit{Match} \times \mathit{Env} \times \mathit{ValEnv} \\
s & \in & \mathit{Store} & = & \mathit{Addr} \xrightarrow{\text{fin}} \mathit{Val} \\
a & \in & \mathit{Addr} & & \\
E & \in & \mathit{Env} & = & \mathit{ValEnv} \times \mathit{StrEnv} \times \mathit{TyEnv} \\
\mathit{VE} & \in & \mathit{ValEnv} & = & \mathit{VId} \xrightarrow{\text{fin}} \mathit{Val} \\
\mathit{SE} & \in & \mathit{StrEnv} & = & \mathit{StrId} \xrightarrow{\text{fin}} \mathit{Env} \\
\mathit{TE} & \in & \mathit{TyEnv} & = & \mathit{TyId} \xrightarrow{\text{fin}} \mathit{ValInt} \\
 & & & & \\
B & \in & \mathit{Basis} & = & \mathit{FunEnv} \times \mathit{SigEnv} \times \mathit{Env} \\
F & \in & \mathit{FunEnv} & = & \mathit{FunId} \xrightarrow{\text{fin}} (\mathit{StrId} \times \mathit{Int}) \times \mathit{StrExp} \times \mathit{Basis} \\
G & \in & \mathit{SigEnv} & = & \mathit{SigId} \xrightarrow{\text{fin}} \mathit{Int} \\
I & \in & \mathit{Int} & = & \mathit{ValInt} \times \mathit{StrInt} \times \mathit{TyInt} \\
\mathit{VI} & \in & \mathit{ValInt} & = & \mathrm{Fin}(\mathit{VId}) \\
\mathit{SI} & \in & \mathit{StrInt} & = & \mathit{StrId} \xrightarrow{\text{fin}} \mathit{Int} \\
\mathit{TI} & \in & \mathit{TyInt} & = & \mathit{TyId} \xrightarrow{\text{fin}} \mathit{ValInt}
\end{array}
$$

$A \xrightarrow{\text{fin}} B$: 집합 A의 유한한 부분집합에서 집합 B로 가는 함수들의 집합.

$\mathrm{Fin}(S)$: 집합 S의 유한한 부분집합들의 집합

가정 : 프로그램의 실행에는 프로그램 텍스트에서의 "`of` *ty*", "`:`*ty*", "*tyvarseq*", "`where type` *longtypbind*"와 연산자의 새치기(fixity)는 없는것으로 한다.

메모리 축적 : 메모리 상태는 축적된다. 필요할 때를 제외하고는 룰에서 이를 드러내진 않고 항상 그러한 것으로 간주한다. 예를들어,

$$\frac{A \vdash phrase_1 \Rightarrow b \quad A' \vdash phrase_2 \Rightarrow b'}{A \vdash phrase \Rightarrow b'}$$

는 메모리 상태 s 가 다음과 같이 축적되는 것을 생략한 것이다:

$$\frac{s, A \vdash phrase_1 \Rightarrow b, s_1 \quad s_1, A' \vdash phrase_2 \Rightarrow b', s_2}{s, A \vdash phrase \Rightarrow b', s_2}$$

예외상황 전파 : 발생한 예외상황은 전파된다. 예를 들어,

$$\frac{A \vdash phrase_1 \Rightarrow b \quad A' \vdash phrase_2 \Rightarrow b'}{A \vdash phrase \Rightarrow b'}$$

는 발생한 예외상황이 전파되는 다음의 두개의 룰을 함께 정의하는 것으로 한다:

$$\frac{A \vdash phrase_1 \Rightarrow [x]}{A \vdash phrase \Rightarrow [x]}$$

$$\frac{A \vdash phrase_1 \Rightarrow b \quad A' \vdash phrase_2 \Rightarrow [x]}{A \vdash phrase \Rightarrow [x]}$$

$\langle\rangle$: 의미구조의 규칙에서도 $\langle\rangle$는 문법구조에서 처럼, 덧 붙일 수 있는 것을 표현한다. 예를들어,

$$\frac{A\ \langle B\rangle}{C\ \langle D\rangle}$$

는 다음의 두가지 규칙을 의미한다:

$$\frac{A}{C} \qquad \frac{A\ B}{C\ D}$$

a/b: "a/b"는 "a 혹은 b"를 표현한다. 여러뭉치가 사용될 때는 앞의 것은 앞의 것 끼리, 뒤에 것은 뒤에 것끼리 있는 것을 표현한다. 에를들어,

$$\frac{A\ a'/b'}{B\ a/b}$$

는 다음의 두가지 규칙을 의미한다:

$$\frac{A\ a'}{B\ a} \qquad \frac{A\ b'}{B\ b}$$

$f + g$: "$f + \{x \mapsto y\}$"는 함수 f의 정의영역에서 x 엔트리를 y로 바꾸거나 ($x \in \mathrm{Dom}\, f$인 경우) 확장한다 ($x \notin \mathrm{Dom}\, f$인 경우). 일반적으로 "$f + g$"는 함수 g가 함수 f를 바꾸거나 확장하는 것인데, 필요하면 g를 f의 타입 A로 확장시킨 후에 (g in A) 정의되는 것으로 한다.

B of A: $A = \langle \cdots, B, \cdots \rangle$일때 B를 뜻한다.

g in A: "g in A"는 G의 원소 g를 A의 원소로 만드는데, 이 때 A의 원소가 가져야 하는 부품은 공집합으로 한다. 예를들어, $A = G \times H$이고 $H = X \xrightarrow{\text{fin}} Y$일 때 "$g$ in A"는 "$\langle g, \{\} \rangle$"를 뜻한다.

$[x]$: 예외상황 값 x를 "$[x]$"로 쓰면, 발생한 예외상황을 뜻한다.

모듈 접속 $\boxed{E \downarrow I}$

$$\frac{VE \downarrow VI = VE' \quad SE \downarrow SI = SE' \quad TE \downarrow TI = TE'}{(VE, SE, TE) \downarrow (VI, SI, TI) = (VE', SE', TE')} \tag{3.1}$$

$$\overline{VE \downarrow VI = \{vid \mapsto VE(vid) \mid vid \in VI\}} \tag{3.2}$$

$$\overline{SE \downarrow SI = \{strid \mapsto E \downarrow I \mid SE(strid) = E, SI(strid) = I\}} \tag{3.3}$$

$$\overline{TE \downarrow TI = \{tyid \mapsto VI' \mid TE(tyid) = VI, TI(tyid) = VI'\}} \tag{3.4}$$

실행 환경 펼치기 $\boxed{\mathrm{rec}\, VE}$

$$\frac{VE(vid) = (match, E, VE')}{\mathrm{rec}\, VE(vid) = (match, E, VE)} \tag{3.5}$$

$$\frac{VE(vid) = v \notin Closure}{\mathrm{rec}\, VE(vid) = v} \tag{3.6}$$

제 1 절 프로그램 모듈의 실행

선언 $\boxed{E \vdash dec \Rightarrow E'}$

$$\frac{E \vdash valdec \Rightarrow VE}{E \vdash valdec \Rightarrow VE \text{ in } Env} \tag{3.7}$$

$$\frac{\vdash typbind \Rightarrow VE, TE}{E \vdash \texttt{type}\ typbind \Rightarrow (VE, TE) \text{ in } Env} \tag{3.8}$$

$$\frac{\vdash exnbind \Rightarrow VE}{E \vdash \texttt{exception}\ exnbind \Rightarrow VE \text{ in } Env} \tag{3.9}$$

$$\frac{E \vdash dec_1 \Rightarrow E_1 \quad E + E_1 \vdash dec_2 \Rightarrow E_2}{E \vdash \texttt{local}\ dec_1\ \texttt{in}\ dec_2\ \texttt{end} \Rightarrow E_2} \tag{3.10}$$

$$\frac{E \vdash dec_1 \Rightarrow E_1 \quad E + E_1 \vdash dec_2 \Rightarrow E_2}{E \vdash dec_1\ \langle ; \rangle\ dec_2 \Rightarrow E_1 + E_2} \tag{3.11}$$

값 선언

$\boxed{E \vdash valdec \Rightarrow VE}$

$$\frac{E \vdash valbind \Rightarrow VE}{E \vdash \texttt{val}\ valbind \Rightarrow VE} \tag{3.12}$$

$$\frac{E \vdash valdec_1 \Rightarrow VE_1 \quad E + VE_1 \vdash valdec_1 \Rightarrow VE_2}{E \vdash valdec_1\ \langle ; \rangle\ valdec_2 \Rightarrow VE_1 + VE_2} \tag{3.13}$$

모듈 선언

$\boxed{B \vdash strdec \Rightarrow E}$

$$\frac{E \text{ of } B \vdash dec \Rightarrow E'}{B \vdash dec \Rightarrow E'} \tag{3.14}$$

$$\frac{B \vdash strbind \Rightarrow SE}{B \vdash \texttt{structure}\ strbind \Rightarrow SE \text{ in } Env} \tag{3.15}$$

$$\frac{B \vdash strdec_1 \Rightarrow E_1 \quad B + E_1 \vdash strdec_2 \Rightarrow E_2}{B \vdash strdec_1\ \langle ; \rangle\ strdec_2 \Rightarrow E_1 + E_2} \tag{3.16}$$

모듈함수 선언

$\boxed{B \vdash fntordec \Rightarrow F}$

$$\frac{G \text{ of } B \vdash sig_1 \Rightarrow I}{B \vdash \texttt{functor}\ fctid\ \texttt{(}\ strid\ \texttt{:}\ sig_1\ \texttt{)}\ \texttt{=}\ strexp \Rightarrow \{fctid \mapsto (strid : I, strexp, B)\}} \tag{3.17}$$

모듈타입 선언

$\boxed{B \vdash sigdec \Rightarrow G}$

$$\frac{B \vdash sigbind \Rightarrow G}{B \vdash \texttt{signature}\ sigbind \Rightarrow G} \tag{3.18}$$

모듈 식

$\boxed{B \vdash strexp \Rightarrow E}$

$$\frac{B(strlongid) = E}{B \vdash strlongid \Rightarrow E} \tag{3.19}$$

$$\frac{\langle B \vdash strdec \Rightarrow E \rangle}{B \vdash \texttt{struct}\ \langle strdec \rangle\ \texttt{end} \Rightarrow \{\}\langle +E \rangle} \tag{3.20}$$

$$\frac{B \vdash strexp \Rightarrow E \quad G \text{ of } B \vdash sig \Rightarrow I}{B \vdash strexp\ \texttt{:}\ sig \Rightarrow E \downarrow I} \tag{3.21}$$

$$\frac{\begin{array}{c}(F \text{ of } B)(fctid) = (strid : I, strexp', B') \\ B \vdash strexp \Rightarrow E \quad B' + \{strid \mapsto E \downarrow I\} \vdash strexp' \Rightarrow E'\end{array}}{B \vdash fctid\ (strexp) \Rightarrow E'} \tag{3.22}$$

값 정의

$\boxed{E \vdash valbind \Rightarrow VE}$

$$\frac{E \vdash e \Rightarrow v \quad E, v \vdash pat \Rightarrow VE \quad \langle E \vdash valbind \Rightarrow VE' \rangle}{E \vdash pat = e \ \langle \texttt{and}\ valbind \rangle \Rightarrow VE \langle + VE' \rangle} \tag{3.23}$$

$$\frac{E \vdash valbind \Rightarrow VE}{E \vdash \texttt{rec}\ valbind \Rightarrow \mathrm{rec}\, VE} \tag{3.24}$$

