프로그램 분석 시스템 동물원
Program Analysis System Zoo

Kwangkeun Yi
Research On Program Analysis System
Programming Research Laboratory
`ropas.snu.ac.kr`
SNU/KAIST

2004년 3월 19일

차 례

1 장

문법구조

1.1 문법

다음의 표기법을 따른다:

$\mid$	alternative	$()$	grouping
$\langle\rangle$	optional	$\bullet^*$	zero or more
$\bullet^+$	one or more	$\dagger$	sugared alternative
αrow	one or more α's separated by **,**		

$$
\begin{array}{rcl}
\mathit{integer} & ::= & \langle\texttt{-}\rangle(\texttt{0}-\texttt{9})^{+} \\
 & | & (\texttt{0X}|\texttt{0x})(\texttt{0}-\texttt{9}|\texttt{A}-\texttt{F}|\texttt{a}-\texttt{f})^{+} \\
 & | & (\texttt{0O}|\texttt{0o})(\texttt{0}-\texttt{7})^{+} \\
 & | & (\texttt{0B}|\texttt{0b})(\texttt{0}-\texttt{1})^{+} \\
\mathit{comment} & ::= & \text{balanced } \texttt{(* *)}\text{, between which any character can appear.} \\
 & | & \text{from } \texttt{//} \text{ to the end of the line} \\
 & & \\
\mathit{alphanum} & ::= & \texttt{a}-\texttt{z} \mid \texttt{A}-\texttt{Z} \mid \mathit{hangul} \mid \texttt{0}-\texttt{9} \mid \texttt{\_} \mid \texttt{'} \\
\mathit{upper} & ::= & \texttt{A}-\texttt{Z} \mid \texttt{\_} \\
\mathit{lower} & ::= & \texttt{a}-\texttt{z} \mid \mathit{hangul} \\
\mathit{hangul} & ::= & \text{syllables of KSX1001 (a.k.a. KSC5601 or eur-kr)} \\
 & | & \text{syllables of KSX1005-1 (a.k.a. KSC5700, unicode, or ISO/IEC10646-1)} \\
\mathit{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{\textasciitilde} \mid \texttt{`} \mid \texttt{\textasciicircum} \mid \texttt{|} \mid \texttt{*} \\
\mathit{lid} & ::= & \mathit{lower}(\mathit{alphanum})^{*} \\
\mathit{uid} & ::= & \mathit{upper}(\mathit{alphanum})^{*} \\
\mathit{sid} & ::= & \mathit{sym}\,\mathit{sym}^{+} \\
\mathit{id} & ::= & \mathit{lid} \mid \mathit{uid} \mid \mathit{sid} \\
\mathit{varid} & ::= & \mathit{id} \\
\mathit{ctlid} & ::= & \mathit{id} \\
\mathit{elmtid} & ::= & \mathit{id} \\
\mathit{setid} & ::= & \mathit{uid} \\
\mathit{latid} & ::= & \mathit{uid} \\
\mathit{domid} & ::= & \mathit{setid} \mid \mathit{latid} \\
\mathit{anaid} & ::= & \mathit{id} \\
\mathit{sigid} & ::= & \mathit{id} \\
\mathit{temid} & ::= & \mathit{id} \\
\mathit{cvarid} & ::= & \mathit{id} \\
\mathit{conid} & ::= & \mathit{id} \\
\alpha\mathit{longid} & ::= & \alpha\mathit{id} \ \mid\ \mathit{anaid}.\alpha\mathit{id}
\end{array}
$$

$$
\begin{array}{rcl}
\mathit{topdec} & ::= & \mathit{anadec} \\
 & \mid & \mathit{sigdec} \\
 & \mid & \mathit{temdec} \\
 & \mid & \mathit{topdec}_1\ \mathit{topdec}_2 \\
 & & \\
\mathit{adec} & ::= & \mathit{domdec} \\
 & \mid & \mathit{semdec} \\
 & \mid & \mathit{querydec} \\
 & \mid & \mathit{adec}_1\ \mathit{adec}_2 \\
 & & \\
\mathit{anadec} & ::= & \texttt{analysis}\ \mathit{anaid}\ \texttt{=}\ \mathit{anaexp} \\
\mathit{anaexp} & ::= & \texttt{ana}\ \mathit{adec}\ \texttt{end} \\
 & \mid & \mathit{temid}\ \texttt{(}\mathit{anaexprow}\texttt{)} \\
 & \mid & \mathit{anaid} \\
 & & \\
\mathit{sigdec} & ::= & \texttt{signature}\ \mathit{sigid}\ \texttt{=}\ \mathit{sigexp} \\
\mathit{sigexp} & ::= & \texttt{sig}\ \mathit{adesc}\ \texttt{end} \\
 & \mid & \mathit{sigid} \\
 & & \\
\mathit{temdec} & ::= & \texttt{analysis}\ \mathit{temid}\texttt{(}(\mathit{anaid}\ \texttt{:}\ \mathit{sigexp})\mathit{row}\texttt{)}\ \texttt{=}\ \mathit{anaexp} \\
 & & \\
\mathit{adesc} & ::= & \texttt{set}\ \mathit{setdesc} \\
 & \mid & \texttt{lattice}\ \mathit{latdesc} \\
 & \mid & \texttt{val}\ \mathit{varid}\ \texttt{:}\ \mathit{ty} \\
 & \mid & \texttt{eqn}\ \mathit{varid}\ \texttt{:}\ \mathit{ty} \\
 & \mid & \texttt{query}\ \mathit{ctlid}\ \texttt{:}\ \mathit{ty} \\
 & \mid & \mathit{adesc}_1\ \mathit{adesc}_2 \\
 & \dagger & \texttt{set}\ \mathit{setdescrow} \\
 & \dagger & \texttt{lattice}\ \mathit{setdescrow} \\
\mathit{setdesc} & ::= & \mathit{setid} \mid \mathit{setid}\ \texttt{:}\ \mathit{kind} \mid \mathit{setbind} \\
\mathit{latdesc} & ::= & \mathit{latid} \mid \mathit{latid}\ \texttt{:}\ \mathit{kind} \mid \mathit{latbind}
\end{array}
$$

domdec	::=	*setdec* $\mid$ *latdec* $\mid$ *winadec*	
setdec	::=	`set` *setbind*	
setbind	::=	*setid* `=` *setexp*	
setexp	::=	`/`*tylongid*`/`	nML type id
	$\mid$	*setlongid*	set id
	$\mid$	`{` e_1 `...` e_2 `}`	integer interval set
	$\mid$	`{` *elmtidrow* `}`	enumerated set
	$\mid$	`power` *setexp*	power set
	$\mid$	*setexp*$_1$ `*` *setexp*$_2$	cartesian product
	$\mid$	*setexp*$_1$ `+` *setexp*$_2$	separated sum
	$\mid$	*setexp*$_1$ `->` *setexp*$_2$	finite function set
	$\mid$	*setexp* `constraint` *cnstdec*	constraint set
	$\mid$	`(` *setexp* `)`	
cnstdec	::=	`var = {` *cvaridrow* `}` $\langle$`index` *setexp*$\rangle$	
		`rhs =` *rhs*	
rhs	::=	*cvar*	
	$\mid$	*conid* $\langle$*carg*$\rangle$ $\langle$`: atomic`$\rangle$	
	$\mid$	*rhs*$_1$ $\mathtt{\vert}$ *rhs*$_2$	
cvar	::=	`var` $\mid$ `var` *setlongid*	
carg	::=	*cvar*	
	$\mid$	*setexp*	
	$\mid$	`(` *cargrow* `)`	
latdec	::=	`lattice` *latbind*	
latbind	::=	*latid* `=` *latexp*	
latexp	::=	`/`*strlongid*`/`	nML structure id
	$\mid$	*latlongid*	lattice id
	$\mid$	`flat` *setexp*	flat lattice
	$\mid$	`power` *setexp*	powerset lattice
	$\mid$	*latexp*$_1$ `*` *latexp*$_2$	cartesian product
	$\mid$	*latexp*$_1$ `+` *latexp*$_2$	coalesced sum
	$\mid$	*latexp*$_1$ `->` *latexp*$_2$	atomic function lattice
	$\mid$	*setexp* `->` *latexp*	dependent product lattice
	$\mid$	`order` *setexp* `with` *jmdec* $\langle$*jmdec*$\rangle$	lattice with an explicit join/meet
	$\mid$	`(` *latexp* `)`	
jmdec	::=	`join` *match*	user-defined join operator
	$\mid$	`meet` *match*	user-defined meet operator
winadec	::=	`widen` *latid* `with` *match*	
	$\mid$	`narrow` *latid* `with` *match*	
kind	::=	`syntree` $\mid$ `index` $\mid$ `integer` $\mid$ `power`	
	$\mid$	`sum` $\mid$ `product` $\mid$ `arrow`	

semdec	::=	*valdec*	
	$\mid$	*eqndec*	
	$\mid$	*ccrdec*	
	$\mid$	*cimdec*	
valdec	::=	`val` *vbind*	auxiliary semantic value
	$\mid$	`val rec` *vbind*	auxiliary semantic value
	†	`fun` *fbind*	
	†	`map` *fbind*	
vbind	::=	*pat* `=` *e* ⟨`and` *vbind*⟩	
fbind	::=	*varid pat* `=` *e* $\mathtt{\vert} \cdots \mathtt{\vert}$ *varid pat* `=` *e* ⟨`and` *fbind*⟩	
eqndec	::=	`eqn` *ebind*	semantic equation
	$\mid$	`eqn rec` *ebind*	semantic equation
	†	`eqn` *efbind*	
ebind	::=	*varid* `=` *e* ⟨`and` *ebind*⟩	
efbind	::=	*varid pat* `=` *e* $\mathtt{\vert} \cdots \mathtt{\vert}$ *varid pat* `=` *e* ⟨`and` *efbind*⟩	
ccrdec	::=	`ccr` *ccrbind*	constraint closure rule
ccrbind	::=	*cnstguard* $\texttt{----}^{+}$ *constraintrow* ⟨$\mathtt{\vert}$ *ccrbind*⟩	
cnstguard	::=	*constraint*	
	$\mid$	*guard*	
	$\mid$	$\mathit{cnstguard}_1$ `,` $\mathit{cnstguard}_2$	
constraint	::=	*cvarexp* `<-` *rhsexp*	
	†	*cvarexp* `<-` *rhsexp* $(\texttt{+}\ \mathit{rhsexp})^{+}$	
rhsexp	::=	*cvarexp*	
	$\mid$	*conlongid* ⟨*cargexp*⟩	
cargexp	::=	*cvarexp*	
	$\mid$	*pat*	
	$\mid$	`(` *cargexprow* `)`	
cvarexp	::=	*cvarlongid* $\mid$ *cvarlongid* `@` *pat*	
cimdec	::=	`cim` *cimbind*	constraint conid's image declaration
cimbind	::=	*conlongid* ⟨*pat*⟩ `=` *e* ⟨$\mathtt{\vert}$ *cimbind*⟩	

e	::=	/ *nexp* /	nML expr
	$\mid$	*setlongid*	set itself
	$\mid$	*const*	constant
	$\mid$	*varlongid*	bound id
	$\mid$	*constraint*	constraint
	$\mid$	e_1 *bop* e_2	binary op
	$\mid$	{ e_1 ... e_2 }	integer set
	$\mid$	{ *erow* }	set
	$\mid$	{ *erow* $\mid$ *qual* }	set comprehension
	$\mid$	{ *mrulerow* }	map
	$\mid$	{ *mrulerow* $\mid$ *qual* }	map comprehension
	$\mid$	{}	empty set/map
	$\mid$	+ e	fold join
	$\mid$	* e	fold meet
	$\mid$	(e_1 , e_2)	tuple
	$\mid$	e . 1 $\mid$ e . 2	projection
	$\mid$	let *valdec* in e end	local expr
	$\mid$	fn *match*	abstraction
	$\mid$	e_1 e_2	application or map image
	$\mid$	(e)	
	$\mid$	e : *ty*	coercion
	$\mid$	$\langle$pre $\mid$ post$\rangle$ (*varlongid* $\mid$ *cvarlongid*) @ e	solution look-up
	†	(e , *erow*)	tuple
	†	e . *domlongid*	projection
	†	e [*mrule*]	modifying map
	†	mp *match*	map
	†	case e of *match*	branch
	†	if e_1 then e_2 else e_3	branch
	†	map e_1 e_2	mapping
bop	::=	+ $\mid$ * $\mid$ -	join, meet, set-minus
	$\mid$	*rop*	relational operators
const	::=	*integer*	
	$\mid$	*elmtlongid*	set element id
	$\mid$	top	lattice top
	$\mid$	^	lattice top
	$\mid$	bottom	lattice bottom
	$\mid$	--	lattice bottom
	$\mid$	true	
	$\mid$	false	
ty	::=	int $\mid$ *domlongid* $\mid$ / *tylongid* /	
	$\mid$	ty_1 * ty_2 $\mid$ ty_1 + ty_2	
	$\mid$	ty_1 -> ty_2 $\mid$ power *ty* $\mid$ flat *ty*	
	$\mid$	(*ty*)	
	$\mid$	*ty* : *kind*	

qual	::=	$gen\ \langle \texttt{,}\ guard\rangle$	
gen	::=	*pat* `from` *e*	for each element of a set
	$\mid$	*mpat* `from` *e*	for each entry of a map
	$\mid$	gen_1 `,` gen_2	
guard	::=	e_1 *rop* e_2	relation
	$\mid$	e_1 `in` e_2	membership
	$\mid$	`not` *guard*	
	$\mid$	$guard_1$ `and` $guard_2$	
	$\mid$	$guard_1$ `or` $guard_2$	
	$\mid$	`!` *gen* `.` *guard*	for all
	$\mid$	`?` *gen* `.` *guard*	for some
	$\mid$	`(` *guard* `)`	
	†	*guardrow*	conjunction
rop	::=	`<` $\mid$ `>` $\mid$ `=` $\mid$ `<=` $\mid$ `>=`	
match	::=	*mrule* $\langle$`|` *match*$\rangle$	
mrule	::=	*pat* `=>` *e*	
pat	::=	`/`*npat*`/`	nML pattern
	$\mid$	`_`	wild pattern
	$\mid$	*varid*	pattern var
	$\mid$	`{` *patrow* $\langle$`...`$\rangle$ `}`	set pattern
	$\mid$	`{` pat_1 `...` pat_2 `}`	interval set pattern
	$\mid$	`{` *mpatrow* $\langle$`...`$\rangle$ `}`	map pattern
	$\mid$	`(` pat_1 `,` pat_2 `)`	tuple pattern
	$\mid$	*pat* `with` *guard*	guarded pattern
	$\mid$	pat_1 `or` pat_2	or pattern
	$\mid$	*varid* `as` *pat*	as pattern
	$\mid$	*pat* `:` *ty*	
	$\mid$	`(` *pat* `)`	
	†	*const*	const pattern
	†	`(` *pat* `,` *patrow* `)`	tuple pattern
	†	*pat* *rop* *e*	relation pattern
	†	*pat* `in` *e*	member pattern
mpat	::=	*pat* `=>` *pat*	

$\mathit{querydec}$	$::=$	$\texttt{query}\ \mathit{ctlbind}$	
$\mathit{ctlbind}$	$::=$	$\mathit{ctlid}\ \texttt{=}\ \mathit{ctl}\ \langle \texttt{and}\ \mathit{ctlbind} \rangle$	
ctl	$::=$	$\mathit{varid}\ \texttt{:}\ \langle \texttt{pre} \mid \texttt{post} \rangle\ \mathit{varid}\ \texttt{.}\ (\mathit{form} \mid \mathit{guard})$	CTL formula with a binder
	$\mid$	$\mathit{varid}\ \texttt{:}\ \langle \texttt{pre} \mid \texttt{post} \rangle\ \texttt{(}\mathit{anaid}\,\texttt{.}\,\mathit{varid}\texttt{)}\ \texttt{.}\ (\mathit{form} \mid \mathit{guard})$	CTL formula with a binder
	$\mid$	$\texttt{(}\ \mathit{ctl}\ \texttt{)}$	
form	$::=$	$\mathit{ctlid}\ \mathit{varid}$	ctl application
	$\mid$	$\texttt{not}\ \mathit{form}$	
	$\mid$	$\mathit{form}_1\ \texttt{and}\ \mathit{form}_2$	
	$\mid$	$\mathit{form}_1\ \texttt{or}\ \mathit{form}_2$	
	$\mid$	$\mathit{form}_1\ \texttt{->}\ \mathit{form}_2$	implication
	$\mid$	$\mathit{upath}\ \mathit{ctl}$	unary path formula
	$\mid$	$\mathit{bpath}\ \texttt{(}\ \mathit{ctl}_1\ \texttt{,}\ \mathit{ctl}_2\ \texttt{)}$	binary path formula
	$\mid$	$\texttt{(}\ \mathit{form}\ \texttt{)}$	
	$\dagger$	$\mathit{form}_1\ \texttt{<->}\ \mathit{form}_2$	equivalence
upath	$::=$	$\texttt{AX} \mid \texttt{AF} \mid \texttt{AG}$	
	$\mid$	$\texttt{EX} \mid \texttt{EF} \mid \texttt{EG}$	
bpath	$::=$	$\texttt{AU} \mid \texttt{EU}$	until

1.2 설탕 구조

자유로운 이름이 설탕이 녹으면서 묶이지 않도록 한다.

$\texttt{set}\ \mathit{setdesc}_1\,\texttt{,}\cdots\texttt{,}\,\mathit{setdesc}_n$	$\equiv$	$\texttt{set}\ \mathit{setdesc}_1\ \cdots\ \texttt{set}\ \mathit{setdesc}_n$	
$\texttt{lattice}\ \mathit{latdesc}_1\,\texttt{,}\cdots\texttt{,}\,\mathit{latdesc}_n$	$\equiv$	$\texttt{lattice}\ \mathit{latdesc}_1\ \cdots\ \texttt{lattice}\ \mathit{latdesc}_n$	
$\texttt{(}\ e_1\texttt{,}e_2\texttt{,}e_3\ \texttt{)}$	$\equiv$	$\texttt{(}\ e_1\texttt{,(}\ e_2\texttt{,}e_3\ \texttt{))}$	
$e\ \texttt{.}\ \mathit{domlongid}$ as $e\ \texttt{.}\ \cdots\texttt{.}\mathit{domid}$	$\equiv$	$e\ \texttt{.}\ \texttt{k}$	$e : D = A_1 \times \cdots \times A_n$ and $\mathit{domid} = A_k$
$e\ \texttt{[}\ \mathit{pat}\ \texttt{=>}\ e'\ \texttt{]}$	$\equiv$	$\texttt{\{}\ \mathit{pat}\ \texttt{=>}\ e'\ \texttt{,}\ \texttt{x}\ \texttt{=>}\ e\ \texttt{x}\ \texttt{\}}$	new $\texttt{x}$
$\texttt{mp}\ \mathit{mrule}_1\ \texttt{\|}\cdots\texttt{\|}\ \mathit{mrule}_n$	$\equiv$	$\texttt{\{}\ \mathit{mrule}_1\ \texttt{,}\cdots\texttt{,}\ \mathit{mrule}_n\ \texttt{\}}$	
$\texttt{case}\ e\ \texttt{of}\ \mathit{match}$	$\equiv$	$\texttt{(}\ \texttt{fn}\ \mathit{match}\ \texttt{)}\ e$	
$\texttt{if}\ e_1\ \texttt{then}\ e_2\ \texttt{else}\ e_3$	$\equiv$	$\texttt{case}\ e_1\ \texttt{of}\ \texttt{true}\ \texttt{=>}\ e_2\ \texttt{\|}\ \texttt{false}\ \texttt{=>}\ e_3$	
$\texttt{map}\ e_1\ e_2$	$\equiv$	$\texttt{\{}\ e_1\ \texttt{x}\ \texttt{\|}\ \texttt{x}\ \texttt{from}\ e_2\ \texttt{\}}$	
$\mathit{guard}_1\ \texttt{,}\ \mathit{guard}_2$	$\equiv$	$\mathit{guard}_1\ \texttt{and}\ \mathit{guard}_2$	
$\mathit{cvarexp}\ \texttt{<-}\ \mathit{rhsexp}_1\ \texttt{+}\ \mathit{rhsexp}_2$	$\equiv$	$\mathit{cvarexp}\ \texttt{<-}\ \mathit{rhsexp}_1\ \texttt{,}\ \mathit{cvarexp}\ \texttt{<-}\ \mathit{rhsexp}_2$	
$\texttt{(}\ \mathit{pat}_1\ \texttt{,}\ \mathit{pat}_2\ \texttt{,}\ \mathit{pat}_3\ \texttt{)}$	$\equiv$	$\texttt{(}\ \mathit{pat}_1\ \texttt{,}\ \texttt{(}\ \mathit{pat}_2\ \texttt{,}\ \mathit{pat}_3\ \texttt{)}\ \texttt{)}$	
const	$\equiv$	$\texttt{x}\ \texttt{with}\ \texttt{x}\ \texttt{=}\ \mathit{const}$	new $\texttt{x}$
$\mathit{pat}\ \mathit{rop}\ e$	$\equiv$	$\texttt{x}\ \texttt{as}\ \mathit{pat}\ \texttt{with}\ \texttt{x}\ \mathit{rop}\ e$	new $\texttt{x}$
$\mathit{pat}\ \texttt{in}\ e$	$\equiv$	$\texttt{x}\ \texttt{as}\ \mathit{pat}\ \texttt{with}\ \texttt{x}\ \texttt{in}\ e$	new $\texttt{x}$
$\texttt{fun}\ \mathit{varid}\ \mathit{pat}_1\ \texttt{=}\ e_1\ \texttt{\|}\ \mathit{varid}\ \mathit{pat}_2\ \texttt{=}\ e_2$	$\equiv$	$\texttt{val rec}\ \mathit{varid}\ \texttt{=}\ \texttt{fn}\ \mathit{pat}_1\ \texttt{=>}\ e_1\ \texttt{\|}\ \mathit{pat}_2\ \texttt{=>}\ e_2$	
$\texttt{map}\ \mathit{varid}\ \mathit{pat}_1\ \texttt{=}\ e_1\ \texttt{\|}\ \mathit{varid}\ \mathit{pat}_2\ \texttt{=}\ e_2$	$\equiv$	$\texttt{val}\ \mathit{varid}\ \texttt{=}\ \texttt{\{}\ \mathit{pat}_1\ \texttt{=>}\ e_1\ \texttt{,}\ \mathit{pat}_2\ \texttt{=>}\ e_2\ \texttt{\}}$	
$\texttt{eqn}\ \mathit{varid}\ \mathit{pat}_1\ \texttt{=}\ e_1\ \texttt{\|}\ \mathit{varid}\ \mathit{pat}_2\ \texttt{=}\ e_2$	$\equiv$	$\texttt{eqn rec}\ \mathit{varid}\ \texttt{=}\ \texttt{fn}\ \mathit{pat}_1\ \texttt{=>}\ e_1\ \texttt{\|}\ \mathit{pat}_2\ \texttt{=>}\ e_2$	
$\mathit{form}_1\ \texttt{<->}\ \mathit{form}_2$	$\equiv$	$\mathit{form}_1\ \texttt{->}\ \mathit{form}_2\ \texttt{and}\ \mathit{form}_2\ \texttt{->}\ \mathit{form}_1$	

1.3 우선순위와 방향성

- 도메인 식에서의 우선순위(내림차순)와 방향성

constructs	associativity
`power`, `flat`	right
`*`	right
`+`	left
`->`	right
`order`	–

- 분석 식에서의 우선순위(내림차순)와 방향성

constructs	associativity
`@`	left
`.`	left
`[`*mrule*`]`	left
application	left
`+` (prefix), `*` (prefix)	right
`*` (infix)	left
`+` (infix), `-` (infix)	left
<,>,=,<=,>=	left
`not`	right
`and`	right
`or`	right
`in`	right
`,`	left
`:`	left
`case`, `fn`, `mp`	right

- 패턴에서의 우선순위(내림차순)와 방향성

constructs	associativity
`:`	left
`as`	left
`with`	left

- 탐색 식에서의 우선순위(내림차순)와 방향성

constructs	associativity
`not`	right
`and`	left
`or`	left
`->`	left
(`A`|`E`|`U`)(`X`|`F`|`G`)	right

1.4 예약된 심볼들

```
analysis ana end signature sig set lattice atomic val eqn query
power constraint index var rhs flat order join meet widen narrow
with syntree index integer sum product arrow val rec fun map ccr
cim and pre post top bottom true false int not or let in fn mp
case of as from widen AX AF AG AU EX EF EG EU ( ) : | { }
... * + -> <- < > [ ] => _ ! ? . , = <= >= <-> @ ^ __
```

1.5 문법적인 제약들

- 집합과 래티스에서 재귀적인 선언은 불가능하다. 스스로 재귀적이던가

몇개서 서로 재귀적으로 물려서 선언되는 것은 없다. 이 제약조건은 집합과 래티스는 각각 nML의 한 모듈로 컴파일되는 데, nML에서는 상호 재귀적인 모듈선언이 불가능 하기 때문이다.