타입 정의

$\boxed{\vdash typbind \Rightarrow VE, TE}$

$$\frac{\langle \vdash typbind \Rightarrow VE, TE \rangle}{\vdash tyid = ty \ \langle \texttt{and}\ typbind \rangle \Rightarrow \{\} \ \langle + VE \rangle, \ \{tyid \mapsto \{\}\} \ \langle + TE \rangle} \tag{3.25}$$

$$\frac{\vdash conbind \Rightarrow VE \quad \langle \vdash typbind \Rightarrow VE', TE' \rangle}{\vdash tyid = conbind \ \langle \texttt{and}\ typbind \rangle \Rightarrow VE \ \langle + VE' \rangle, \{tyid \mapsto \mathrm{Dom}\ VE\} \ \langle + TE' \rangle} \tag{3.26}$$

데이타 구성자 정의

$\boxed{\vdash conbind \Rightarrow VE}$

$$\frac{\langle \vdash conbind \Rightarrow VE \rangle}{\vdash conid \ \langle \mid\ conbind \rangle \Rightarrow \{conid \mapsto conid\} \ \langle + VE \rangle} \tag{3.27}$$

예외상황 정의

$\boxed{\vdash exnbind \Rightarrow VE}$

$$\frac{\langle \vdash exnbind \Rightarrow VE \rangle}{\vdash conid \ \langle \texttt{and}\ exnbind \rangle \Rightarrow \{conid \mapsto conid\} \ \langle + VE \rangle} \tag{3.28}$$

모듈타입 정의

$\boxed{B \vdash sigbind \Rightarrow G}$

$$\frac{G \ \mathrm{of}\ B \vdash sig \Rightarrow I \quad \langle B \vdash sigbind \Rightarrow G \rangle}{B \vdash sigid = sig \ \langle \texttt{and}\ sigbind \rangle \Rightarrow \{sigid \mapsto I\} \ \langle + G \rangle} \tag{3.29}$$

모듈 정의

$\boxed{B \vdash strbind \Rightarrow SE}$

$$\frac{B \vdash strexp \Rightarrow E \quad \langle B \vdash strbind \Rightarrow SE \rangle}{B \vdash strid = strexp \ \langle \texttt{and}\ strbind \rangle \Rightarrow \{strid \mapsto E\} \ \langle + SE \rangle} \tag{3.30}$$

모듈 타입

$\boxed{G \vdash sig \Rightarrow I}$

$$\frac{G(sigid) = I}{G \vdash sigid \Rightarrow I} \tag{3.31}$$

$$\frac{\langle G \vdash spec \Rightarrow I \rangle}{G \vdash \texttt{sig}\ \langle spec \rangle\ \texttt{end} \Rightarrow \{\}\langle +I \rangle} \tag{3.32}$$

접속 방안

$\boxed{G \vdash spec \Rightarrow I}$

$$\frac{\vdash valdesc \Rightarrow VI}{G \vdash \texttt{val}\ valdesc \Rightarrow VI\ \text{in}\ Int} \tag{3.33}$$

$$\frac{\vdash typdesc \Rightarrow TI}{G \vdash \texttt{type}\ typdesc \Rightarrow TI\ \text{in}\ Int} \tag{3.34}$$

$$\frac{\vdash exndesc \Rightarrow VI}{G \vdash \texttt{exception}\ exndesc \Rightarrow VI\ \text{in}\ Int} \tag{3.35}$$

$$\frac{G \vdash sig \Rightarrow I}{G \vdash \texttt{include}\ sig \Rightarrow I} \tag{3.36}$$

$$\frac{G \vdash strdesc \Rightarrow SI}{G \vdash \texttt{structure}\ strdesc \Rightarrow SI\ \text{in}\ Int} \tag{3.37}$$

$$\frac{G \vdash spec_1 \Rightarrow I_1 \quad G \vdash spec_2 \Rightarrow I_2}{G \vdash spec_1\ \langle ; \rangle\ spec_2 \Rightarrow I_1 + I_2} \tag{3.38}$$

값 접속방안

$\boxed{\vdash valdesc \Rightarrow VI}$

$$\frac{\langle \vdash valdesc \Rightarrow VI \rangle}{\vdash varid\ :\ ty\ \langle \texttt{and}\ valdesc \rangle \Rightarrow \{varid\}\ \langle \cup VI \rangle} \tag{3.39}$$

$$\frac{\langle \vdash valdesc \Rightarrow VI \rangle}{\vdash \texttt{(}opid\texttt{)}\ :\ ty\ \langle \texttt{and}\ valdesc \rangle \Rightarrow \{opid\}\ \langle \cup VI \rangle} \tag{3.40}$$

타입 접속방안

$\boxed{\vdash typdesc \Rightarrow TI}$

$$\frac{\langle \vdash typdesc \Rightarrow TI \rangle}{\vdash tyvarseq\ tyid\ \langle \texttt{and}\ typdesc \rangle \Rightarrow \{tyid \mapsto \{\}\}\ \langle + TI \rangle} \tag{3.41}$$

$$\frac{\langle \vdash typdesc \Rightarrow TI \rangle}{\vdash tyvarseq\ tyid\ \texttt{=}\ ty\ \langle \texttt{and}\ typdesc \rangle \Rightarrow \{tyid \mapsto \{\}\}\ \langle + TI \rangle} \tag{3.42}$$

$$\frac{\vdash condesc \Rightarrow VI \quad \langle \vdash typdesc \Rightarrow TI \rangle}{\vdash tyvarseq\ tyid\ \texttt{=}\ condesc\ \langle \texttt{and}\ typdesc \rangle \Rightarrow \{tyid \mapsto VI\}\ \langle + TI \rangle} \tag{3.43}$$

데이타 구성자 접속방안

$\boxed{\vdash condesc \Rightarrow VI}$

$$\frac{\langle \vdash condesc \Rightarrow VI \rangle}{\vdash conid\ \langle \mid\ condesc \rangle \Rightarrow \{conid\}\ \langle \cup VI \rangle} \tag{3.44}$$

예외상황 접속방안

$\boxed{\vdash exndesc \Rightarrow VI}$

$$\frac{\langle \vdash exndesc \Rightarrow VI \rangle}{\vdash conid\ \langle \texttt{and}\ exndesc \rangle \Rightarrow \{conid\}\ \langle \cup VI \rangle} \tag{3.45}$$

모듈 접속방안

$\boxed{G \vdash strdesc \Rightarrow SI}$

$$\frac{G \vdash sig \Rightarrow I \quad \langle G \vdash strdesc \Rightarrow SI \rangle}{G \vdash strid\ :\ sig\ \langle \texttt{and}\ strdesc \rangle \Rightarrow \{strid \mapsto I\}\ \langle +SI \rangle} \tag{3.46}$$

프로그램 선언

$\boxed{B \vdash topdec \Rightarrow B'}$

$$\frac{B \vdash sigdec \Rightarrow G}{B \vdash sigdec \Rightarrow B + G} \tag{3.47}$$

$$\frac{B \vdash fntordec \Rightarrow F}{B \vdash fntordec \Rightarrow B + F} \tag{3.48}$$

$$\frac{B \vdash strdec \Rightarrow E}{B \vdash strdec \Rightarrow B + E} \tag{3.49}$$

$$\frac{B \vdash topdec_1 \Rightarrow B_1 \quad B + B_1 \vdash topdec_1 \Rightarrow B_2}{B \vdash topdec_1\ \langle ; \rangle\ topdec_2 \Rightarrow B_1 + B_2} \tag{3.50}$$

제 2 절 프로그램 식의 실행

프로그램 식

$\boxed{E \vdash e \Rightarrow v/[x]}$

$$\frac{}{E \vdash const \Rightarrow c} \tag{3.51}$$

$$\frac{E(varlongid) = v}{E \vdash varlongid \Rightarrow v} \tag{3.52}$$

$$\frac{E(conlongid) = v}{E \vdash conlongid \Rightarrow v} \tag{3.53}$$

$$\frac{E(oplongid) = v}{E \vdash \texttt{(}\ oplongid\ \texttt{)} \Rightarrow v} \tag{3.54}$$

$$\frac{}{E \vdash \texttt{\{\}} \Rightarrow \{\}\ \text{as}\ Record} \tag{3.55}$$

$$\frac{E \vdash labrow \Rightarrow r}{E \vdash \texttt{\{}\ labrow\ \texttt{\}} \Rightarrow r} \tag{3.56}$$

$$\frac{E \vdash e \Rightarrow \{lab \mapsto v, \cdots\}}{E \vdash e\ \texttt{.}\ lab \Rightarrow v} \tag{3.57}$$

$$\frac{}{E \vdash \texttt{[||]} \Rightarrow \{\} \text{ as } \mathit{Array}} \tag{3.58}$$

$$\frac{E \vdash \texttt{\{0 = ref } e_0\texttt{ , } \cdots \texttt{ , n = ref } e_n\texttt{\}} \Rightarrow r}{E \vdash \texttt{[|} e_0\texttt{ , } \cdots \texttt{ ,} e_n\texttt{|]} \Rightarrow r \text{ as } \mathit{Array}} \tag{3.59}$$

$$\frac{s, E \vdash e_1 \Rightarrow \{i \mapsto a, \cdots\}, s' \quad s', E \vdash e_2 \Rightarrow i, s''}{s, E \vdash e_1 \texttt{ . [} e_2 \texttt{]} \Rightarrow s''(a), s''} \tag{3.60}$$

$$\frac{s, E \vdash e_1 \Rightarrow \{i \mapsto a, \cdots\}, s' \quad s', E \vdash e_2 \Rightarrow i, s'' \quad s'', E \vdash e_3 \Rightarrow v, s'''}{s, E \vdash e_1 \texttt{ . [} e_2 \texttt{] <- } e_3 \Rightarrow \{\}, s''' + \{a \mapsto v\}} \tag{3.61}$$

$$\frac{E \vdash \mathit{valdec} \Rightarrow \mathit{VE} \quad E + \mathit{VE} \vdash e \Rightarrow v}{E \vdash \texttt{let } \mathit{valdec} \texttt{ in } e \texttt{ end} \Rightarrow v} \tag{3.62}$$

$$\frac{E \vdash e \Rightarrow v}{E \vdash \texttt{(} e \texttt{)} \Rightarrow v} \tag{3.63}$$

$$\frac{E \vdash e_1 \Rightarrow (\mathit{match}, E', \mathit{VE}) \quad E \vdash e_2 \Rightarrow v' \quad E' + \operatorname{rec} \mathit{VE}, v' \vdash \mathit{match} \Rightarrow v/\text{fail}}{E \vdash e_1\ e_2 \Rightarrow v/[\texttt{Match}]} \tag{3.64}$$

$$\frac{E \vdash e_1 \Rightarrow \mathit{conid} \quad E \vdash e_2 \Rightarrow v}{E \vdash e_1\ e_2 \Rightarrow (\mathit{conid}, v)} \tag{3.65}$$

$$\frac{E \vdash e_1 \Rightarrow \mathit{primop} \quad E \vdash e_2 \Rightarrow v \quad \mathit{primop}\ v \Rightarrow v'}{E \vdash e_1\ e_2 \Rightarrow v'} \tag{3.66}$$

$$\frac{}{E \vdash \texttt{fn } \mathit{match} \Rightarrow (\mathit{match}, E, \{\})} \tag{3.67}$$

$$\frac{E \vdash e \Rightarrow v}{E \vdash e \texttt{ handle } \mathit{match} \Rightarrow v} \tag{3.68}$$

$$\frac{E \vdash e \Rightarrow [x] \quad E, x \vdash \mathit{match} \Rightarrow v/\text{fail}}{E \vdash e \texttt{ handle } \mathit{match} \Rightarrow v/[x]} \tag{3.69}$$

$$\frac{E \vdash e \Rightarrow x}{E \vdash \texttt{raise } e \Rightarrow [x]} \tag{3.70}$$

$$\frac{s, E \vdash e \Rightarrow v, s' \quad a \notin \operatorname{Dom} s'}{s, E \vdash \texttt{ref } e \Rightarrow a, s' + \{a \mapsto v\}} \tag{3.71}$$

$$\frac{s, E \vdash e_1 \Rightarrow a, s' \quad s', E \vdash e_2 \Rightarrow v, s''}{s, E \vdash e_1 \texttt{ := } e_2 \Rightarrow \{\}, s'' + \{a \mapsto v\}} \tag{3.72}$$

$$\frac{s, E \vdash e_1 \Rightarrow a, s'}{s, E \vdash \texttt{! } e \Rightarrow s'(a), s'} \tag{3.73}$$

$$\frac{E \vdash e_1 \Rightarrow v_1 \quad E \vdash e_2 \Rightarrow v_2}{E \vdash e_1 \texttt{ ; } e_2 \Rightarrow v_2} \tag{3.74}$$

$$\frac{E \vdash e \Rightarrow v' \quad E, v' \vdash \mathit{match} \Rightarrow v/\text{fail}}{E \vdash \texttt{case } e \texttt{ of } \mathit{match} \Rightarrow v/[\texttt{Match}]} \tag{3.75}$$