- as-패턴과 with-패턴이 녹을 수 있는 설탕이기 위해서는 패턴이 계산식(expression)의 형태를 가지고 있어야 한다. 예를들어, _(wildcard)가 패턴에 있으면 않된다.
- 제약식 선언(*cnstdec*)에서 선언되는 제약식 함수심볼(*conid*)들과 제약식 변수(*cvarid*)들은 모두 달라야 한다.
- 제약식 푸는 규칙(*ccrdec*)에서 하나의 제약식(*constraint*)이나 조건절(*guard*)에서 사용되는 패턴 변수들은 모두 달라야 한다.

1.6 예

- Simultaneous equations:

```
analysis Eqn =
 ana
   lattice A = power {a,b,c,d}

   eqn x1 = x2 + x3 * {a,b}
   and x2 = {b,c} * x3
   and x3 = x1 + x2
 end
```

- 0CFA analysis for lambda calculus in the abstract interpretation style:

```
analysis TinyCfa =
 ana
    set Var = /Exp.var/
    set Lam = /Exp.expr/
    lattice Val = power Lam
    lattice State = Var -> Val

    widen Val with {/Lam(x,Lam _)/ ...} => top

    eqn E(/x/,s) = s /x/
      | E(/Lam(x,e)/, s) = {/Lam(x,e)/}
      | E(/App(e1,e2)/, s) = let val lams = E(/e1/, s)
                                 val v = E(/e2/, s)
                             in
                              +{ E(/e/,s+bot[/x/=>v]) | /Lam(x,e)/ from lams }
                             end
 end
```

- 0CFA analysis for lambda calculus in Heintze's style constraint-based analysis (a.k.a. set-based analysis)

```
analysis Sba =
 ana
```

```
  set Var = /Ast.id/
  set Exp = /Ast.exp/
  set Val = power Exp
           constraint
             var = {X} index Var + Exp
             rhs = var
                 | app(var, var)
                 | lam(Var, Exp) : atomic

  (* constraint collection *)

  eqn Col /Ast.Var(x)/ = {}
    | Col /Ast.Lam(x,body) as e/ = { X@/e/ <- lam(/x/,/body/) }
                                 + Col /body/
    | Col /Ast.App(e,e') as e/ = { X@/e/ <- app(X@/e/, X@/e'/) }
                               + Col /e/ + Col /e'/

  (* constraint closure rule *)

  ccr   X@a <- app(X@b, X@c), X@b <- lam(/x/, /body/)
        -----------------------------------------
        X@a <- X@/body/, X@/x/ <- X@c
 end
```

- Exception analysis for ML core, parameterized by a CFA:

```
signature CFA = sig
                  lattice Env
                  lattice Fns = power /Ast.exp/
                  eqn Lam: /Ast.exp/:index * Env -> Fns
                end

analysis ExnAnal(Cfa: CFA) =
 ana
   set Exp = /Ast.exp/
   set Var = /Ast.var/
   set Exn = /Ast.exn/
   set UncaughtExns = power Exn
                      constraint
                      var = {X, P} index Var + Exp
                      rhs = var
                          | app_x(/Ast.exp/, var)
                          | app_p(/Ast.exp/, var)
                          | exn(Exn)                          : atomic
                          | minus(var, /Ast.exp/, power Exn) : atomic
                          | cap(var, /Ast.exp/, Exn)         : atomic

   (* constraint collection equation *)

   eqn Col /Ast.Var(x)/ = {}
     | Col /Ast.Const/ = {}
     | Col /Ast.Lam(x,e)/ = Col /e/
     | Col /e as Ast.Fix(f,x,e',e'')/ = Col /e'/ + Col /e''/
                                      + { X@/e/ <- X@/e''/, P@/e/ <- P@/e''/ }
     | Col /e as Ast.Con(e',k)/ = Col /e'/
                               + { X@/e/ <- exn(/k/), P@/e/ <- P@/e'/ }
     | Col /e as Ast.Decon(e')/ = Col /e'/
```

```
                              + { X@/e/ <- X@/e'/, P@/e/ <- P@/e'/ }
    | Col /e as Ast.Exn(k,e')/ = Col /e'/
                              + { X@/e/ <- exn /k/, X@/e/ <- X@/e'/ }
    | Col /e as Ast.App(e',e'')/ = Col /e'/ + Col /e''/
                              + { X@/e/ <- app_x(/e'/, X@/e''/),
                                  P@/e/ <- app_p(/e'/, X@/e''/),
                                  P@/e/ <- P@/e'/, P@/e/ <- P@/e''/ }
    | Col /e as Ast.Case(e',k,e'',e''')/ =
              Col /e'/ + Col /e''/ + Col /e'''/
              + { X@/e/ <- X@/e''/, X@/e/ <- X@/e'''/ }
              + { P@/e/ <- P@/e'/, P@/e/ <- P@/e''/, P@/e/ <- P@/e'''/ }
    | Col /e as Ast.Raise(e')/ = Col /e'/ + { P@e <- X@/e'/ }
    | Col /e as Ast.Mraise(e',Ks)/ =
              let
                val K = /Ast.list2set Ks/
              in
                Col /e'/ + { P@e <- minus(X@/e'/,/e'/, K) }
              end
    | Col /e as Ast.Praise(e', k)/ =
              Col /e'/ + { P@/e/ <- cap(X@/e'/,/e'/,/k/) }
    | Col /e as Ast.Handle(e', f as Ast.Lam(x,e''))/ =
              Col /e'/ + Col /e''/
              + { X@/e/ <- X@/e'/, X@/e/ <- app_x(/f/, P@/e'/) }
              + { X@/x/ <- P@/e'/, P@/e/ <- app_p(/f/, P@/e'/) }


  (* constraint closure rules *)

  ccr  X@a <- app_x(/e/,X@b), /Ast.Lam(x,e')/ in post Cfa.Lam@/e/
       -----------------------------------------
             X@a <- X@/e'/, X@/x/ <- X@b

   |   X@a <- app_x(/e/,P@b), /Ast.Lam(x,e')/ in post Cfa.Lam@/e/
       -----------------------------------------
            X@a <- X@/e'/, X@/x/ <- P@b

   |   P@a <- app_p(/e/,X@b), /Ast.Lam(x,e')/ in post Cfa.Lam@/e/
       -----------------------------------------
            P@a <- P@/e'/, X@/x/ <- X@b

   |   P@a <- app_p(/e/,P@b), /Ast.Lam(x,e')/ in post Cfa.Lam@/e/
       -----------------------------------------
            P@a <- P@/e'/, X@/x/ <- P@b


  (* constraint image definition *)

  cim exn(k) = {k}
    | minus(X,/e/,K) = if /Ast.exncarryexn(e)/ then X
                       else { x | x from X, not (x in K) }
    | cap(X,/e/,k) = if /Ast.exncarryexn(e)/ then X
                     else { x | x from X, x = k }
end
```

- A CTL bind:

```
query SatelliteInvariant = x: pre CP.E. (EG y: post CP.E. x <= y)
```

The "`SatelliteInvariant`" holds at a program point iff there exists ("`E`" in "`EG`") an execution path from the program point such that every ("`G`" in "`EG`") post-state along the path is larger than or equal to the pre-state of the starting program point. The pre-states and post-states associated with the program points are the results ("`CP.E`") from analysis "`CP`".

2 장

프로그램 분석의 기획

$$
\begin{array}{rrcl}
\mathit{Type} & \tau & ::= & \mathit{int} \mid \mathit{bool} \mid s \mid \ell \mid \tau_1 \times \tau_2 \mid \tau_1 \to \tau_2 \mid \mathit{power}\ \tau \mid \mathit{ty}_{nML} \mid \tau : \kappa \\
\mathit{Set} & s & ::= & \mathit{int} \mid \{\mathit{elmtidrow}\} \mid \mathit{power}\ s \\
 & & \mid & s_1 \times s_2 \mid s_1 + s_2 \mid s_1 \mapsto s_2 \\
 & & \mid & \mathit{tylongid}_{nML} \\
\mathit{Lattice} & \ell & ::= & \mathit{flat}\ s \mid \mathit{ordered}\ s \mid \mathit{power}\ s \\
 & & \mid & \ell_1 \times \ell_2 \mid \ell_1 + \ell_2 \mid \ell_1 \mapsto \ell_2 \mid s \mapsto \ell \\
 & & \mid & \mathit{strlongid}_{nML} \\
\mathit{Kind} & \kappa & ::= & \mathit{index} \mid \mathit{syntree} \mid \mathit{integer} \mid \mathit{power} \mid \mathit{sum} \mid \mathit{product} \mid \mathit{arrow} \mid \cdot
\end{array}
$$

$$
\begin{array}{rcrcl}
 & & \mathit{Pre} & = & \mathit{Type} \cup \{\cdot\} \\
 & & \mathit{Post} & = & \mathit{Type} \\
 & & \mathit{Syntree} & = & \mathit{Set} \cup \{\cdot\} \\
 & & \mathit{Index} & = & \mathit{Set} \cup \{\cdot\} \\
p \text{ or } (s_1, s_2) & \in & \mathit{Pivot} & = & \mathit{Syntree} \times \mathit{Index} \\
(\tau_1, \tau_2, p) & \in & \mathit{EqnType} & = & \mathit{Pre} \times \mathit{Post} \times \mathit{Pivot} \\
\mathit{VE} & \in & \mathit{VarEnv} & = & \mathit{VarId} \xrightarrow{\text{fin}} \mathit{Type} \cup \mathit{EqnType} \\
\mathit{CE} & \in & \mathit{CnstEnv} & = & \mathit{CvarEnv} \times \mathit{ConEnv} \\
\mathit{CV} & \in & \mathit{CvarEnv} & = & \mathit{CvarId} \xrightarrow{\text{fin}} \mathit{Set} \cup \mathit{Set} \times \mathit{Index} \\
\mathit{CN} & \in & \mathit{ConEnv} & = & \mathit{ConId} \xrightarrow{\text{fin}} \mathit{Type} \\
\mathit{SE} & \in & \mathit{SetEnv} & = & \mathit{SetId} \xrightarrow{\text{fin}} \mathit{Set} \cup \mathit{Kind} \\
\mathit{LE} & \in & \mathit{LatEnv} & = & \mathit{LatId} \xrightarrow{\text{fin}} \mathit{Lattice} \cup \mathit{Kind} \\
E \text{ or } (\mathit{VE}, \mathit{SE}, \mathit{LE}, \mathit{CE}) & \in & \mathit{Env} & = & \mathit{VarEnv} \times \mathit{SetEnv} \times \mathit{LatEnv} \times \mathit{CnstEnv} \\
 & & & & \\
\mathit{AE} & \in & \mathit{AnaEnv} & = & \mathit{AnaId} \xrightarrow{\text{fin}} \mathit{Env} \\
\mathit{GE} & \in & \mathit{SigEnv} & = & \mathit{SigId} \xrightarrow{\text{fin}} \mathit{Env} \\
\mathit{TE} & \in & \mathit{TemEnv} & = & \mathit{TemId} \xrightarrow{\text{fin}} \mathit{ParamEnv} \times \mathit{Env} \\
 & & \mathit{ParamEnv} & = & \cup_{k \geq 1} (\mathit{AnaId} \times \mathit{Env})^k \\
C & \in & \mathit{Context} & = & \mathit{AnaEnv} \times \mathit{SigEnv} \times \mathit{TemEnv} \times \mathit{Env}
\end{array}
$$

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

$\langle\rangle$: 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. 예를들어,

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

는 다음의 두가지 규칙을 의미한다:

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

$\mathrm{Dom}\, f, \mathrm{Ran}\, f$ "$\mathrm{Dom}\, f$"과 "$\mathrm{Ran}\, f$"은 각각 함수 f가 정의된 집합과 f의 결과 집합을 의미한다.

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

$f + g$: "$f + x \mapsto y$"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.

A of B: "A of B" 는 B가 $(\cdots, A, \cdots)$일때 A를 뜻한다.

$\tau \setminus \tau'$: 곱(product) 타입 τ에서 τ'을 뺀 결과 타입을 뜻한다. 아무 타입도 남지않으면 $\cdot$을 남긴다.

$\tau' \in \tau$: 곱(product) 타입 τ에서 τ'을 부품으로 가지고 있으면 참, 아니면 거짓이다.

$C_\beta(\alpha longid)$:

$$\begin{array}{rcl} C_\beta(\alpha id) & = & (\beta \text{ of } (E \text{ of } C))(\alpha id) \\ C_\beta(anaid.\alpha id) & = & (\beta \text{ of } (AE \text{ of } C)(anaid))(\alpha id) \end{array}$$

$Kind(\tau)$:

$$\begin{array}{rclrcl} Kind(power\ \tau) & = & power & Kind(\tau_1 \times \tau_2) & = & product \\ Kind(\tau_1 + \tau_2) & = & sum & Kind(\tau_1 \mapsto \tau_2) & = & arrow \\ Kind(int) & = & integer & Kind(\tau : \kappa) & = & \kappa \end{array}$$

프로그램 분석 정의

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

$$\frac{C \vdash anadec \Rightarrow AE}{C \vdash anadec \Rightarrow AE \text{ in } Context} \tag{2.1}$$

$$\frac{C \vdash sigdec \Rightarrow GE}{C \vdash sigdec \Rightarrow GE \text{ in } Context} \tag{2.2}$$

$$\frac{C \vdash temdec \Rightarrow TE}{C \vdash temdec \Rightarrow TE \text{ in } Context} \tag{2.3}$$

$$\frac{C \vdash topdec_1 \Rightarrow C_1 \quad C + C_1 \vdash topdec_2 \Rightarrow C_2}{C \vdash topdec_1\ topdec_2 \Rightarrow C_1 + C_2} \tag{2.4}$$

분석기 선언

$$\boxed{C \vdash anadec \Rightarrow AE}$$

$$\frac{C \vdash anaexp \Rightarrow E}{C \vdash \texttt{analysis}\ anaid\ \texttt{=}\ anaexp \Rightarrow \{anaid \mapsto E\}} \tag{2.5}$$

분석기 정의식

$$\boxed{C \vdash anaexp \Rightarrow E}$$

$$\frac{C \vdash adec \Rightarrow E}{C \vdash \texttt{ana}\ adec\ \texttt{end} \Rightarrow E} \tag{2.6}$$

$$\frac{\begin{array}{c} TE(temid) = (((anaid_1, E_1'), \cdots, (anaid_n, E_n')), E) \\ \forall i.C \vdash anaexp_i \Rightarrow E_i \quad \forall i.E_i : E_i' \end{array}}{C \vdash temid\ \texttt{(}\ anaexp_1, \cdots, anaexp_n\ \texttt{)} \Rightarrow E} \tag{2.7}$$

분석기 타입 선언

$$\boxed{C \vdash sigdec \Rightarrow GE}$$

$$\frac{C \vdash sigexp \Rightarrow E}{C \vdash \texttt{signature}\ sigid\ \texttt{=}\ sigexp \Rightarrow \{sigid \mapsto E\}} \tag{2.8}$$

분석기 타입식

$$\boxed{C \vdash sigexp \Rightarrow E}$$

$$\frac{C \vdash adesc \Rightarrow E}{C \vdash \texttt{sig}\ adesc\ \texttt{end} \Rightarrow E} \tag{2.9}$$

$$\frac{GE(sigid) = E}{C \vdash sigid \Rightarrow E} \tag{2.10}$$

분석기 타입식 내용

$$\boxed{C \vdash adesc \Rightarrow E}$$

$$\frac{C \vdash setdesc \Rightarrow E}{C \vdash \texttt{set}\ setdesc \Rightarrow E} \tag{2.11}$$

$$\frac{C \vdash \mathit{latdesc} \Rightarrow E}{C \vdash \texttt{lattice}\ \mathit{latdesc} \Rightarrow E} \tag{2.12}$$

$$\frac{C \vdash \mathit{ty} \Rightarrow \tau}{C \vdash \texttt{val}\ \mathit{varid} : \mathit{ty} \Rightarrow \{\mathit{varid} \mapsto \tau\} \text{ in } \mathit{Env}} \tag{2.13}$$

$$\frac{C \vdash \mathit{ty} \Rightarrow \tau_1 \to \tau_2 \quad C \vdash \text{air'ed kinds}(\mathit{ty}) \Rightarrow (s_1, s_2) \quad \tau_1' = \tau_1 \setminus s_1 \setminus s_2}{C \vdash \texttt{eqn}\ \mathit{varid} : \mathit{ty} \Rightarrow \{\mathit{varid} \mapsto (\tau_1', \tau_2, (s_1, s_2))\} \text{ in } \mathit{Env}} \tag{2.14}$$

$$\frac{C \vdash \mathit{ty} \Rightarrow \tau \to \mathit{bool}}{C \vdash \texttt{query}\ \mathit{ctlid} : \mathit{ty} \Rightarrow \{\mathit{ctlid} \mapsto \mathit{ty}\} \text{ in } \mathit{Env}} \tag{2.15}$$

$$\frac{C \vdash \mathit{adesc}_1 \Rightarrow E_1 \quad C + E_1 \vdash \mathit{adesc}_2 \Rightarrow E_2}{C \vdash \mathit{adesc}_1\ \mathit{adesc}_2 \Rightarrow E_1 + E_2} \tag{2.16}$$

집합의 종류 $\boxed{C \vdash \mathit{setdesc} \Rightarrow E}$

$$\frac{}{C \vdash \mathit{setid} \Rightarrow \{\mathit{setid} \mapsto \cdot\} \text{ in } \mathit{Env}} \tag{2.17}$$

$$\frac{}{C \vdash \mathit{setid} : \mathit{kind} \Rightarrow \{\mathit{setid} \mapsto \mathit{kind}\} \text{ in } \mathit{Env}} \tag{2.18}$$

$$\frac{C \vdash \mathit{setbind} \Rightarrow E}{C \vdash \mathit{setbind} \text{ as } \mathit{setdesc} \Rightarrow E} \tag{2.19}$$

래티스의 종류 $\boxed{C \vdash \mathit{latdesc} \Rightarrow E}$

$$\frac{}{C \vdash \mathit{latid} \Rightarrow \{\mathit{latid} \mapsto \cdot\} \text{ in } \mathit{Env}} \tag{2.20}$$

$$\frac{\mathit{kind} \neq \{\texttt{syntree}, \texttt{index}, \texttt{integer}\}}{C \vdash \mathit{latid} : \mathit{kind} \Rightarrow \{\mathit{latid} \mapsto \mathit{kind}\} \text{ in } \mathit{Env}} \tag{2.21}$$

$$\frac{C \vdash \mathit{latbind} \Rightarrow E}{C \vdash \mathit{latbind} \text{ as } \mathit{latdesc} \Rightarrow E} \tag{2.22}$$

분석기와 분석기 타입 매치 $\boxed{E : E'}$

$$\frac{\mathit{VE} : \mathit{VE}' \quad \mathit{SE} : \mathit{SE}' \quad \mathit{LE} : \mathit{LE}' \quad \mathit{CV} : \mathit{CV}' \quad \mathit{CN} : \mathit{CN}'}{E : E'} \tag{2.23}$$

$$\frac{\forall \mathit{varid} \in \text{Dom } \mathit{VE}'. \mathit{VE}(\mathit{varid}) = \mathit{VE}'(\mathit{varid})}{\mathit{VE} : \mathit{VE}'} \tag{2.24}$$

$$\frac{\forall setid \in \mathrm{Dom}\ SE'.SE(setid) : SE'(setid)}{SE : SE'} \tag{2.25}$$

$$\frac{\forall latid \in \mathrm{Dom}\ LE'.LE(latid) : LE'(latid)}{LE : LE'} \tag{2.26}$$

$$\frac{\forall cvarid \in \mathrm{Dom}\ CV'.CV(cvarid) = CV'(cvarid)}{CV : CV'} \tag{2.27}$$

$$\frac{\forall conid \in \mathrm{Dom}\ CN'.CN(conid) = CN'(conid)}{CN : CN'} \tag{2.28}$$

$$\frac{}{\tau : \tau} \tag{2.29}$$

$$\frac{}{\tau : \cdot} \tag{2.30}$$

$$\frac{}{\tau : Kind(\tau)} \tag{2.31}$$

(2.25,2.31) 집합을 선언할 때는 그 종류가 *index*나 *syntree*라는 것은 밝혀질 수 없으므로, 그러한 집합 종류와는 타입매치될 수 없다. 어느 집합이 그러한 종류인지는 그 집합을 이용한 분석방정식이 선언될 때 밝혀진다.