레코드 내용

$\boxed{E \vdash labrow \Rightarrow r}$

$$\frac{E \vdash e \Rightarrow v \quad \langle E \vdash labrow \Rightarrow r \rangle}{E \vdash lab \texttt{ = } e \langle \texttt{, } labrow \rangle \Rightarrow \{lab \mapsto v\} \langle + r \rangle} \tag{3.76}$$

패턴 매치

$\boxed{E, v \vdash match \Rightarrow v'/\text{fail}}$

$$\frac{E, v \vdash pat \Rightarrow VE \quad E + VE \vdash e \Rightarrow v'}{E, v \vdash pat \texttt{ => } e \Rightarrow v'} \tag{3.77}$$

$$\frac{E, v \vdash pat \Rightarrow \text{fail}}{E, v \vdash pat \texttt{ => } e \Rightarrow \text{fail}} \tag{3.78}$$

$$\frac{E, v \vdash pat \Rightarrow VE \quad E + VE \vdash e \Rightarrow v'}{E, v \vdash pat \texttt{ => } e \texttt{ | } match \Rightarrow v'} \tag{3.79}$$

$$\frac{E, v \vdash pat \Rightarrow \text{fail} \quad E, v \vdash match \Rightarrow v'}{E, v \vdash pat \texttt{ => } e \texttt{ | } match \Rightarrow v'} \tag{3.80}$$

패턴

$\boxed{E, v \vdash pat \Rightarrow VE/\text{fail}}$

$$\frac{}{E, v \vdash \texttt{\_} \Rightarrow \{\}} \tag{3.81}$$

$$\frac{c = v}{E, v \vdash c \Rightarrow \{\}} \tag{3.82}$$

$$\frac{c \neq v}{E, v \vdash c \Rightarrow \text{fail}} \tag{3.83}$$

$$\frac{}{E, \{\} \text{ as } Record \vdash \texttt{\{\}} \Rightarrow \{\}} \tag{3.84}$$

$$\frac{E, r \vdash patrow \Rightarrow VE/\text{fail}}{E, r \vdash \texttt{\{ } patrow \texttt{ \}} \Rightarrow VE/\text{fail}} \tag{3.85}$$

$$\frac{}{E, \{\} \text{ as } Array \vdash \texttt{[||]} \Rightarrow \{\}} \tag{3.86}$$

$$\frac{\begin{array}{c} r = y \text{ as } Record \\ E, r \vdash \{ \texttt{0 = ref } pat_0 \texttt{, } \cdots \texttt{, n = ref } pat_n \} \Rightarrow VE/\text{fail} \end{array}}{E, y \vdash \texttt{[|} pat_0 \texttt{ , } \cdots \texttt{ , } pat_n \texttt{|]} \Rightarrow VE/\text{fail}} \tag{3.87}$$

$$\frac{E, v \vdash pat \Rightarrow VE/\text{fail}}{E, v \vdash \texttt{(} pat \texttt{)} \Rightarrow VE/\text{fail}} \tag{3.88}$$

$$\frac{s(a) = v \quad E, v \vdash pat \Rightarrow VE/\text{fail}}{s, E, a \vdash \texttt{ref } pat \Rightarrow VE/\text{fail}} \tag{3.89}$$

$$\frac{E(conlongid) = v}{E, v \vdash conlongid \Rightarrow \{\}} \tag{3.90}$$

$$\frac{E(conlongid) = v' \quad v \neq v'}{E, v \vdash conlongid \Rightarrow \text{fail}} \tag{3.91}$$

$$\frac{}{E, v \vdash varid \Rightarrow \{varid \mapsto v\}} \tag{3.92}$$

$$\frac{E(conlongid) = conid \quad v = (conid, v') \quad E, v' \vdash pat \Rightarrow VE/\text{fail}}{E, v \vdash conlongid\ pat \Rightarrow VE/\text{fail}} \tag{3.93}$$

$$\frac{E(conlongid) = conid \quad v \neq (conid, v')}{E, v \vdash conlongid\ pat \Rightarrow \text{fail}} \tag{3.94}$$

$$\frac{E, v \vdash pat \Rightarrow VE/\text{fail}}{E, v \vdash varid\ \texttt{as}\ pat \Rightarrow VE + \{varid \mapsto v\}/\text{fail}} \tag{3.95}$$

$$\frac{E, v \vdash pat \Rightarrow VE/\text{fail}}{E, v \vdash \texttt{(}opid\texttt{)}\ \texttt{as}\ pat \Rightarrow VE + \{opid \mapsto v\}/\text{fail}} \tag{3.96}$$

$$\frac{E, v \vdash pat_1 \Rightarrow VE}{E, v \vdash pat_1 \mid pat_2 \Rightarrow VE} \tag{3.97}$$

$$\frac{E, v \vdash pat_1 \Rightarrow \text{fail} \quad E, v \vdash pat_2 \Rightarrow VE/\text{fail}}{E, v \vdash pat_1 \mid pat_2 \Rightarrow VE/\text{fail}} \tag{3.98}$$

레코드 패턴

$\boxed{E, r \vdash patrow \Rightarrow VE/\text{fail}}$

$$\frac{}{E, r \vdash \ldots \Rightarrow \{\}} \tag{3.99}$$

$$\frac{E, r(lab) \vdash pat \Rightarrow VE_1/\text{fail} \quad \langle E, r \vdash patrow \Rightarrow VE_2/\text{fail}\rangle}{E, r \vdash lab\ \texttt{=}\ pat\ \langle\texttt{,}\ patrow\rangle \Rightarrow VE_1\ \langle + VE_2\rangle/\text{fail}} \tag{3.100}$$

기본 연산자

$\boxed{primop\ v \Rightarrow v'}$

$$\frac{}{\texttt{=}(c, c) \Rightarrow true} \tag{3.101}$$

$$\frac{}{\texttt{=}(a, a) \Rightarrow true} \tag{3.102}$$

$$\frac{}{\texttt{=}(primop, primop) \Rightarrow true} \tag{3.103}$$

$$\frac{}{\texttt{=}(conid, conid) \Rightarrow true} \tag{3.104}$$

$$\frac{\texttt{=}(v_1, v_2) \Rightarrow true}{\texttt{=}((conid, v_1), (conid, v_2)) \Rightarrow true} \tag{3.105}$$

$$\frac{\text{Dom}\, r_1 = \text{Dom}\, r_2 \quad \forall lab \in \text{Dom}\, r_1.\texttt{=}(r_1(lab), r_2(lab)) \Rightarrow true}{\texttt{=}(r_1, r_2) \Rightarrow true} \tag{3.106}$$

$$\frac{\mathrm{Dom}\, y_1 = \mathrm{Dom}\, y_2 \quad \forall i \in \mathrm{Dom}\, y_1 . \mathsf{=}(y_1(i), y_2(i)) \Rightarrow \mathit{true}}{\mathsf{=}(y_1, y_2) \Rightarrow \mathit{true}} \tag{3.107}$$

$$\frac{\neg(\mathsf{=}(v_1, v_2) \Rightarrow \mathit{true})}{\mathsf{=}(v_1, v_2) \Rightarrow \mathit{false}} \tag{3.108}$$

$$\frac{\mathsf{=}(v_1, v_2) \Rightarrow c}{\mathsf{<>}(v_1, v_2) \Rightarrow \neg c} \tag{3.109}$$

제 4 장

프로그램의 기획

$$
\begin{array}{rcrcl}
\tau & \in & \mathit{Type} & = & \mathit{TyVar} \cup \mathit{RowType} \cup \mathit{FunType} \cup \mathit{DataType} \\
(\tau_1, \cdots, \tau_k) \text{ or } \tau^{(k)} & \in & \mathit{Type}^k & & \\
(\alpha_1, \cdots, \alpha_k) \text{ or } \alpha^{(k)} & \in & \mathit{TyVar}^k & & \\
\varrho & \in & \mathit{RowType} & = & \mathit{Lab} \xrightarrow{\text{fin}} \mathit{Type} \\
\tau \to \tau' & \in & \mathit{FunType} & = & \mathit{Type} \times \mathit{Type} \\
 & & \mathit{DataType} & = & \cup_{k \geq 0} \mathit{DataType}^{(k)} \\
\tau^{(k)} t & \in & \mathit{DataType}^{(k)} & = & \mathit{Type}^k \times \mathit{TyName}^{(k)} \\
\theta \text{ or } \Lambda\alpha.\tau & \in & \mathit{TypeFn} & = & \cup_{k \geq 0} \mathit{TyVar}^k \times \mathit{Type} \\
\sigma \text{ or } \forall\alpha.\tau & \in & \mathit{TypeScheme} & = & \cup_{k \geq 0} \mathit{TyVar}^k \times \mathit{Type} \\
\mathit{TS} & \in & \mathit{TyStr} & = & \mathit{TypeFn} \times \mathit{ValEnv} \\
\mathit{SE} & \in & \mathit{StrEnv} & = & \mathit{StrId} \xrightarrow{\text{fin}} \mathit{Env} \\
\mathit{TE} & \in & \mathit{TyEnv} & = & \mathit{TyId} \xrightarrow{\text{fin}} \mathit{TyStr} \\
\mathit{VE} & \in & \mathit{ValEnv} & = & \mathit{VId} \xrightarrow{\text{fin}} \mathit{TypeScheme} \\
E \text{ or } (\mathit{SE}, \mathit{TE}, \mathit{VE}) & \in & \mathit{Env} & = & \mathit{StrEnv} \times \mathit{TyEnv} \times \mathit{ValEnv} \\
T & \in & \mathit{TyNameSet} & = & \mathrm{Fin}(\mathit{TyName}) \\
U & \in & \mathit{TyVarSet} & = & \mathrm{Fin}(\mathit{TyVar}) \\
C \text{ or } T, U, E & \in & \mathit{Context} & = & \mathit{TyNameSet} \times \mathit{TyVarSet} \times \mathit{Env} \\
 & & \mathit{VId} & = & \mathit{VarId} \cup \mathit{OpId} \cup \mathit{ConId} \\
 & & & & \\
\Sigma \text{ or } (T)E & \in & \mathit{Sig} & = & \mathit{TyNameSet} \times \mathit{Env} \\
\Phi \text{ or } (T)(E, (T')E') & \in & \mathit{FunSig} & = & \mathit{TyNameSet} \times (\mathit{Env} \times \mathit{Sig}) \\
G & \in & \mathit{SigEnv} & = & \mathit{SigId} \xrightarrow{\text{fin}} \mathit{Sig} \\
F & \in & \mathit{FunEnv} & = & \mathit{FunId} \xrightarrow{\text{fin}} \mathit{FunSig} \\
B \text{ or } T, F, G, E & \in & \mathit{Basis} & = & \mathit{TyNameSet} \times \mathit{FunEnv} \times \mathit{SigEnv} \times \mathit{Env}
\end{array}
$$

$X^{(k)}$: k개의 인자를 받아들이는 물건들.