분석기 틀 선언 $\boxed{C \vdash temdec \Rightarrow TE}$

$$\frac{\begin{array}{c} C \vdash sigexp_1 \Rightarrow E_1 \quad C \vdash sigexp_2 \Rightarrow E_2 \\ C + \{anaid_1 \mapsto E_1, anaid_2 \mapsto E_2\} \vdash anaexp \Rightarrow E \end{array}}{\begin{array}{c} C \vdash \texttt{analysis}\ temid\ \texttt{(}\ anaid_1\ \texttt{:}\ sigexp_1\ \texttt{,}\ anaid_2\ \texttt{:}\ sigexp_2\ \texttt{)}\ \texttt{=}\ anaexp \Rightarrow \\ \{temid \mapsto (((anaid_1, E_1), (anaid_2, E_2)), E)\} \end{array}} \tag{2.32}$$

분석내용 선언 $\boxed{C \vdash adec \Rightarrow E}$

$$\frac{C \vdash adec_1 \Rightarrow E_1 \quad C + E_1 \vdash adec_2 \Rightarrow E_2}{C \vdash adec_1\ adec_2 \Rightarrow E_1 + E_2} \tag{2.33}$$

$$\frac{C \vdash domdec \Rightarrow E}{C \vdash domdec\ \mathrm{as}\ adec \Rightarrow E} \tag{2.34}$$

$$\frac{C \vdash semdec \Rightarrow E}{C \vdash semdec\ \mathrm{as}\ adec \Rightarrow E} \tag{2.35}$$

$$\frac{E \vdash querydec \Rightarrow VE}{C \vdash querydec\ \mathrm{as}\ adec \Rightarrow VE\ \mathrm{in}\ Env} \tag{2.36}$$

도메인 선언

$$\boxed{C \vdash domdec \Rightarrow E}$$

$$\frac{C \vdash setbind \Rightarrow E}{C \vdash \texttt{set}\ setbind \Rightarrow E} \tag{2.37}$$

$$\frac{C \vdash latbind \Rightarrow E}{C \vdash \texttt{lattice}\ latbind \Rightarrow E} \tag{2.38}$$

$$\frac{\tau = LE(latid) \quad E \vdash match \Rightarrow \tau \to \tau \text{ or } \tau \times \tau \to \tau}{C \vdash \texttt{widen}\ latid\ \texttt{with}\ match \Rightarrow \{\}} \tag{2.39}$$

$$\frac{\tau = LE(latid) \quad E \vdash match \Rightarrow \tau \to \tau \text{ or } \tau \times \tau \to \tau}{C \vdash \texttt{narrow}\ latid\ \texttt{with}\ match \Rightarrow \{\}} \tag{2.40}$$

집합의 정의

$$\boxed{C \vdash setbind \Rightarrow E}$$

$$\frac{C \vdash setexp \Rightarrow s, VE \quad setid \notin \mathrm{Dom}\ SE \cup \mathrm{Dom}\ LE}{C \vdash setid\ \texttt{=}\ setexp \Rightarrow (VE, \{setid \mapsto s\}, \{\}, \{\})} \tag{2.41}$$

$$\frac{\begin{array}{c} C \vdash setexp \Rightarrow s, VE \quad Kind(s) = power \\ C + VE, s \vdash cnstdec \Rightarrow CE \quad setid \notin \mathrm{Dom}\ SE \cup \mathrm{Dom}\ LE \end{array}}{C \vdash setid\ \texttt{=}\ setexp\ \texttt{constraint}\ cnstdec \Rightarrow (VE, \{setid \mapsto s\}, \{\}, CE)} \tag{2.42}$$

(2.41) 집합의 이름은 이미 정의된 집합이나 래티스의 이름이 아니어야한다.

래티스의 정의

$$\boxed{C \vdash latbind \Rightarrow E}$$

$$\frac{C \vdash latexp \Rightarrow \ell, VE \quad latid \notin \mathrm{Dom}\ SE \cup \mathrm{Dom}\ LE}{C \vdash latid\ \texttt{=}\ latexp \Rightarrow (VE, \{\}, \{latid \mapsto \ell\}, \{\})} \tag{2.43}$$

(2.43) 래티스의 이름은 이미 정의된 집합이나 래티스의 이름이 아니어야 한다.

집합식

$$\boxed{C \vdash setexp \Rightarrow s, VE}$$

주의: s는 집합의 속내용이다, 집합의 이름이 아니고.

$$\frac{}{C \vdash \texttt{/}tylongid\texttt{/} \Rightarrow tylongid_{nML}, \{\}} \tag{2.44}$$

$$\frac{s = C_{SE}(setlongid)}{C \vdash setlongid \Rightarrow s, \{\}} \tag{2.45}$$

$$\frac{C \vdash e_i \Rightarrow int \quad i = 1, 2}{C \vdash \texttt{\{}\ e_1\ \ldots\ e_2\ \texttt{\}} \Rightarrow int, \{\}} \tag{2.46}$$

$$\frac{\begin{array}{c}\{elmtidrow\} \cap \mathrm{Dom}\ VE = \emptyset \\ VE' = \{elmtid \mapsto \{elmtidrow\} \mid elmtid \in \{elmtidrow\}\}\end{array}}{C \vdash \texttt{\{}\ elmtidrow\ \texttt{\}} \Rightarrow \{elmtidrow\},\ VE'} \tag{2.47}$$

$$\frac{C \vdash setexp \Rightarrow s,\ VE}{C \vdash \texttt{power}\ setexp \Rightarrow power\ s,\ VE} \tag{2.48}$$

$$\frac{C \vdash setexp_1 \Rightarrow s_1,\ VE_1 \quad C + VE_1 \vdash setexp_2 \Rightarrow s_2,\ VE_2}{C \vdash setexp_1\ \texttt{*}\ setexp_2 \Rightarrow s_1 \times s_2,\ VE_1 + VE_2} \tag{2.49}$$

$$\frac{C \vdash setexp_1 \Rightarrow s_1,\ VE_1 \quad C + VE_1 \vdash setexp_2 \Rightarrow s_2,\ VE_2}{C \vdash setexp_1\ \texttt{+}\ setexp_2 \Rightarrow s_1 + s_2,\ VE_1 + VE_2} \tag{2.50}$$

$$\frac{C \vdash setexp_1 \Rightarrow s_1,\ VE_1 \quad C + VE_1 \vdash setexp_2 \Rightarrow s_2,\ VE_2}{C \vdash setexp_1\ \texttt{->}\ setexp_2 \Rightarrow s_1 \mapsto s_2,\ VE_1 + VE_2} \tag{2.51}$$

$$\frac{C \vdash setexp \Rightarrow s,\ VE}{C \vdash \texttt{(}\ setexp\ \texttt{)} \Rightarrow s,\ VE} \tag{2.52}$$

(2.46) 정수의 구간 부분집합도 정수로 한다.

(2.47) 집합의 원소들로 쓰이는 이름들은 새로운 이름들이어야 한다.

제약식 집합

$\boxed{C, s \vdash cnstdec \Rightarrow CE}$

$$\frac{\begin{array}{c}C \vdash setexp \Rightarrow s', \{\} \quad CV' \stackrel{\text{let}}{=} \{\forall i. cvarid_i \mapsto (s, s')\} \\ \mathrm{Dom}\ CV \cap \{cvaridrow\} = \{\} \quad C + CV', s \vdash rhs \Rightarrow CN\end{array}}{C, s \vdash \texttt{var = \{}\ cvaridrow\ \texttt{\} index}\ setexp\ \texttt{rhs = }\ rhs \Rightarrow (CV', CN)} \tag{2.53}$$

$$\frac{CV' \stackrel{\text{let}}{=} \{\forall i. cvarid_i \mapsto s\} \quad \mathrm{Dom}\ CV \cap \{cvaridrow\} = \{\} \quad C + CV', s \vdash rhs \Rightarrow CN}{C, s \vdash \texttt{var = \{}\ cvaridrow\ \texttt{\} rhs = }\ rhs \Rightarrow (CV', CN)} \tag{2.54}$$

(2.53,2.54) 제약식 변수들($cvaridrow$)은 모두 달라야 한다.

제약식의 오른팔 선언

$\boxed{C, s \vdash rhs \Rightarrow CN}$

$$\frac{\langle SE(setlongid) = s \rangle}{C, s \vdash \texttt{var}\ \langle setlongid \rangle \Rightarrow \{\}} \tag{2.55}$$

$$\frac{}{C, s \vdash conid\ \langle \texttt{: atomic} \rangle \Rightarrow \{conid \mapsto s\}} \tag{2.56}$$

$$\frac{C, s \vdash carg \Rightarrow \tau_1}{C, s \vdash conid\ carg\ \langle \texttt{: atomic} \rangle \Rightarrow \{conid \mapsto \tau_1 \to s\}} \tag{2.57}$$

$$\frac{C, s \vdash rhs_1 \Rightarrow CN_1 \quad C, s \vdash rhs_2 \Rightarrow CN_2 \quad \mathrm{Dom}\ CN_1 \cap \mathrm{Dom}\ CN_2 = \emptyset}{C, s \vdash rhs_1\ \texttt{|}\ rhs_2 \Rightarrow CN_1 + CN_2} \tag{2.58}$$

(2.58) 제약식에 사용되는 함수심볼들은 모두 달라야 한다.

제약식 함수심볼의 변수들 $\boxed{C, s \vdash carg \Rightarrow \tau}$

$$\frac{}{C, s \vdash \texttt{var} \Rightarrow s} \tag{2.59}$$

$$\frac{SE(setlongid) = s'}{C, s \vdash \texttt{var}\ setlongid \Rightarrow s'} \tag{2.60}$$

$$\frac{C \vdash setexp \Rightarrow s', \{\}}{C, s \vdash setexp\ as\ carg \Rightarrow s'} \tag{2.61}$$

$$\frac{C, s \vdash carg_1 \Rightarrow s_1 \quad C \vdash carg_2 \Rightarrow s_2}{C, s \vdash \texttt{(}\ carg_1\ \texttt{,}\ carg_2\ \texttt{)} \Rightarrow s_1 \times s_2} \tag{2.62}$$

(2.61) 제약식 함수심볼의 선언에 쓰는 집합식은 새로운 환경(VE)을 만들어내지 않아야 한다. (for convenience, not must)

래티스 식 $\boxed{C \vdash latexp \Rightarrow \ell, VE}$

주의: ℓ는 래티스의 속내용이다, 래티스의 이름이 아니고.

$$\frac{}{C \vdash \texttt{/}strlongid\texttt{/} \Rightarrow strlongid_{nML}, \{\}} \tag{2.63}$$

$$\frac{\ell = C_{LE}(latlongid)}{C \vdash latid \Rightarrow \ell, \{\}} \tag{2.64}$$

$$\frac{C \vdash setexp \Rightarrow s, VE}{C \vdash \texttt{flat}\ setexp \Rightarrow flat\ s, VE} \tag{2.65}$$

$$\frac{C \vdash setexp \Rightarrow s, VE}{C \vdash \texttt{power}\ setexp \Rightarrow power\ s, VE} \tag{2.66}$$

$$\frac{\begin{array}{c} C \vdash setexp \Rightarrow s, VE \quad C \vdash match_1 \Rightarrow s \times s \rightarrow s \\ \langle C \vdash match_2 \Rightarrow s \times s \rightarrow s \rangle \quad isLattice(s, match_1 \langle, match_2 \rangle) \end{array}}{C \vdash setexp\ \texttt{with join/meet}\ match_1 \langle \texttt{meet/join}\ match_2 \rangle \Rightarrow ordered\ s, VE} \tag{2.67}$$

$$\frac{C \vdash latexp_1 \Rightarrow \ell_1, VE_1 \quad C + VE_1 \vdash latexp_2 \Rightarrow \ell_2, VE_2}{C \vdash latexp_1\ \texttt{*}\ latexp_2 \Rightarrow \ell_1 \times \ell_2, VE_1 + VE_2} \tag{2.68}$$

$$\frac{C \vdash \mathit{latexp}_1 \Rightarrow \ell_1,\, VE_1 \quad C + VE_1 \vdash \mathit{latexp}_2 \Rightarrow \ell_2,\, VE_2}{C \vdash \mathit{latexp}_1 \texttt{ + } \mathit{latexp}_2 \Rightarrow \ell_1 + \ell_2,\, VE_2} \tag{2.69}$$

$$\frac{C \vdash \mathit{latexp}_1 \Rightarrow \ell_1,\, VE_1 \quad C + VE_1 \vdash \mathit{latexp}_2 \Rightarrow \ell_2,\, VE_2}{C \vdash \mathit{latexp}_1 \texttt{ -> } \mathit{latexp}_2 \Rightarrow \ell_1 \mapsto \ell_2,\, VE_2} \tag{2.70}$$

$$\frac{C \vdash \mathit{setexp} \Rightarrow s,\, VE_1 \quad C + VE_1 \vdash \mathit{latexp} \Rightarrow \ell,\, VE_2}{C \vdash \mathit{setexp} \texttt{ -> } \mathit{latexp} \Rightarrow s \mapsto \ell,\, VE_2} \tag{2.71}$$

$$\frac{C \vdash \mathit{latexp} \Rightarrow \ell,\, VE}{C \vdash \texttt{ (} \mathit{latexp} \texttt{) } \Rightarrow \ell,\, VE} \tag{2.72}$$

(2.67) $\mathit{isLattice}(s, \mathit{match}\langle, \mathit{match}\rangle)$는 사용자가 정의한 join/meet 연산($\mathit{match}$)이 래티스 조건을 만족하는 지 확인한다.