X^k: k개의 X 원소들로 구성된 집합.

Type name binders: 타입이름을 묶어주는 것은 오직 두가지 뿐. 모듈 타입 $(T)E \in \mathit{Sig}$에서의 (T)와 모듈함수 타입 $(T)(E, (T')E') \in \mathit{FunSig}$에서의 (T).

tynames A: A에서 묶여있지 않은 타입 이름들.

Type variable binders: 타입변수 α_i를 묶어주는 것은 오직 두가지 뿐. 타입함수 $\Lambda\alpha^{(k)}.\tau$와 타입 구도 $\forall\alpha^{(k)}.\tau$.

tyvars A: A에서 묶여있지 않은 타입 변수들.

tyvars ε: 프로그램 텍스트 ε에서 묶여있지 않은 타입 변수들. 프로그램 텍스트에서 타입 변수를 묶어주는 경우는 세가지: 타입정의(*tyvarseq id* = *ty*), 데이타 타입 정의(*tyvarseq id* = *conbind*), 그리고 값 선언(`val` *tyvarseq valbind*). 타입 변수가 프로그램 텍스트에 드러나는 경우는 위의 세 경우와 다음의 셋: 프로그램 식 ("e : ty"), 패턴 ("p : ty"), 그리고 예외상황 선언 ("`of` ty").

Well-formed A: A가 제대로 구성되었다는 것은 타입 구조의 내용 (θ, VE) 에서 타입함수 θ가 이름으로 달랑있거나 (이 경우는 테이타 타입인 경우로서 VE가 데이타 구성자들의 타입을 표현하고 있다), 아니면 VE가 공집합인 경우이다 (이 경우는 데이타 타입이 아닌 타입 정의).

Type realization $\varphi\, t$: 타입이름 t를 실현한다는 것은 그 이름을 구체적인 타입함수로 치환한다는 것을 의미. 이때, 타입 이름이 동치 확인 가능성을 가지고 있다면 치환한 결과도 그래야 한다.

Supp φ: 치환함수에서 실제로 치환되는 이름들의 집합.

Yield φ: 치환 결과의 타입함수들에서 묶여있지 않는 타입 이름들의 집합.

$\varphi\, A$: 타입이름을 실현하는 것이 일반적인 대상 A에 적용될 때는, 그안에 있는 타입이름을 골라서 그 치환함수 φ를 적용한다. 이 때, 묶여있는 타입 이름들은 건들여지지 않도록 한다. 예를 들어, 모듈 타입 $(T)E$에 φ가 적용되면, E에 있는 타입이름들을 실현하게 되는데, 이 때 T에 묶여 있는 이름을 치환하거나 치환 결과에서 자유로왔던 타입이름들(Yield φ)이 T에 의해 묶이지 않도록 T를 미리 적절히 바꾸어 놓는 것을 전제로한다.

$\oplus$: 묶이지 않은 타입이름들을 T로 모아가면서 융합시키는 것.
$C \oplus TE \stackrel{\text{def}}{=} C + (\text{tynames}\, TE, TE)$. $C \oplus E \stackrel{\text{def}}{=} C + (\text{tynames}\, E, E)$. $B \oplus SE \stackrel{\text{def}}{=} B + (\text{tynames}\, SE, SE)$.

Type function θ: 타입함수 $\theta = \Lambda\alpha^{(k)}.\tau$는 k개의 인자를 받는다. 묶여있는 타입변수들은 모두 다르다.

Type name as type function: 타입 이름 t를 일반적으로 타입함수 $\Lambda\alpha^{(k)}.t$로 간주한다. 이 때 k는 t라는 타입이 받아들이는 인자의 갯수.

Type function application: 타입 함수 θ가 타입들 $\tau^{(k)}$에 적용되는 것을 $\tau^{(k)}\theta$로 표현한다. 즉, θ가 $\Lambda\alpha^{(k)}.\tau$라면 τ에 있는 α_i를 τ_i로 치환하는 것이다: $\tau^{(k)}\theta \stackrel{\text{def}}{=} \{\alpha^{(k)} \mapsto \tau^{(k)}\}\tau$.

$\{t^{(k)} \mapsto \theta^{(k)}\}\tau$: 타입 τ에 있는 타입 이름 t_i를 타입함수 θ_i로 치환하는 경우는, 치환되고 나서 타입함수들이 적용되는 경우가 발생하면 그런것들을 모두 적용한 결과까지를 간단히 $\{t^{(k)} \mapsto \theta^{(k)}\}\tau$로 표현한다.

Type as type scheme: 타입 τ도 타입구도 $\forall().\tau$로 간주한다.

Syntactic context: 프로그램 텍스트에서 ε의 주변까지를 표현할 때 $\mathcal{E}[\varepsilon]$로 표현한다. 텍스트 ε의 주변이 $\mathcal{E}[]$인 것이다.

Non-expansive expr: C라는 환경에서 메모리 반응을 일으키지 않을 것이 확실한 프로그램 식은 다음의 문법(ne)에 의해 정의되는 것들이다.

$$
\begin{array}{rcl}
ne & ::= & c \\
& | & varlongid \\
& | & conlongid \\
& | & (oplongid) \\
& | & \{nerow\} \\
& | & (ne) \\
& | & cone\ ne \\
& | & ne : ty \\
& | & \texttt{fn}\ match
\end{array}
\qquad
\begin{array}{rcl}
nerow & ::= & lab = ne\langle, nerow\rangle \\
cone & ::= & (cone\langle : ty\rangle) \\
& | & conlongid
\end{array}
$$

Clos: 타입 구도를 잡아준다.

$\text{Clos}_A \tau \stackrel{\text{def}}{=} \forall\alpha^{(k)}.\tau$ where $\alpha^{(k)} = \text{tyvars}\,(\tau) \setminus \text{tyvars}\,(A)$.

$\text{Clos}_A\, VE \stackrel{\text{def}}{=} \{vid \mapsto \text{Clos}_A \tau \mid VE(vid) = \tau\}$.

$\text{Clos}\, VE \stackrel{\text{def}}{=} \text{Clos}_\emptyset\, VE$.

$\text{Clos}_{C,valbind}\, VE(vid) \stackrel{\text{def}}{=} \forall\alpha^{(k)}.\tau$

where $VE(vid) = \tau$, $valbind = pat$ = e, and

$$\alpha^{(k)} = \begin{cases} () & \text{if } e \text{ is expansive in } C \\ \text{tyvars}\,\tau \setminus \text{tyvars}\,C & \text{if } e \text{ is non-expansive in } C \end{cases}$$

타입구도의 적응 $\boxed{\sigma_1 \succ \sigma_2}$

$$\frac{\sigma_1 \succ \tau_2 \quad \beta^{(l)} \cap \text{tyvars}\,\sigma_1 = \emptyset}{\sigma_1 \stackrel{\text{let}}{=} \forall\alpha^{(k)}.\tau_1 \succ \forall\beta^{(l)}.\tau_2 \stackrel{\text{let}}{=} \sigma_2} \tag{4.1}$$

$$\frac{\tau_1 = \{\alpha^{(k)} \mapsto \tau^{(k)}\}\tau}{\forall\alpha^{(k)}.\tau \succ \tau_1} \tag{4.2}$$

(4.1) 타입구도 σ_1이 좀더 구체적인 σ_2로 변화될 때 σ_1에서 자유로운 타입 변수들 $\text{tyvars}\,\sigma_1$이 σ_2에서 묶여지지는 말아야 한다.

같은 타입함수들 $\boxed{\theta_1 = \theta_2}$

$$\frac{\text{permutation } \pi : [1, \cdots, k] \quad \{\alpha_i \mapsto \beta_{\pi(i)}\}\tau_1 = \tau_2}{\Lambda\alpha^{(k)}.\tau_1 = \Lambda\beta^{(k)}.\tau_2} \tag{4.3}$$

$$\frac{\alpha'^{(k)} = tyvar\,\tau_1 \cap \alpha^{(k_1)} \quad \beta'^{(k)} = tyvar\,\tau_2 \cap \beta^{(k_2)} \quad \Lambda\alpha'^{(k)}.\tau_1 = \Lambda\beta'^{(k)}.\tau_2}{\Lambda\alpha^{(k_1)}.\tau_1 = \Lambda\beta^{(k_2)}.\tau_2} \tag{4.4}$$

(4.3, 4.4) 실질적으로 묶여있는(4.4) 타입 변수들을 다른 이름으로 바꾸어서 같아지면 된다.

같은 타입구도들 $\boxed{\sigma_1 = \sigma_2}$

$$\frac{\text{permutation } \pi : [1, \cdots, k] \quad \{\alpha_i \mapsto \beta_{\pi(i)}\}\tau_1 = \tau_2}{\forall\alpha^{(k)}.\tau_1 = \forall\beta^{(k)}.\tau_2} \tag{4.5}$$

$$\frac{\alpha'^{(k)} = tyvar\,\tau_1 \cap \alpha^{(k_1)} \quad \beta'^{(k)} = tyvar\,\tau_2 \cap \beta^{(k_2)} \quad \forall\alpha'^{(k)}.\tau_1 = \forall\beta'^{(k)}.\tau_2}{\forall\alpha^{(k_1)}.\tau_1 = \forall\beta^{(k_2)}.\tau_2} \tag{4.6}$$

(4.5, 4.6) 실질적으로 묶여있는(4.6) 타입 변수들을 다른 이름으로 바꾸어서 같아지면 된다.

프로그램에 드러난 타입변수의 유효범위 $\boxed{\texttt{'a scopes } \varepsilon}$

$$\frac{typbind \stackrel{\text{let}}{=} tyvarseq\ id\ \texttt{=}\ ty \quad \texttt{'a} \in ty \quad \texttt{'a} \in tyvarseq}{\texttt{'a scopes } typbind} \tag{4.7}$$

$$\frac{typbind \stackrel{\text{let}}{=} tyvarseq\ id\ \texttt{=}\ conbind \quad \texttt{'a} \in conbind \quad \texttt{'a} \in tyvarseq}{\texttt{'a}\ \text{scopes}\ typbind} \tag{4.8}$$

$$\frac{\begin{array}{c} valdec \stackrel{\text{let}}{=} \texttt{val}\ tyvarseq\ valbind \\ \mathcal{E}[valdec] \quad \texttt{'a} \in \text{tyvars}\ valbind \quad \text{tightest}(\mathcal{E}[], valbind, \texttt{'a}) \end{array}}{\texttt{'a}\ \text{scopes}\ valdec} \tag{4.9}$$

$$\frac{\begin{array}{c} \texttt{'a} \in \text{tyvars}\ Inside \quad \texttt{'a} \notin \text{tyvars}\ Outside \\ \neg\text{tightest}(Outside[I[]], Inside', \texttt{'a}) \quad I \neq [] \quad Inside = I[Inside'] \end{array}}{\text{tightest}(Outside[], Inside, \texttt{'a})} \tag{4.10}$$

(4.10) 프로그램에 있는 타입변수의 유효범위는 그 변수를 자유변수로 품고있는 모두를 포함하는 최소한의(tighest) 범위로 한다. 즉, 이 범위 바깥에서는 그 변수가 자유변수로 있지 않아야하면서, 이 범위 안쪽으로 좁혀들어가면 그 변수를 자유변수로 품고있는 모두를 포함할 수 없게되는 범위.

모듈 타입의 실현 $\boxed{E : \Sigma}$

$$\frac{\Sigma = (T)E_1 \quad \varphi(E_1) = E_2 \quad \text{Supp}\,\varphi \subseteq T}{E_2 : \Sigma} \tag{4.11}$$

(4.11) 기획한 모듈 E_2이 타입 Σ를 실현한 것이 되려면, E_1에서 T에 속한 타입 이름들을 실현한 것이 E_2와 일치해야 한다. "Supp $\varphi \subseteq T$"이 "Supp $\varphi = T$"가 아닌 이유는, 타입이름을 실현한 것이 같은 타입이름일 수 있기 때문이다:

```
type t = int
structure T = struct type t = t end: sig type t end
```

모듈함수 타입의 실현 $\boxed{(E_2, (T_2')E_2') : \Phi}$

$$\frac{\Phi = (T_1)(E_1, (T_1')E_1') \quad \varphi(E_1, (T_1')E_1') = (E_2, (T_2')E_2') \quad \text{Supp}\,\varphi \subseteq T_1}{(E_2, (T_2')E_2') : \Phi} \tag{4.12}$$

(4.12) 기획한 모듈 함수 $(E_2, (T_2')E_2')$가 타입 Φ를 실현한 것이 되려면, 그 모듈 함수 타입의 알맹이 $(E_1, (T_1')E_1')$에서 T_1에 속한 타입 이름들을 실현한 것이 $(E_2, (T_2')E_2')$와 일치해야 한다.

기획 환경의 적응 $\boxed{E_1 \succ E_2}$

$$\frac{SE_1 \succ SE_2 \quad TE_1 \succ TE_2 \quad VE_1 \succ VE_2}{(SE_1, TE_1, VE_1) \succ (SE_2, TE_2, VE_2)} \tag{4.13}$$

$$\frac{\text{Dom}\ SE_1 \supseteq \text{Dom}\ SE_2 \quad \forall x \in \text{Dom}\ SE_2,\ SE_1(x) \succ SE_2(x)}{SE_1 \succ SE_2} \tag{4.14}$$

$$\frac{\text{Dom } TE_1 \supseteq \text{Dom } TE_2 \quad \forall x \in \text{Dom } TE_2,\ TE_1(x) \succ TE_2(x)}{TE_1 \succ TE_2} \tag{4.15}$$

$$\frac{\text{Dom } VE_1 \supseteq \text{Dom } VE_2 \quad \forall x \in \text{Dom } VE_2,\ VE_1(x) \succ VE_2(x)}{VE_1 \succ VE_2} \tag{4.16}$$

타입구조의 적응 $\boxed{TS_1 \succ TS_2}$

$$\frac{\theta_1 = \theta_2 \quad (VE_1 = VE_2 \text{ or } VE_2 = \emptyset)}{(\theta_1, VE_1) \succ (\theta_2, VE_2)} \tag{4.17}$$

(4.17) 타입구조는 데이타 타입의 내용을 담는데, 이것이 적응하는 경우는 데이타 구성자들의 타입내용(VE_i)이 일치하거나 완전히 감추어지는 경우에만이다. 완전히 감추어지는 경우는 모듈타입에서 "`type t`"가 모듈에서는 데이타 타입으로 실현된 경우이다.

제 1 절 프로그램 모듈의 기획

선언 $\boxed{C \vdash dec \Rightarrow E}$

$$\frac{C \vdash valdec \Rightarrow VE}{C \vdash valdec \Rightarrow VE \text{ in } Env} \tag{4.18}$$

$$\frac{C \oplus TE \vdash typbind \Rightarrow VE, TE, T' \quad T' \cap (T \text{ of } C) = \emptyset}{C \vdash \texttt{type}\ typbind \Rightarrow (VE, TE) \text{ in } Env} \tag{4.19}$$

$$\frac{C \vdash exnbind \Rightarrow VE}{C \vdash \texttt{exception}\ exnbind \Rightarrow VE \text{ in } Env} \tag{4.20}$$

$$\frac{C \vdash dec_1 \Rightarrow E_1 \quad E \oplus E_1 \vdash dec_2 \Rightarrow E_2}{C \vdash \texttt{local}\ dec_1\ \texttt{in}\ dec_2\ \texttt{end} \Rightarrow E_2} \tag{4.21}$$

$$\frac{C \vdash dec_1 \Rightarrow E_1 \quad C \oplus E_1 \vdash dec_2 \Rightarrow E_2}{C \vdash dec_1\ \langle ; \rangle\ dec_2 \Rightarrow E_1 + E_2} \tag{4.22}$$

(4.19) 데이타 타입 선언은 새로운 타입을 생성한다 (새로운 타입 이름 t). 재귀적인 데이타 타입 선언이 가능하도록 TE가 $typbind$의 왼쪽에도 있게 된다.

값 선언 $\boxed{C \vdash valdec \Rightarrow VE}$

$$\frac{\begin{array}{c} U = \text{tyvars } tyvarseq \quad C + U \vdash valbind \Rightarrow VE \\ VE' = \text{Clos}_{C,valbind}\, VE \quad U \cap \text{tyvars } VE' = \emptyset \end{array}}{C \vdash \texttt{val}\ tyvarseq\ valbind \Rightarrow VE} \tag{4.23}$$

$$\frac{C \vdash valdec_1 \Rightarrow VE_1 \quad C \oplus VE_1 \vdash valdec_2 \Rightarrow VE_2}{C \vdash valdec_1\ \langle ; \rangle\ valdec_2 \Rightarrow VE_1 + VE_2} \tag{4.24}$$

(4.23) 텍스트에 드러난 타입변수들(U)은, 선언된 값의 타입 구도가 잡히는(Clos) 지금에야 묶여서 일반화 된다($U \cap \text{tyvars}\, VE' = \emptyset$). 그 타입변수들이 *valbind* 안에서 미리 묶이는 것을 막기위해 $C + U$가 *valbind*의 배경이 된다. 기억하라, Clos의 정의에 의해서, 묶여서 일반화되는 타입 변수들 중에는 $C + U$에 있는 것들이 제외된다.

모듈 선언

$\boxed{B \vdash \mathit{strdec} \Rightarrow E}$

$$\frac{(T \text{ of } B, \{\}, E \text{ of } B) \stackrel{\text{let}}{=} C \vdash \mathit{dec} \Rightarrow E'}{B \vdash \mathit{dec} \Rightarrow E'} \tag{4.25}$$

$$\frac{B \vdash \mathit{strbind} \Rightarrow \mathit{SE}}{B \vdash \texttt{structure}\ \mathit{strbind} \Rightarrow \mathit{SE} \text{ in } \mathit{Env}} \tag{4.26}$$

$$\frac{B \vdash \mathit{strdec}_1 \Rightarrow E_1 \quad B \oplus E_1 \vdash \mathit{strdec}_2 \Rightarrow E_2}{B \vdash \mathit{strdec}_1\ \langle ; \rangle\ \mathit{strdec}_2 \Rightarrow E_1 + E_2} \tag{4.27}$$

모듈함수 선언

$\boxed{B \vdash \mathit{fntordec} \Rightarrow F}$

$$\frac{\begin{array}{c} B \vdash \mathit{sig}_1 \Rightarrow (T)E \quad B \oplus \{\mathit{strid} \mapsto E\} \vdash \mathit{strexp} \Rightarrow E' \\ T \cap (T \text{ of } B) = \emptyset \quad T' = \text{tynames}\, E' \setminus ((T \text{ of } B) \cup T) \end{array}}{B \vdash \texttt{functor}\ \mathit{fctid}\ \texttt{(}\ \mathit{strid}\ \texttt{:}\ \mathit{sig}_1\ \texttt{)}\ \texttt{=}\ \mathit{strexp} \Rightarrow \{\mathit{fctid} \mapsto (T)(E, (T')E')\}} \tag{4.28}$$

(4.28) 모듈 함수가 선언되면 모듈함수의 타입이 잡힌다. T는 인자 모듈에 의해서 실현될 타입들이다. 이 이름들은 현재 (모듈 함수가 선언될 때) 알고 있는 타입이름들(T of B)과 겹치지 말아야 한다. T'은 모듈 함수가 적용이 될 때 실현될 타입 이름들이다. 이 것은 만들 모듈 E'에 있는 타입 이름중에서 현재 알고 있는 타입이름들(T of B)과 인자 모듈에서 결정되는 타입이름들(T)을 제외한 것이다.

모듈타입 선언

$\boxed{B \vdash \mathit{sigdec} \Rightarrow G}$

$$\frac{B \vdash \mathit{sigbind} \Rightarrow G}{B \vdash \texttt{signature}\ \mathit{sigbind} \Rightarrow G} \tag{4.29}$$

모듈 식

$\boxed{B \vdash \mathit{strexp} \Rightarrow E}$

$$\frac{B(\mathit{strlongid}) = E}{B \vdash \mathit{strlongid} \Rightarrow E} \tag{4.30}$$

$$\frac{\langle B \vdash \mathit{strdec} \Rightarrow E \rangle}{B \vdash \texttt{struct}\ \langle \mathit{strdec} \rangle\ \texttt{end} \Rightarrow \{\}\langle + E \rangle} \tag{4.31}$$

$$\frac{B \vdash strexp \Rightarrow E \quad B \vdash sig \Rightarrow (T')E' \quad E \succ E'' : (T')E'}{B \vdash strexp : sig \Rightarrow E'} \tag{4.32}$$

$$\frac{\begin{array}{c} B \vdash strexp \Rightarrow E \quad E \succ E'' \\ (E'', (T')E') : (F \text{ of } B)(fctid) \quad T' \cap (\text{tynames } E \cup (T \text{ of } B)) = \emptyset \end{array}}{B \vdash fctid\ (strexp) \Rightarrow E'} \tag{4.33}$$

(4.32) 모듈이 주어진 모듈 타입($(T')E'$)에 맞추어지는 지 확인하고 모듈 타입에서 지정된 내용만큼을(E') 보여준다. 모듈 E가 타입 $(T')E'$에 맞추어 지는지는, E가 적당히 적응($E \succ E''$)해서 $(T')E'$를 실현한 모습($E'' : (T')E'$)이 되는 지를 보는 것이다. 확인 후, 모듈 타입에서 표현하는 것 이외의 내용은 새나가지 않는다 (opaque signature matching). 예를들어:

```
structure S = struct type t = int val x = 1 end :
                sig type t val x:t end
S.x + 1  (* type error; no type information of S.x is known. *)
```

(4.33) 모듈함수가 적용되서 새로운 모듈이 만들어진다. 모듈함수의 타입을 실현하는 것들중에 인자 모듈이 적응된($E \succ E''$) 것을 취하면 된다. 이때, 만들어 지는 모듈 E'에서 실현되는 타입들(T')은 새로운 것이어야 한다 ($T' \cap (\text{tynames } E \cup (T \text{ of } B)) = \emptyset$).