분석 식

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

$$\frac{}{C \vdash \texttt{/}\mathit{nexp}\texttt{/} \Rightarrow \mathit{ty}_{nML}} \tag{2.75}$$

$$\frac{s = C_{SE}(\mathit{setlongid})}{C \vdash \mathit{setlongid} \Rightarrow \mathit{power}\ s} \tag{2.76}$$

$$\frac{\tau = C_{VE}(\mathit{varlongid})}{C \vdash \mathit{varlongid} \Rightarrow \tau} \tag{2.77}$$

$$\frac{C_{CV}(\mathit{cvarlongid}) = (s, s') \quad C, s \vdash \mathit{rhsexp} \Rightarrow \_ \quad C \vdash \mathit{pat} \Rightarrow \_, s'}{C \vdash \mathit{cvarlongid} \texttt{ @ } \mathit{pat} \texttt{ <- } \mathit{rhsexp} \Rightarrow s} \tag{2.78}$$

$$\frac{C_{CV}(\mathit{cvarlongid}) = s \quad C, s \vdash \mathit{rhsexp} \Rightarrow \_}{C \vdash \mathit{cvarlongid} \texttt{ <- } \mathit{rhsexp} \Rightarrow s} \tag{2.79}$$

$$\frac{}{C \vdash \mathit{integer} \Rightarrow \mathit{int}} \tag{2.80}$$

$$\frac{}{C \vdash (\texttt{top}|\texttt{bottom}|\texttt{\^{}}|\texttt{\_\_}) \Rightarrow \ell} \tag{2.81}$$

$$\frac{C \vdash e_i \Rightarrow \mathit{int} \quad i = 1,2}{C \vdash e_1\ (\texttt{+}|\texttt{*}|\texttt{-})\ e_2 \Rightarrow \mathit{int}} \tag{2.82}$$

$$\frac{C \vdash e_i \Rightarrow \ell \quad i = 1,2}{C \vdash e_1\ (\texttt{+}|\texttt{*})\ e_2 \Rightarrow \ell} \tag{2.83}$$

$$\frac{C \vdash e_i \Rightarrow \mathit{power}\ \tau \quad i = 1,2}{C \vdash e_1\ (\texttt{+}|\texttt{*}|\texttt{-})\ e_2 \Rightarrow \mathit{power}\ \tau} \tag{2.84}$$

$$\frac{C \vdash e_i \Rightarrow \tau \quad i = 1,2 \qquad \mathit{ropTy}\ \tau}{C \vdash e_1\ \mathit{rop}\ e_2 \Rightarrow \mathit{bool}} \tag{2.85}$$

$$\frac{C \vdash e_i \Rightarrow \mathit{int} \quad i = 1,2}{C \vdash \texttt{\{}\ e_1\ \texttt{...}\ e_2\ \texttt{\}} \Rightarrow \mathit{power\ int}} \tag{2.86}$$

$$\frac{\forall e \in \{\mathit{erow}\}.C \vdash e \Rightarrow \tau}{C \vdash \texttt{\{}\ \mathit{erow}\ \texttt{\}} \Rightarrow \mathit{power}\ \tau} \tag{2.87}$$

$$\frac{C \vdash \mathit{qual} \Rightarrow \mathit{VE} \quad \forall e \in \{\mathit{erow}\}.C + \mathit{VE} \vdash e \Rightarrow \tau}{C \vdash \texttt{\{}\ \mathit{erow}\ \texttt{|}\ \mathit{qual}\ \texttt{\}} \Rightarrow \mathit{power}\ \tau} \tag{2.88}$$

$$\frac{\forall \mathit{mrule} \in \{\mathit{mrulerow}\}.C \vdash \mathit{mrule} \Rightarrow \tau_1 \rightarrow \tau_2 \quad \tau_1 \mapsto \tau_2 \in \mathrm{Ran}\ C_{SE} \cup \mathrm{Ran}\ C_{LE}}{C \vdash \texttt{\{}\ \mathit{mrulerow}\ \texttt{\}} \Rightarrow \tau_1 \mapsto \tau_2} \tag{2.89}$$

$$\frac{\begin{array}{c} C \vdash \mathit{qual} \Rightarrow \mathit{VE} \quad \tau_1 \mapsto \tau_2 \in \mathrm{Ran}\ C_{SE} \cup \mathrm{Ran}\ C_{LE} \\ \forall \mathit{mrule} \in \{\mathit{mrulerow}\}.C + \mathit{VE} \vdash \mathit{mrule} \Rightarrow \tau_1 \rightarrow \tau_2 \end{array}}{C \vdash \texttt{\{}\ \mathit{mrulerow}\ \texttt{|}\ \mathit{qual}\ \texttt{\}} \Rightarrow \tau_1 \mapsto \tau_2} \tag{2.90}$$

$$\frac{C \vdash e \Rightarrow \mathit{power\ int}/\mathit{power}\ \ell/\mathit{power\ power}\ \tau}{C \vdash (\texttt{+}|\texttt{*})\ e \Rightarrow \mathit{int}/\ell/\mathit{power}\ \tau} \tag{2.91}$$

$$\frac{C \vdash e_i \Rightarrow \tau_i \quad i = 1,2}{C \vdash \texttt{(}\ e_1\ \texttt{,}\ e_2\ \texttt{)} \Rightarrow \tau_1 \times \tau_2} \tag{2.92}$$

$$\frac{C \vdash e \Rightarrow \tau_1 \times \tau_2}{C \vdash e\ \texttt{.}\ \texttt{1} \Rightarrow \tau_1} \tag{2.93}$$

$$\frac{C \vdash e \Rightarrow \tau_1 \times \tau_2}{C \vdash e\ \texttt{.}\ \texttt{2} \Rightarrow \tau_2} \tag{2.94}$$

$$\frac{C \vdash \mathit{valdec} \Rightarrow \mathit{VE} \quad C + \mathit{VE} \vdash e \Rightarrow \tau}{C \vdash \texttt{let}\ \mathit{valdec}\ \texttt{in}\ e\ \texttt{end} \Rightarrow \tau} \tag{2.95}$$

$$\frac{C \vdash \mathit{match} \Rightarrow \tau_1 \rightarrow \tau_2}{C \vdash \texttt{fn}\ \mathit{match} \Rightarrow \tau_1 \rightarrow \tau_2} \tag{2.96}$$

$$\frac{C \vdash e_1 \Rightarrow \tau_1 \rightarrow \tau_2 \text{ or } \tau_1 \mapsto \tau_2 \quad C \vdash e_2 \Rightarrow \tau_1}{C \vdash e_1\ e_2 \Rightarrow \tau_2} \tag{2.97}$$

$$\frac{C \vdash e \Rightarrow \tau}{C \vdash \texttt{(}\ e\ \texttt{)} \Rightarrow \tau} \tag{2.98}$$

$$\frac{C \vdash e \Rightarrow \tau \quad C \vdash \mathit{ty} \Rightarrow \tau}{C \vdash e\ \texttt{:}\ \mathit{ty} \Rightarrow \tau} \tag{2.99}$$

$$\frac{C_{VE}(\mathit{varlongid}) = (\tau_1, \tau_2, (s_1, s_2)) \quad C \vdash e \Rightarrow s_2}{C \vdash \langle \texttt{pre} \rangle\ \mathit{varlongid}\ \texttt{@}\ e \Rightarrow \tau_1} \tag{2.100}$$

$$\frac{C_{VE}(\mathit{varlongid}) = (\tau_1, \tau_2, (s_1, s_2)) \quad C \vdash e \Rightarrow s_2}{C \vdash \langle \texttt{post} \rangle\ \mathit{varlongid}\ \texttt{@}\ e \Rightarrow \tau_2} \tag{2.101}$$

$$\frac{C \vdash e \Rightarrow \tau' \quad \tau \in \mathrm{Ran}\ C_{SE} \cup \mathrm{Ran}\ C_{LE} \quad \tau = \cdots + \tau' + \cdots}{C \vdash e \Rightarrow \tau} \tag{2.102}$$

$$\frac{C \vdash e \Rightarrow s \quad \mathit{flat}\ s \in \mathrm{Ran}\ C_{LE}}{C \vdash e \Rightarrow \mathit{flat}\ s} \tag{2.103}$$

(2.76) 분석식에서 *uid*는 정의된 집합의 이름이어야 한다.

(2.77) 분석식에서 *lid*는 집합의 원소이름(*elmtid*)이거나 정의된 변수(*varid*) 이름이어야 한다.

(2.87) 집합의 원소나 래티스의 원소를 모아놓을 수 있다.

(2.89,2.90) 함수테이블은 사용자가 정의한 집합이나 래티스의 원소들이어야 한다. ($\tau_1 \mapsto \tau_2 \in \mathrm{Ran}\ C_{SE} \cup \mathrm{Ran}\ C_{LE}$).

(2.102,2.103) 최소로 필요한 타입변환이 언제 어떻게 필요한 지는 타입검증 알고리즘이 알아내야 한다.

관계연산 가능한 타입

$\boxed{\mathit{ropTy}\ \tau}$

$$\frac{\tau \text{ has neither } \tau_1 \rightarrow \tau_2 \text{ nor nML type as its component.}}{\mathit{ropTy}\ \tau} \tag{2.104}$$

타입 식

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

$$\frac{}{C \vdash \texttt{int} \Rightarrow \mathit{int}} \tag{2.105}$$

$$\frac{}{C \vdash \texttt{/}\mathit{tylongid}\texttt{/} \Rightarrow \mathit{tylongid}_{nML}} \tag{2.106}$$

$$\frac{\tau = (\mathit{SE} + \mathit{LE})(\mathit{domlongid})}{C \vdash \mathit{domlongid} \Rightarrow \tau} \tag{2.107}$$

$$\frac{C \vdash \mathit{ty} \Rightarrow \tau}{C \vdash \texttt{power}\ \mathit{ty} \Rightarrow \mathit{power}\ \tau} \tag{2.108}$$

$$\frac{C \vdash \mathit{ty}_i \Rightarrow \tau_i \quad i = 1, 2}{C \vdash \mathit{ty}_1\ \texttt{->}\ \mathit{ty}_2 \Rightarrow \tau_1 \rightarrow \tau_2} \tag{2.109}$$

$$\frac{C \vdash ty_i \Rightarrow \tau_i \quad i = 1, 2}{C \vdash ty_1 \; \texttt{*} \; ty_2 \Rightarrow \tau_1 \times \tau_2} \tag{2.110}$$

$$\frac{C \vdash ty_i \Rightarrow \tau_i \quad i = 1, 2 \qquad \tau_1 + \tau_2 \in \text{Ran } C_{SE} \cup \text{Ran } C_{LE}}{C \vdash ty_1 \; \texttt{+} \; ty_2 \Rightarrow \tau_1 + \tau_2} \tag{2.111}$$

$$\frac{C \vdash ty \Rightarrow \tau \quad \mathit{flat}\,\tau \in \text{Ran } C_{LE}}{C \vdash \texttt{flat} \; ty \Rightarrow \mathit{flat}\,\tau} \tag{2.112}$$

$$\frac{C \vdash ty \Rightarrow \tau}{C \vdash \texttt{(} \; ty \; \texttt{)} \Rightarrow \tau} \tag{2.113}$$

$$\frac{C \vdash ty \Rightarrow \tau}{C \vdash ty \; \texttt{:} \; \mathit{kind} \Rightarrow \tau \; \texttt{:} \; \mathit{kind}} \tag{2.114}$$

$$\frac{C \vdash ty \Rightarrow s \quad \text{air } s \; \texttt{:} \; \mathit{index}}{C \vdash ty \; \texttt{:} \; \texttt{index} \Rightarrow s \; \texttt{:} \; \mathit{index}} \tag{2.115}$$

$$\frac{C \vdash ty \Rightarrow s \quad \text{air } s \; \texttt{:} \; \mathit{syntree}}{C \vdash ty \; \texttt{:} \; \texttt{syntree} \Rightarrow s \; \texttt{:} \; \mathit{syntree}} \tag{2.116}$$

(2.109) 함수 타입식은 함수집합/래티스($\tau_1 \mapsto \tau_2$)가 아니고 함수 계산식의 타입($\tau_1 \to \tau_2$)을 이른다.