값 정의

$\boxed{C \vdash valbind \Rightarrow VE}$

$$\frac{C \vdash pat \Rightarrow (VE, \tau) \quad C \vdash e \Rightarrow \tau \quad \langle C \vdash valbind \Rightarrow VE' \rangle}{C \vdash pat = e\ \langle \texttt{and}\ valbind \rangle \Rightarrow VE \langle + VE' \rangle} \tag{4.34}$$

$$\frac{C + VE \vdash valbind \Rightarrow VE \quad \text{tynames } VE \subseteq T \text{ of } C}{C \vdash \texttt{rec}\ valbind \Rightarrow VE} \tag{4.35}$$

(4.34) 패턴 pat에서 결정된 타입과 프로그램 식 e에서 결정된 타입이 같아야 한다.

(4.35) 재귀적인 $valbind$가 자기자신을 물면서 정의되기 때문에 VE가 왼편에도 있게 된다. VE안에서 알려지지 않은 외부의 타입이름들은 현재의 배경에서 알려진 것들 이어야 한다(tynames $VE \subseteq T$ of C).

타입 정의

$\boxed{C \vdash typbind \Rightarrow VE, TE, T}$

$$\frac{tyvarseq = \alpha^{(k)} \quad C \vdash ty \Rightarrow \tau \quad \langle C \vdash typbind \Rightarrow VE, TE, T \rangle}{\begin{array}{c} C \vdash tyvarseq\ tyid = ty\ \langle \texttt{and}\ typbind \rangle \Rightarrow \\ \{\}\ \langle + VE \rangle, \{tyid \mapsto (\Lambda\alpha^{(k)}.\tau, \{\})\}\ \langle + TE \rangle, \{\}\ \langle + T \rangle \end{array}} \tag{4.36}$$

$$\frac{\begin{array}{c} tyvarseq = \alpha^{(k)} \quad C, \alpha^{(k)} t \vdash conbind \Rightarrow VE \quad \text{arity}\, t = k \\ \langle C \vdash typbind \Rightarrow VE', TE', T' \quad t \notin T' \rangle \end{array}}{\begin{array}{l} C \vdash tyvarseq\ tyid \texttt{ = } conbind\ \langle \texttt{and}\ typbind \rangle \Rightarrow \\ \text{Clos}\, VE\ \langle + VE' \rangle, \{tyid \mapsto (t, \text{Clos}\, VE)\}\ \langle + TE' \rangle, \{t\}\ \langle \cup T' \rangle \end{array}} \tag{4.37}$$

(4.37) 만들어지는 데이타 타입 이름 t는 새로운 것이어야 하는데, 이 조건은 (4.19)에서 확인된다. 또한, (4.19)에서 데이타 타입의 재귀적인 선언이 가능하도록 C가 이미 준비된다.

데이타 구성자 정의

$\boxed{C, \tau \vdash conbind \Rightarrow VE}$

$$\frac{\langle C \vdash ty \Rightarrow \tau' \rangle \quad \langle\langle C, \tau \vdash conbind \Rightarrow VE \rangle\rangle}{\begin{array}{l} C, \tau \vdash conid\ \langle \texttt{of}\ ty \rangle\ \langle\langle \mid conbind \rangle\rangle \Rightarrow \\ \{conid \mapsto \tau\}\ \langle + \{conid \mapsto \tau' \to \tau\} \rangle\ \langle\langle + VE \rangle\rangle \end{array}} \tag{4.38}$$

예외상황 정의

$\boxed{C \vdash exnbind \Rightarrow VE}$

$$\frac{\langle C \vdash ty \Rightarrow \tau \rangle \quad \langle\langle C \vdash exnbind \Rightarrow VE \rangle\rangle}{\begin{array}{l} C \vdash conid\ \langle \texttt{of}\ ty \rangle\ \langle\langle \texttt{and}\ exnbind \rangle\rangle \Rightarrow \\ \{conid \mapsto \texttt{exn}\}\ \langle + \{conid \mapsto \tau \to \texttt{exn}\} \rangle\ \langle\langle + VE \rangle\rangle \end{array}} \tag{4.39}$$

모듈타입 정의

$\boxed{B \vdash sigbind \Rightarrow G}$

$$\frac{B \vdash sig \Rightarrow \Sigma \quad \langle B \vdash sigbind \Rightarrow G \rangle}{B \vdash sigid \texttt{ = } sig\ \langle \texttt{and}\ sigbind \rangle \Rightarrow \{sigid \mapsto \Sigma\}\ \langle + G \rangle} \tag{4.40}$$

모듈 정의

$\boxed{B \vdash strbind \Rightarrow SE}$

$$\frac{B \vdash strexp \Rightarrow E \quad \langle B + \text{tynames}\, E \vdash strbind \Rightarrow SE \rangle}{B \vdash strid \texttt{ = } strexp\ \langle \texttt{and}\ strbind \rangle \Rightarrow \{strid \mapsto E\}\ \langle + SE \rangle} \tag{4.41}$$

(4.41) 모듈들이 같이 정의될 때, 모듈들에서 새롭게 만들어지는 타입이름(tynames E)들은 서로 달라야 한다. 따라서 B + tynames E가 두번째 *strbind*의 경우에 배경으로 작용한다.

모듈 타입

$\boxed{B \vdash sig \Rightarrow \Sigma}$

$$\frac{(G \text{ of } B)(sigid) = (T)E \quad T \cap (T \text{ of } B) = \emptyset}{B \vdash sigid \Rightarrow (T)E} \tag{4.42}$$

$$\frac{\langle B \vdash spec \Rightarrow E \quad T = \text{tynames}\, E \setminus T \text{ of } B \rangle}{B \vdash \texttt{sig}\ \langle spec \rangle\ \texttt{end} \Rightarrow \{\}\ \langle + (T)E \rangle} \tag{4.43}$$

$$\frac{\begin{array}{c} B \vdash sig \Rightarrow (T)E \quad E(tylongid) = (t, \{\}) \quad t \in T \\ C \text{ of } B \vdash ty \Rightarrow \tau \quad tyvarseq = \alpha^{(k)} \quad \varphi = \{t \mapsto \Lambda\alpha^{(k)}.\tau\} \\ \varphi E \text{ well-formed} \quad T' = \text{tynames}\,\varphi E \setminus T \text{ of } B \end{array}}{B \vdash sig \texttt{ where type } tyvarseq\ tylongid \texttt{ = } ty \Rightarrow (T')\varphi E} \tag{4.44}$$

(4.42) 모듈이 접속되면서 모듈 타입에 명시된 타입이름들 T가 실현되는데, 이 이름들은 현재 배경에서 이미 알려진 이름이 아니어야 한다. T로 묶여진 이름들은 임의의 이름들일 수 있으므로 항상 그렇게 되도록 바꿀 수 있다.

(4.43) 모듈이 접속되면서 모듈 타입에 명시된 타입이름들 T가 실현되는데, 이 이름들은 현재 배경에서 이미 알려진 이름이 아니어야 한다.

(4.44) 모듈 타입에 명시된 타입 이름들(T)은 모듈이 접속되면서 실현될 터인데, "`where type`$\cdots$"에 의해 그 실현 내용이 더욱 제한된다. 이 때, 데이타 타입이었던 것을 단순 타입으로 제한하는 일은 발생하지 말아야 한다 (φE well-formed, $E(tylongid) = (t, \{\})$). 새로 만들어 지는 모듈 타입이 앞으로 실현할 타입이름들 T'은 현재 배경에서 이미 알려진 이름이 아니어야 한다.

접속 방안

$\boxed{B \vdash spec \Rightarrow E}$

$$\frac{C \text{ of } B \vdash valdesc \Rightarrow VE}{B \vdash \texttt{val } valdesc \Rightarrow \text{Clos}\,VE \text{ in } Env} \tag{4.45}$$

$$\frac{C \text{ of } B \oplus TE \vdash typdesc \Rightarrow VE, TE, T' \quad T' \cap (T \text{ of } B) = \emptyset}{B \vdash \texttt{type } typdesc \Rightarrow (VE, TE) \text{ in } Env} \tag{4.46}$$

$$\frac{C \text{ of } B \vdash exndesc \Rightarrow VE}{B \vdash \texttt{exception } exndesc \Rightarrow VE \text{ in } Env} \tag{4.47}$$

$$\frac{B \vdash strdesc \Rightarrow SE}{B \vdash \texttt{structure } strdesc \Rightarrow SE \text{ in } Env} \tag{4.48}$$

$$\frac{B \vdash sig \Rightarrow T(E)}{B \vdash \texttt{include } sig \Rightarrow E} \tag{4.49}$$

$$\frac{B \vdash spec_1 \Rightarrow E_1 \quad B \oplus E_1 \vdash spec_2 \Rightarrow E_2 \quad \text{Dom}\,E_1 \cap \text{Dom}\,E_2 = \emptyset}{B \vdash spec_1\ \langle ; \rangle\ spec_2 \Rightarrow E_1 + E_2} \tag{4.50}$$

(4.49) 이미 정의된 모듈타입이 현재의 접속방안으로 포섭될 때는, 그 모듈 타입에서 외부와 접속할 때 실현될 타입이름들 T는 (4.42,4.43,4.44)에 의해서 이미 새로운 이름들로 준비된다.

(4.50) 접속방안이 정의될 때 같은 이름의 값이나 타입이 중복되서는 않된다.

값 접속방안

$\boxed{C \vdash valdesc \Rightarrow VE}$

$$\frac{C \vdash ty \Rightarrow \tau \quad \langle \vdash valdesc \Rightarrow VE \rangle}{C \vdash varid \ \texttt{:} \ ty \ \langle \texttt{and} \ valdesc \rangle \Rightarrow \{varid \mapsto \tau\} \ \langle + VE \rangle} \tag{4.51}$$

$$\frac{C \vdash ty \Rightarrow \tau \quad \langle \vdash valdesc \Rightarrow VE \rangle}{C \vdash \texttt{(}opid\texttt{)} \ \texttt{:} \ ty \ \langle \texttt{and} \ valdesc \rangle \Rightarrow \{opid \mapsto \tau\} \ \langle + VE \rangle} \tag{4.52}$$

타입 접속방안

$\boxed{C \vdash typdesc \Rightarrow VE, TE, T}$

$$\frac{tyvarseq = \alpha^{(k)} \quad \text{arity}\, t = k \quad \langle C \vdash typdesc \Rightarrow VE, TE, T' \quad t \notin T' \rangle}{\begin{array}{c} C \vdash tyvarseq \ tyid \ \langle \texttt{and} \ typdesc \rangle \Rightarrow \\ \{\} \ \langle + VE \rangle, \{tyid \mapsto (t, \{\})\} \ \langle + TE \rangle, \{t\} \ \langle \cup T' \rangle \end{array}} \tag{4.53}$$

$$\frac{tyvarseq = \alpha^{(k)} \quad C \vdash ty \Rightarrow \tau \quad \langle C \vdash typbind \Rightarrow VE, TE, T \rangle}{\begin{array}{c} C \vdash tyvarseq \ tyid \ \texttt{=} \ ty \ \langle \texttt{and} \ typbind \rangle \Rightarrow \\ \{\} \ \langle + VE \rangle, \{tyid \mapsto (\Lambda\alpha^{(k)}.\tau, \{\})\} \ \langle + TE \rangle, \{\} \ \langle + T \rangle \end{array}} \tag{4.54}$$

$$\frac{\begin{array}{c} tyvarseq = \alpha^{(k)} \quad C, \alpha^{(k)} t \vdash condesc \Rightarrow VE \quad \text{arity}\, t = k \\ \langle C \vdash typdesc \Rightarrow VE', TE', T' \quad t \notin T' \rangle \end{array}}{\begin{array}{c} C \vdash tyvarseq \ tyid \ \texttt{=} \ condesc \ \langle \texttt{and} \ typdesc \rangle \Rightarrow \\ \text{Clos}V\,E \langle + VE' \rangle, \{tyid \mapsto (t, \text{Clos}\, VE)\} \langle + TE \rangle, \{t\} \ \langle \cup T' \rangle \end{array}} \tag{4.55}$$

(4.53), (4.55) 만들어지는 타입 이름 t는 새로운 것이어야 하는데, 이 조건은 (4.46)에서 확인된다.