(2.111,2.112) 합 타입은 집합이나 래티스로 정의되어 있어야 한다. Flat 래티스 타입은 래티스로 정의되어 있어야 한다.(번역의 편리를 위해서.)

바람에 실려온 힌트

$\boxed{C \vdash \text{air'ed kinds}(\star) \Rightarrow (s_1, s_2)}$

$\star$ is either e or ty.

$$\frac{s_1 \; \texttt{:} \; \mathit{syntree} \;\; s_2 \; \texttt{:} \; \mathit{index} \text{ are air'ed from } \star}{C \vdash \text{air'ed kinds}(\star) \Rightarrow (s_1, s_2)} \tag{2.117}$$

$$\frac{\text{only } s \; \texttt{:} \; \mathit{syntree} \text{ is air'ed from } \star}{C \vdash \text{air'ed kinds}(\star) \Rightarrow (s, s)} \tag{2.118}$$

$$\frac{\text{only } s \; \texttt{:} \; \mathit{index} \text{ is air'ed from } \star}{C \vdash \text{air'ed kinds}(\star) \Rightarrow (\cdot, s)} \tag{2.119}$$

$$\frac{\text{nothing is air'ed from } \star}{C \vdash \text{air'ed kinds}(\star) \Rightarrow (\cdot, \cdot)} \tag{2.120}$$

패턴 매치

$\boxed{C \vdash \mathit{match} \Rightarrow \tau_1 \to \tau_2}$

$$\frac{C \vdash \mathit{mrule} \Rightarrow \tau_1 \to \tau_2 \quad \langle C \vdash \mathit{match} \Rightarrow \tau_1 \to \tau_2 \rangle}{C \vdash \mathit{mrule} \; \langle \texttt{|} \; \mathit{match} \rangle \Rightarrow \tau_1 \to \tau_2} \tag{2.121}$$

매치 룰

$$\boxed{C \vdash mrule \Rightarrow \tau_1 \rightarrow \tau_2}$$

$$\frac{C \vdash pat \Rightarrow VE, \tau_1 \quad C + VE \vdash e \Rightarrow \tau_2}{C \vdash pat \texttt{ => } e \Rightarrow \tau_1 \rightarrow \tau_2} \tag{2.122}$$

패턴

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

$$\frac{}{C \vdash \texttt{/}npat\texttt{/} \Rightarrow \{\}, tylongid_{nML}} \tag{2.123}$$

$$\frac{}{C \vdash \texttt{\_} \Rightarrow \{\}, \tau} \tag{2.124}$$

$$\frac{}{C \vdash varid \Rightarrow \{varid \mapsto \tau\}, \tau} \tag{2.125}$$

$$\frac{C \vdash patrow \Rightarrow VE, \tau}{C \vdash \texttt{\{} \ patrow \ \langle \ldots \rangle \ \texttt{\}} \Rightarrow VE, power\ \tau} \tag{2.126}$$

$$\frac{C \vdash pat_1 \Rightarrow VE_1, int \quad C \vdash pat_2 \Rightarrow VE_2, int \quad \mathrm{Dom}\ VE_1 \cap \mathrm{Dom}\ VE_2 = \emptyset}{C \vdash \texttt{\{} \ pat_1 \ \texttt{...} \ pat_2 \ \texttt{\}} \Rightarrow VE_1 + VE_2, power\ int} \tag{2.127}$$

$$\frac{C \vdash mpatrow \Rightarrow VE, \tau_1 \mapsto \tau_2}{C \vdash \texttt{\{} \ mpatrow \ \langle \ldots \rangle \ \texttt{\}} \Rightarrow VE, \tau_1 \mapsto \tau_2} \tag{2.128}$$

$$\frac{C \vdash pat_1 \Rightarrow VE_1, \tau_1 \quad C \vdash pat_2 \Rightarrow VE_2, \tau_2 \quad \mathrm{Dom}\ VE_1 \cap \mathrm{Dom}\ VE_2 = \emptyset}{C \vdash \texttt{(} \ pat_1 \ \texttt{,} \ pat_2 \ \texttt{)} \Rightarrow VE_1 + VE_2, \tau_1 \times \tau_2} \tag{2.129}$$

$$\frac{C \vdash pat \Rightarrow VE, \tau \quad C + VE \vdash guard}{C \vdash pat \ \texttt{with} \ guard \Rightarrow VE, \tau} \tag{2.130}$$

$$\frac{C \vdash pat_1 \Rightarrow VE, \tau \quad C \vdash pat_2 \Rightarrow VE, \tau}{C \vdash pat_1 \ \texttt{or} \ pat_2 \Rightarrow VE, \tau} \tag{2.131}$$

$$\frac{C \vdash pat \Rightarrow VE, \tau}{C \vdash varid \ \texttt{as} \ pat \Rightarrow VE + \{varid \mapsto \tau\}, \tau} \tag{2.132}$$

$$\frac{C \vdash pat \Rightarrow VE, \tau \quad C \vdash ty \Rightarrow \tau}{C \vdash pat \ \texttt{:} \ ty \Rightarrow VE, \tau} \tag{2.133}$$

$$\frac{C \vdash pat \Rightarrow VE, \tau}{C \vdash \texttt{(} \ pat \ \texttt{)} \Rightarrow VE, \tau} \tag{2.134}$$

$$\frac{C \vdash pat \Rightarrow VE, \tau' \quad \tau \in \mathrm{Ran}\ C_{SE} \cup \mathrm{Ran}\ C_{LE} \quad \tau = \cdots + \tau' + \cdots}{C \vdash pat \Rightarrow VE, \tau} \tag{2.135}$$

$$\frac{C \vdash pat \Rightarrow VE, s \quad \mathit{flat}\ s \in \mathrm{Ran}\ C_{LE}}{C \vdash pat \Rightarrow VE, \mathit{flat}\ s} \tag{2.136}$$

(2.135,2.136) 패턴의 타입 변환은 원래의 VE를 변화시키지 않는다. 최소로 필요한 타입변환이 언제 어떻게 필요한 지는 타입검증 알고리즘이 알아내야 한다.

패턴들

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

$$\frac{C \vdash pat \Rightarrow VE, \tau \quad C \vdash patrow \Rightarrow VE', \tau \quad \mathrm{Dom}\ VE \cap \mathrm{Dom}\ VE' = \emptyset}{C \vdash pat\ \texttt{,}\ patrow \Rightarrow VE + VE', \tau} \tag{2.137}$$

함수 패턴

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

$$\frac{C \vdash pat_1 \Rightarrow VE_1, \tau_1 \quad C \vdash pat_2 \Rightarrow VE_2, \tau_2 \quad \mathrm{Dom}\ VE_1 \cap \mathrm{Dom}\ VE_2 = \emptyset}{C \vdash pat_1\ \texttt{=>}\ pat_2 \Rightarrow VE_1 + VE_2, \tau_1 \mapsto \tau_2} \tag{2.138}$$

함수 패턴들

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

$$\frac{C \vdash mpat \Rightarrow VE, \tau \quad C \vdash mpatrow \Rightarrow VE', \tau \quad \mathrm{Dom}\ VE \cap \mathrm{Dom}\ VE' = \emptyset}{C \vdash mpat\ \texttt{,}\ mpatrow \Rightarrow VE + VE', \tau} \tag{2.139}$$

집합 원소의 자격

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

$$\frac{C \vdash gen \Rightarrow VE \quad \langle C + VE \vdash guard \rangle}{C \vdash gen \langle\ \texttt{,}\ guard \rangle \Rightarrow VE} \tag{2.140}$$

집합 원소의 소속

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

$$\frac{C \vdash pat \Rightarrow VE, \tau \quad C \vdash e \Rightarrow power\ \tau}{C \vdash pat\ \texttt{from}\ e \Rightarrow VE} \tag{2.141}$$

$$\frac{C \vdash mpat \Rightarrow VE, \tau_1 \mapsto \tau_2 \quad C \vdash e \Rightarrow \tau_1 \mapsto \tau_2}{C \vdash mpat\ \texttt{from}\ e \Rightarrow VE} \tag{2.142}$$

$$\frac{C \vdash gen_1 \Rightarrow VE_1 \quad C \vdash gen_2 \Rightarrow VE_2 \quad \mathrm{Dom}\ VE_1 \cap \mathrm{Dom}\ VE_2 = \emptyset}{C \vdash gen_1\ \texttt{,}\ gen_2 \Rightarrow VE_1 + VE_2} \tag{2.143}$$

집합 원소의 제한

$\boxed{C \vdash guard}$

$$\frac{C \vdash e_i \Rightarrow \tau \quad i = 1, 2 \quad ropTy\ \tau}{C \vdash e_1\ rop\ e_2} \tag{2.144}$$

$$\frac{C \vdash e_1 \Rightarrow \tau \quad C \vdash e_2 \Rightarrow \mathit{power}\ \tau}{C \vdash e_1\ \texttt{in}\ e_2} \tag{2.145}$$

$$\frac{C \vdash \mathit{guard}}{C \vdash \texttt{not}\ \mathit{guard}} \tag{2.146}$$

$$\frac{C \vdash \mathit{guard}_1 \quad C \vdash \mathit{guard}_2}{C \vdash \mathit{guard}_1\ (\texttt{and}|\texttt{or})\ \mathit{guard}_2} \tag{2.147}$$

$$\frac{C \vdash \mathit{gen} \Rightarrow \mathit{VE} \quad C + \mathit{VE} \vdash \mathit{guard}}{C \vdash \texttt{!}\ \mathit{gen}\ \texttt{.}\ \mathit{guard}} \tag{2.148}$$

$$\frac{C \vdash \mathit{gen} \Rightarrow \mathit{VE} \quad C + \mathit{VE} \vdash \mathit{guard}}{C \vdash \texttt{?}\ \mathit{gen}\ \texttt{.}\ \mathit{guard}} \tag{2.149}$$

$$\frac{C \vdash \mathit{guard}}{C \vdash \texttt{(}\ \mathit{guard}\ \texttt{)}} \tag{2.150}$$

(2.144) $\mathit{ropTy}\ \tau$는 타입 τ가 계산함수타입 (함수집합/함수래티스가 아닌)을 포함하고 있지 않고, nML 타입이 아니어야한다.