데이타 구성자 접속방안

$\boxed{C, \tau \vdash condesc \Rightarrow VE}$

$$\frac{\langle C \vdash ty \Rightarrow \tau' \rangle \quad \langle\langle C, \tau \vdash condesc \Rightarrow VE \rangle\rangle}{\begin{array}{c} C, \tau \vdash conid \ \langle \texttt{of} \ ty \rangle \ \langle\langle \mid condesc \rangle\rangle \Rightarrow \\ \{conid \mapsto \tau\} \ \langle + \{conid \mapsto \tau' \to \tau\} \rangle \ \langle\langle + VE \rangle\rangle \end{array}} \tag{4.56}$$

예외상황 접속방안

$\boxed{C \vdash exndesc \Rightarrow VE}$

$$\frac{\langle C \vdash ty \Rightarrow \tau \quad \text{tyvars}\, \tau = \emptyset \rangle \quad \langle\langle C \vdash exndesc \Rightarrow VE \rangle\rangle}{\begin{array}{c} C \vdash conid \ \langle \texttt{of} \ ty \rangle \ \langle\langle \texttt{and} \ exndesc \rangle\rangle \Rightarrow \\ \{conid \mapsto \texttt{exn}\} \ \langle + \{conid \mapsto \tau \to \texttt{exn}\} \rangle \ \langle\langle + VE \rangle\rangle \end{array}} \tag{4.57}$$

모듈 접속방안

$\boxed{B \vdash strdesc \Rightarrow SE}$

$$\frac{B \vdash sig \Rightarrow (T)E \quad \langle B + \text{tynames}\, E \vdash strdesc \Rightarrow SE \rangle}{B \vdash strid \ \texttt{:} \ sig \ \langle \texttt{and} \ strdesc \rangle \Rightarrow \{strid \mapsto E\} \ \langle + SE \rangle} \tag{4.58}$$

(4.58) 모듈의 접속방안들이 같이 정의될 때, 접속될 모듈들이 실현할 타입이름(tynames E)들은 서로 달라야 한다. 따라서 $B + \text{tynames}\, E$가 두번째 *strdesc*의 경우에 배경으로 작용한다.

프로그램 선언

$\boxed{B \vdash topdec \Rightarrow B'}$

$$\frac{B \vdash sigdec \Rightarrow G \quad B' = (\text{tynames}\, G, G) \text{ in } Basis \quad \text{tyvars}\, B' = \emptyset}{B \vdash sigdec \Rightarrow B'} \tag{4.59}$$

$$\frac{B \vdash fntordec \Rightarrow F \quad B' = (\text{tynames}\, F, F) \text{ in } Basis \quad \text{tyvars}\, B' = \emptyset}{B \vdash fntordec \Rightarrow B'} \tag{4.60}$$

$$\frac{B \vdash strdec \Rightarrow E \quad B' = (\text{tynames}\, E, E) \text{ in } Basis \quad \text{tyvars}\, B' = \emptyset}{B \vdash strdec \Rightarrow B'} \tag{4.61}$$

$$\frac{B \vdash topdec_1 \Rightarrow B_1 \quad B + B_1 \vdash topdec_1 \Rightarrow B_2}{B \vdash topdec_1 \ \langle ; \rangle \ topdec_2 \Rightarrow B_1 + B_2} \tag{4.62}$$

제 2 절 프로그램 식의 기획

프로그램 식

$\boxed{C \vdash e \Rightarrow \tau}$

$$\frac{}{C \vdash c \Rightarrow \text{type}(c)} \tag{4.63}$$

$$\frac{E(varlongid\,/\,conlongid) = \sigma \quad \sigma \succ \tau}{C \vdash varlongid\,/\,conlongid \Rightarrow \tau} \tag{4.64}$$

$$\frac{}{C \vdash \texttt{\{\}} \Rightarrow \texttt{unit}} \tag{4.65}$$

$$\frac{C \vdash labrow \Rightarrow \varrho}{C \vdash \texttt{\{}\ labrow\ \texttt{\}} \Rightarrow \varrho \text{ in } Type} \tag{4.66}$$

$$\frac{C \vdash e \Rightarrow \{lab \mapsto \tau, \cdots\}}{C \vdash e\ \texttt{.}\ lab \Rightarrow \tau} \tag{4.67}$$

$$\frac{}{C \vdash \texttt{[||]} \Rightarrow \tau\ \texttt{array}} \tag{4.68}$$

$$\frac{C \vdash e_i \Rightarrow \tau \quad 0 \le i \le n}{C \vdash \texttt{[|}e_0\texttt{,}\cdots\texttt{,}e_n\texttt{|]} \Rightarrow \tau\ \texttt{array}} \tag{4.69}$$

$$\frac{C \vdash e_1 \Rightarrow \tau\ \texttt{array} \quad C \vdash e_2 \Rightarrow \texttt{int}}{C \vdash e_1\ \texttt{.}\ \texttt{[}\ e_2\ \texttt{]} \Rightarrow \tau} \tag{4.70}$$

$$\frac{C \vdash e_1 \Rightarrow \tau\ \texttt{array} \quad C \vdash e_2 \Rightarrow \texttt{int} \quad C \vdash e_3 \Rightarrow \tau}{C \vdash e_1\ \texttt{.}\ \texttt{[}\ e_2\ \texttt{]}\ \texttt{<-}\ e_3 \Rightarrow \texttt{unit}} \tag{4.71}$$

$$\frac{C \vdash valdec \Rightarrow VE \quad C + VE \vdash e \Rightarrow \tau}{C \vdash \texttt{let}\ valdec\ \texttt{in}\ e\ \texttt{end} \Rightarrow \tau} \tag{4.72}$$

$$\frac{C \vdash e \Rightarrow \tau}{C \vdash \texttt{(}\ e\ \texttt{)} \Rightarrow \tau} \tag{4.73}$$

$$\frac{C \vdash e_1 \Rightarrow \tau' \to \tau \quad C \vdash e_2 \Rightarrow \tau'}{C \vdash e_1\ e_2 \Rightarrow \tau} \tag{4.74}$$

$$\frac{C \vdash e \Rightarrow \tau \quad C \vdash ty \Rightarrow \tau}{C \vdash e\ \texttt{:}\ ty \Rightarrow \tau} \tag{4.75}$$

$$\frac{C \vdash match \Rightarrow \tau}{C \vdash \texttt{fn}\ match \Rightarrow \tau} \tag{4.76}$$

$$\frac{C \vdash e \Rightarrow \tau \quad C \vdash match \Rightarrow \texttt{exn} \to \tau}{C \vdash e\ \texttt{handle}\ match \Rightarrow \tau} \tag{4.77}$$

$$\frac{C \vdash e \Rightarrow \texttt{exn}}{C \vdash \texttt{raise}\ e \Rightarrow \tau} \tag{4.78}$$

$$\frac{C \vdash e \Rightarrow \tau}{C \vdash \texttt{ref}\ e \Rightarrow \tau\ \texttt{ref}} \tag{4.79}$$

$$\frac{C \vdash e_1 \Rightarrow \tau\ \texttt{ref} \quad C \vdash e_2 \Rightarrow \tau}{C \vdash e_1\ \texttt{:=}\ e_2 \Rightarrow \texttt{unit}} \tag{4.80}$$

$$\frac{C \vdash e_1 \Rightarrow \tau\ \texttt{ref}}{C \vdash \texttt{!}\ e \Rightarrow \tau} \tag{4.81}$$

$$\frac{C \vdash e_1 \Rightarrow \tau' \quad C \vdash e_2 \Rightarrow \tau}{C \vdash e_1\ \texttt{;}\ e_2 \Rightarrow \tau} \tag{4.82}$$

$$\frac{C \vdash e \Rightarrow \tau' \quad C \vdash match \Rightarrow \tau' \to \tau}{C \vdash \texttt{case}\ e\ \texttt{of}\ match \Rightarrow \tau} \tag{4.83}$$

(4.64) 잡혀진 타입구도를 구체적인 타입으로 적응시킨다.

(4.78) 예외상황 발생식은 임의의 타입을 갖는다.

레코드 내용

$\boxed{C \vdash labrow \Rightarrow \varrho}$

$$\frac{C \vdash e \Rightarrow \tau \quad \langle C \vdash labrow \Rightarrow \varrho \quad lab \notin \mathrm{Dom}\ \varrho \rangle}{C \vdash lab\ \texttt{=}\ e\ \langle\texttt{,}\ labrow\rangle \Rightarrow \{lab \mapsto \tau\}\ \langle +\varrho\rangle} \tag{4.84}$$

(4.84) 한 레코드의 레이블들은 모두 달라야 한다.

패턴 매치

$\boxed{C \vdash match \Rightarrow \tau}$

$$\frac{C \vdash pat \Rightarrow (VE, \tau) \quad C + VE \vdash e \Rightarrow \tau'}{C \vdash pat\ \texttt{=>}\ e \Rightarrow \tau \to \tau'} \tag{4.85}$$

$$\frac{C \vdash pat \Rightarrow (VE, \tau) \quad C + VE \vdash e \Rightarrow \tau' \quad C \vdash match \Rightarrow \tau \to \tau'}{C \vdash pat \texttt{ => } e \mid match \Rightarrow \tau \to \tau'} \tag{4.86}$$

패턴

$\boxed{C \vdash pat \Rightarrow (VE, \tau)}$

$$\frac{}{C \vdash \_ \Rightarrow (\{\}, \tau)} \tag{4.87}$$

$$\frac{}{C \vdash c \Rightarrow (\{\}, \mathrm{type}(c))} \tag{4.88}$$

$$\frac{C \vdash patrow \Rightarrow (VE, \varrho)}{C \vdash \texttt{\{ } patrow \texttt{ \}} \Rightarrow (VE, \varrho \text{ in } Type)} \tag{4.89}$$

$$\frac{\begin{array}{c}0 \le i, j \le n \quad i \ne j \\ C \vdash pat_i \Rightarrow (VE_i, \tau) \quad \mathrm{Dom}\ VE_i \cap \mathrm{Dom}\ VE_j = \emptyset\end{array}}{C \vdash \texttt{[|} pat_0 \texttt{,} \cdots pat_n \texttt{|]} \Rightarrow (\bigcup_{i=0}^{n} VE_i, \tau\ \texttt{array})} \tag{4.90}$$

$$\frac{C \vdash pat \Rightarrow (VE, \tau)}{C \vdash \texttt{(} pat \texttt{)} \Rightarrow (VE, \tau)} \tag{4.91}$$

$$\frac{C \vdash pat \Rightarrow (VE, \tau)}{C \vdash \texttt{ref } pat \Rightarrow (VE, \tau\ \texttt{ref})} \tag{4.92}$$

$$\frac{}{C \vdash varid \Rightarrow (\{varid \mapsto \tau\}, \tau)} \tag{4.93}$$

$$\frac{}{C \vdash \texttt{(}opid\texttt{)} \Rightarrow (\{opid \mapsto \tau\}, \tau)} \tag{4.94}$$

$$\frac{C(conlongid) = \sigma \quad \sigma \succ \tau^{(k)}t}{C \vdash conlongid \Rightarrow (\{\}, \tau^{(k)}t)} \tag{4.95}$$

$$\frac{C(conlongid) = \sigma \quad \sigma \succ \tau' \to \tau \quad C \vdash pat \Rightarrow (VE, \tau')}{C \vdash conlongid\ pat \Rightarrow (VE, \tau)} \tag{4.96}$$

$$\frac{C \vdash pat \Rightarrow (VE, \tau) \quad C \vdash ty \Rightarrow \tau}{C \vdash pat \texttt{ : } ty \Rightarrow (VE, \tau)} \tag{4.97}$$

$$\frac{\langle C \vdash ty \Rightarrow \tau \rangle \quad C \vdash pat \Rightarrow (VE, \tau) \quad varid \notin \mathrm{Dom}\ VE}{C \vdash varid \langle \texttt{: } ty \rangle\ \texttt{as } pat \Rightarrow (VE + \{varid \mapsto \tau\}, \tau)} \tag{4.98}$$

$$\frac{\langle C \vdash ty \Rightarrow \tau \rangle \quad C \vdash pat \Rightarrow (VE, \tau) \quad opid \notin \mathrm{Dom}\ VE}{C \vdash \texttt{(}opid\texttt{)} \langle \texttt{: } ty \rangle\ \texttt{as } pat \Rightarrow (VE + \{opid \mapsto \tau\}, \tau)} \tag{4.99}$$

$$\frac{C \vdash pat_1 \Rightarrow (VE, \tau) \quad C \vdash pat_2 \Rightarrow (VE, \tau)}{C \vdash pat_1 \texttt{ | } pat_2 \Rightarrow (VE, \tau)} \tag{4.100}$$

(4.90) 같은 패턴 이름이 배열 패턴에서 중복되서 나타나면 않된다.

(4.95) 패턴에 있는 이름이 인자가 필요없는 데이타 타입 구성자이거나 예외상황 구성자인 경우.

(4.96) 패턴에 있는 이름이 인자를 필요로하는 데이타 타입 구성자이거나 예외상황 구성자인 경우.

(4.100) 두 패턴이 "또는"의 관계로 있을때는 두 패턴에서 기획되는 것이 (VE, τ) 일치해야 한다. 즉, 한 패턴에서 변수가 사용되었으면 다른 패턴에서도 같은 변수가 사용되야 하고, 그 타입이 일치해야 한다.

레코드 패턴

$\boxed{C \vdash patrow \Rightarrow (VE, \varrho)}$

$$\frac{}{C \vdash \ldots \Rightarrow (\{\}, \varrho)} \tag{4.101}$$

$$\frac{\begin{array}{c} C \vdash pat \Rightarrow (VE_1, \tau) \\ \langle C \vdash patrow \Rightarrow (VE_2, \varrho) \quad \mathrm{Dom}\ VE_1 \cap \mathrm{Dom}\ VE_2 = \emptyset \rangle \end{array}}{C \vdash lab\ \texttt{=}\ pat\ \langle\texttt{,}\ patrow\rangle \Rightarrow (VE_1\ \langle + VE_2\rangle, \{lab \mapsto \tau\}\langle + \varrho\rangle)} \tag{4.102}$$

(4.102) 같은 패턴 이름이 레코드 패턴에서 중복되서 나타나면 않된다.

타입식

$\boxed{C \vdash ty \Rightarrow \tau}$

$$\frac{tyvar = \alpha}{C \vdash tyvar \Rightarrow \alpha} \tag{4.103}$$

$$\frac{C \vdash tyrow \Rightarrow \varrho}{C \vdash \{tyrow\} \Rightarrow \varrho\ \mathrm{in}\ Type} \tag{4.104}$$

$$\frac{\begin{array}{c} tyseq = (ty_1, \cdots, ty_k) \quad \forall i,\ C \vdash ty_i \Rightarrow \tau_i \\ (TE\ \mathrm{of}\ C)(tylongid) = (\theta, VE) \end{array}}{C \vdash tyseq\ tylongid \Rightarrow \tau^{(k)}\theta} \tag{4.105}$$

$$\frac{C \vdash ty_1 \Rightarrow \tau_1 \quad C \vdash ty_2 \Rightarrow \tau_2}{C \vdash ty_1\ \texttt{->}\ ty_2 \Rightarrow \tau_1 \to \tau_2} \tag{4.106}$$

$$\frac{C \vdash ty \Rightarrow \tau}{C \vdash (ty) \Rightarrow \tau} \tag{4.107}$$

(4.105) $longid$가 의미하는 타입함수 θ를 구체적인 타입들 ty_i에 적용한 결과 타입을 만든다.

레코드 내용 타입

$\boxed{C \vdash tyrow \Rightarrow \varrho}$

$$\frac{C \vdash ty \Rightarrow \tau \quad \langle C \vdash tyrow \Rightarrow \varrho \quad lab \notin \mathrm{Dom}\ \varrho\rangle}{C \vdash lab\ \texttt{:}\ ty\ \langle\texttt{,}\ tyrow\rangle \Rightarrow \{lab \mapsto \tau\}\ \langle + \varrho\rangle} \tag{4.108}$$

(4.108) 한 레코드의 레이블들은 모두 달라야 한다.

부록 A

Standard ML및 OCaml과의 비교점

nML은 Standard ML보다는 OCaml과 더욱 비슷하다고 할 수 있다. 모든 것을 비교한 것은 아니지만, 주목할 점들은 아래와 같다.

- 프로그램내에 있는 이름들은 OCaml의 방식을 따른다. 즉, 변수 이름과 타입 이름들은 소문자로 시작한다. 모듈, 모듈함수, 모듈 타입, 데이타 구성자, 예외상황 구성자들은 대문자로 시작한다. 한글에서 대문자 이름은 _로 시작한다.

  이와같은 간단한 제약으로 프로그램이 읽기 쉬워진다.

- 기본적으로 제공되는 연산자와 데이타 및 예외상황 구성자들(Section 2)은 프로그램에서 다시 정의될 수 없다. 예를들어 +가 다르게 정의될 수 있다는 가능성은 너무 열려진 것이 아닐까.

  당연한 제약으로 프로그램이 읽기 쉬워진다.

- 연산자(*opid* ::= *prefixid*|*infixid*)의 새치기 위치(fixity)가 연산자의 이름으로 정해진다. Standard ML과는 달리, 프로그래머가 새치기 위치를 선언하거나 취소할 수는 없다. 새치기 성질을 없앤 연산자는 괄호안에 넣어 표현한다.

- 타입과 예외상황의 선언은 프로그램 식(expression)내에서는 불가능하다. 즉, 모듈 레벨에서만 가능하며 `let`-프로그램식에서는 불가능하다.

  Standard ML에서와 같이 프로그램식이 계산중에 데이타 타입을 생성하거나 예외상황의 구성자를 만드는 것이 유용한 경우는 드믈다.

- 레코드 타입은 Standard ML을 따른다. 즉, 레코드 타입마다 타입 이름이 주어질 필요는 없다. 반면에, OCaml에서 처럼 부분적인 레코드 필드 이름만으로 레코드 타입을 유추할 수는 없다.

- OCaml과는 달리 변조가능한 레코드 필드(mutable record field)를 선언할 수 없다. 변조가능한 레코드 필드는 배열(array) 뿐이다. 변조가능한 변수

는 변수의 메모리 주소를 알 방법은 없지만 그곳에 있는 값을 변조할 수 있는 변수를 뜻한다.

변조가능한 레코드 필드가 자유롭게 선언될 수 있으면 의미정의가 복잡해 지고, 프로그램을 이해하는 것이 그만큼 어려워 진다.

- 무더기 패턴(or-pattern)에서 OCaml과는 달리, 같은 패턴 변수를 사용할 수 있다. 예를 들어, "`A x | B x => 1`"이 가능하다 (rule (4.100)).

- 모듈 시스템은 OCaml 방식을 따른다 (opaque signature matching + manifest types). 즉, 모듈타입과 접속한 모듈은, 모듈타입에서 지정한 정보이외에는 새나가지는 않으며(opaque signature matching, rule (4.32)), 모듈타입에서 타입의 정의를 드러낼 수는 있다 (manifest types, rule (4.44)). Standard ML에서와 같이 모듈이 접속되면서 모듈타입에서 지정한 것 보다 많은 내용이 유출되는 (transparent signature matching) 경우는 없다.

  Standard ML의 모듈 시스템보다 간단할 필요가 있다. 일반 프로그래머들이 Standard ML 모듈 시스템의 모든것을 이해하고 프로그램한다는 것을 기대하기는 어렵다.

- 프로그램에 드러난 타입변수의 유효범위는 (rule (4.10)) Standard ML의 방식대로 한다.

- 타입이 동치확인 가능한지 (equality type) 아닌지를 기획단계에서 (타입시스템에 의해) 검증하지 않는다. 실행중에 "e_1=e_2" 에서 e_i값이 함수값을 품고 있어서 동치확인이 불가능하면 예외상황을 발생시킨다.

  Standard ML에서는 동치확인 가능한 타입들을 기획단계에서 검증해 주는 대신에, 프로그래머가 간단하게 이해할 수 없는 부분이 생겼다. 예를 들어, 타입 변수의 동치확인 가능성(`'a` 이면 불가능, `''a` 이면 가능)이 역할을 하는 곳이 있고 하지 않는 곳이 있다. 타입 선언들에서는 타입 변수의 동치확인 성질이 아무 역할을 하지 않지만, 값 선언에서는 역할을 한다. 즉 아래의 타입선언들은 같은 것이지만

```
type 'a t = 'a * int
type ''a t = ''a * int
```

  아래 두 함수는 전혀 다른 함수이다:

```
fun f(x:'a t, y) = x=y
fun g(x:''a t, y) = x=y
```

  타입변수가 `'a` 이냐 `''a` 이냐에 따라 판이해 진다. 함수 `f`는 위의 타입선언과 관련해서 받아들여지지 않는다. "`x: 'a t`" 에서는 `x` 가 동치확인 불가능하다고 선언해 놓고 "`x=y`" 에서는 동치계산을 시키고 있으므로. 함수 `g`는 `x`의 선언된 타입과 그 사용이 일치하므로, 받아들인다.

  일반 프로그래머들이 타입 변수의 동치확인 가능성이 유효한 경우와 그렇지 않은 경우를 쉽게 이해하리라고 기대할 수 없다. 타입 시스템 바깥에서 (아마도, 프로그램 분석(static analysis)을 이용해서) 동치확인이 될 수 없는 경우를 안전하게 지적할 수 있도록 한다.