분석기 선언

$\boxed{C \vdash \mathit{semdec} \Rightarrow E}$

$$\frac{C \vdash \mathit{valdec} \Rightarrow \mathit{VE}}{C \vdash \mathit{valdec} \Rightarrow \mathit{VE}\ \text{in}\ \mathit{Env}} \tag{2.151}$$

$$\frac{C \vdash \mathit{eqndec} \Rightarrow \mathit{VE}}{C \vdash \mathit{eqndec} \Rightarrow \mathit{VE}\ \text{in}\ \mathit{Env}} \tag{2.152}$$

$$\frac{C \vdash \mathit{ccrdec}}{C \vdash \mathit{ccrdec} \Rightarrow \{\}} \tag{2.153}$$

분석값 선언

$\boxed{C \vdash \mathit{valdec} \Rightarrow \mathit{VE}}$

$$\frac{C \vdash \mathit{vbind} \Rightarrow \mathit{VE}}{C \vdash \texttt{val}\ \mathit{vbind} \Rightarrow \mathit{VE}} \tag{2.154}$$

$$\frac{C + \mathit{VE} \vdash \mathit{vbind} \Rightarrow \mathit{VE}}{C \vdash \texttt{val rec}\ \mathit{vbind} \Rightarrow \mathit{VE}} \tag{2.155}$$

분석값 정의

$\boxed{C \vdash \mathit{vbind} \Rightarrow \mathit{VE}}$

$$\frac{C \vdash \mathit{pat} \Rightarrow \mathit{VE}, \tau \quad C \vdash e \Rightarrow \tau \quad \langle C \vdash \mathit{vbind} \Rightarrow \mathit{VE}' \quad \text{Dom}\ \mathit{VE} \cap \text{Dom}\ \mathit{VE}' = \emptyset \rangle}{C \vdash \mathit{pat}\ \texttt{=}\ e\ \langle \texttt{and}\ \mathit{vbind} \rangle \Rightarrow \mathit{VE}\ \langle + \mathit{VE}' \rangle} \tag{2.156}$$

분석 방정식 선언

$\boxed{C \vdash \mathit{eqndec} \Rightarrow \mathit{VE}}$

$$\frac{C \vdash \mathit{ebind} \Rightarrow \mathit{VE}}{C \vdash \texttt{eqn}\ \mathit{ebind} \Rightarrow \mathit{VE}} \tag{2.157}$$

$$\frac{C \vdash \mathit{ebind} \Rightarrow \mathit{VE}}{C + \mathit{VE} \vdash \texttt{eqn rec}\ \mathit{ebind} \Rightarrow \mathit{VE}} \tag{2.158}$$

분석 방정식 정의

$\boxed{C \vdash \mathit{ebind} \Rightarrow \mathit{VE}}$

$$\frac{C \vdash e \Rightarrow \tau \quad \langle C \vdash \mathit{ebind} \Rightarrow \mathit{VE} \quad \mathit{varid} \notin \mathrm{Dom}\ \mathit{VE}\rangle}{C \vdash \mathit{varid}\ \texttt{=}\ e\ \langle \texttt{and}\ \mathit{ebind}\rangle \Rightarrow \{\mathit{varid} \mapsto \tau\}\ \langle +\mathit{VE}\rangle} \tag{2.159}$$

$$\frac{\begin{array}{c} C \vdash e \Rightarrow \tau_1 \to \tau_2 \quad C \vdash \text{air'ed kinds}(e) \Rightarrow (s_1, s_2) \quad s_1 \in \tau_1 \quad s_2 \in \tau_1 \\ \langle C \vdash \mathit{ebind} \Rightarrow \mathit{VE} \quad \mathit{varid} \notin \mathrm{Dom}\ \mathit{VE}\rangle \end{array}}{C \vdash \mathit{varid}\ \texttt{=}\ e\ \langle \texttt{and}\ \mathit{ebind}\rangle \Rightarrow \{\mathit{varid} \mapsto (\tau_1, \tau_2, (s_1, s_2))\}\ \langle +\mathit{VE}\rangle} \tag{2.160}$$

제약식 푸는 규칙

$\boxed{C \vdash \mathit{ccrdec}}$

$$\frac{C \vdash \mathit{ccrbind}}{C \vdash \texttt{ccr}\ \mathit{ccrbind}} \tag{2.161}$$

제약식 푸는 규칙 정의

$\boxed{C \vdash \mathit{ccrbind}}$

$$\frac{C \vdash \mathit{cnstguard} \Rightarrow \mathit{VE} \quad \forall i. C \vdash \mathit{constraint}_i \Rightarrow \_ \quad \langle C \vdash \mathit{ccrbind}\rangle}{C \vdash \texttt{ccr}\ \mathit{cnstguard}\ \texttt{--}^{+}\ \mathit{constraintrow}\ \langle \texttt{|}\ \mathit{ccrbind}\rangle} \tag{2.162}$$

제약식 또는 조건

$\boxed{C \vdash \mathit{cnstguard} \Rightarrow \mathit{VE}}$

$$\frac{C \vdash \mathit{constraint} \Rightarrow \mathit{VE}}{C \vdash \mathit{constraint}\ \text{as}\ \mathit{cnstguard} \Rightarrow \mathit{VE}} \tag{2.163}$$

$$\frac{C \vdash \mathit{guard}}{C \vdash \mathit{guard}\ \text{as}\ \mathit{cnstguard} \Rightarrow \{\}} \tag{2.164}$$

$$\frac{C \vdash \mathit{cnstguard}_1 \Rightarrow \mathit{VE}_1 \quad C + \mathit{VE}_1 \vdash \mathit{cnstguard}_2 \Rightarrow \mathit{VE}_2}{C \vdash \mathit{cnstguard}_1\ \texttt{,}\ \mathit{cnstguard}_2 \Rightarrow \mathit{VE}_1 + \mathit{VE}_2} \tag{2.165}$$

제약식 하나

$\boxed{C \vdash \mathit{constraint} \Rightarrow \mathit{VE}}$

$$\frac{\mathit{CV}(\mathit{cvarlongid}) = (s, s') \quad C, s \vdash \mathit{rhsexp} \Rightarrow \mathit{VE} \quad C \vdash \mathit{pat} \Rightarrow \_, s'}{C \vdash \mathit{cvarlongid}\ \texttt{@}\ \mathit{pat}\ \texttt{<-}\ \mathit{rhsexp} \Rightarrow \mathit{VE}} \tag{2.166}$$

$$\frac{CV(cvarlongid) = s \quad C, s \vdash rhsexp \Rightarrow VE}{C \vdash cvarlongid \texttt{ <- } rhsexp \Rightarrow VE} \tag{2.167}$$

제약식의 오른팔식 $\boxed{C, s \vdash rhsexp \Rightarrow VE}$

$$\frac{CV(cvarlongid) = s}{C, s \vdash cvarlongid \Rightarrow \{\}} \tag{2.168}$$

$$\frac{CV(cvarlongid) = (s, s') \quad C \vdash pat \Rightarrow VE, s'}{C, s \vdash cvarlongid \texttt{ @ } pat \Rightarrow VE} \tag{2.169}$$

$$\frac{CN(conlongid) = s}{C, s \vdash conlongid \Rightarrow \{\}} \tag{2.170}$$

$$\frac{CN(conlongid) = \tau \to s \quad C, \tau \vdash cargexp \Rightarrow VE}{C, s \vdash conlongid\ cargexp \Rightarrow VE} \tag{2.171}$$

제약식 함수심볼의 인자식 $\boxed{C, \tau \vdash cargexp \Rightarrow VE}$

$$\frac{CV(cvarlongid) = (\tau, s) \quad \langle C \vdash pat \Rightarrow \_, s \rangle}{C, \tau \vdash cvarlongid\ \langle \texttt{@}\ pat \rangle \Rightarrow \{\}} \tag{2.172}$$

$$\frac{C \vdash pat \Rightarrow VE, \tau}{C, \tau \vdash pat \Rightarrow VE} \tag{2.173}$$

$$\frac{C, \tau \vdash cargexp \Rightarrow VE}{C, \tau \vdash \texttt{(}\ cargexp\ \texttt{)} \Rightarrow VE} \tag{2.174}$$

$$\frac{C, \tau_1 \vdash cargexp_1 \Rightarrow VE \quad C, \tau_2 \vdash cargexp_2 \Rightarrow VE'}{C, \tau_1 \times \tau_2 \vdash \texttt{(}\ cargexp_1 \texttt{,}\ cargexp_2\ \texttt{)} \Rightarrow VE + VE'} \tag{2.175}$$

제약식 함수심볼의 이미지 $\boxed{C \vdash cimdec}$

$$\frac{C \vdash cimbind}{C \vdash \texttt{cim}\ cimbind} \tag{2.176}$$

제약식 함수심볼의 이미지 정의 $\boxed{C \vdash cimbind}$

$$\frac{CN(conlongid) = \tau \to s \quad C \vdash pat \Rightarrow VE, \tau \quad C + VE \vdash e \Rightarrow s \quad \langle C \vdash cimbind \rangle}{C \vdash conlongid\ pat \texttt{ = } e\ \langle \texttt{|}\ C \vdash cimbind \rangle} \tag{2.177}$$

$$\frac{CN(conlongid) = s \quad C + VE \vdash e \Rightarrow s \quad \langle C \vdash cimbind \rangle}{C \vdash \texttt{cim}\ conlongid \texttt{ = } e\ \langle \texttt{|}\ C \vdash cimbind \rangle} \tag{2.178}$$

분석 결과의 탐색

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

$$\frac{C + VE \vdash ctlbind \Rightarrow VE}{C \vdash \texttt{query}\ ctlbind \Rightarrow VE} \tag{2.179}$$

(2.179) 탐색식이 재귀적으로 정의될 수 있다.

탐색의 정의

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

$$\frac{C \vdash ctl \Rightarrow \tau \quad \langle C \vdash ctlbind \Rightarrow VE' \rangle}{C \vdash ctlid\ \texttt{=}\ ctl\ \langle \texttt{and}\ ctlbind \rangle \Rightarrow \{ctlid \mapsto \tau\}\ \langle + VE' \rangle} \tag{2.180}$$

탐색 함수

$\boxed{C \vdash ctl \Rightarrow \ell \to bool}$

$$\frac{VE(varlongid_2) = \tau \quad E + \{varid_1 \mapsto \tau\} \vdash form/guard}{C \vdash varid_1\ \texttt{:}\ varlongid_2\ \texttt{.}\ form/guard \Rightarrow \tau \to bool} \tag{2.181}$$

$$\frac{VE(varlongid_2) = (\tau_1, \tau_2, p) \quad \tau' = \tau_1 \setminus p \quad E + \{varid_1 \mapsto \tau'\} \vdash form/guard}{C \vdash varid_1\ \texttt{:}\ \texttt{pre}\ varlongid_2\ \texttt{.}\ form/guard \Rightarrow \tau' \to bool} \tag{2.182}$$

$$\frac{VE(varlongid_2) = (\tau_1, \tau_2, p) \quad E + \{varid_1 \mapsto \tau_2\} \vdash form/guard}{C \vdash varid_1\ \texttt{:}\ \texttt{post}\ varlongid_2\ \texttt{.}\ form/guard \Rightarrow \tau_2 \to bool} \tag{2.183}$$

$$\frac{C \vdash ctl \Rightarrow \tau \to bool}{C \vdash \texttt{(}\ ctl\ \texttt{)} \Rightarrow \tau \to bool} \tag{2.184}$$

탐색 식

$\boxed{C \vdash form}$

$$\frac{VE(ctlid) = \tau \to bool \quad VE(varid) = \tau}{C \vdash ctlid\ varid} \tag{2.185}$$

$$\frac{C \vdash form}{C \vdash \texttt{not}\ form} \tag{2.186}$$

$$\frac{C \vdash form_1 \quad C \vdash form_2}{C \vdash form_1\ (\texttt{and}|\texttt{or}|\texttt{->})\ form_2} \tag{2.187}$$

$$\frac{C \vdash ctl \Rightarrow \_}{C \vdash (\texttt{A}|\texttt{E})(\texttt{X}|\texttt{F}|\texttt{G})\ ctl} \tag{2.188}$$

$$\frac{C \vdash ctl_1 \Rightarrow \_ \quad C \vdash ctl_2 \Rightarrow \_}{C \vdash (\texttt{A}|\texttt{E})\texttt{U}\ \texttt{(}\ ctl_1\ \texttt{,}\ ctl_2\ \texttt{)}} \tag{2.189}$$

$$\frac{C \vdash form}{C \vdash \texttt{(}\ form\ \texttt{)}} \tag{2.190}$$

2.1 요점정리: 프로그램 분석식의 올바른 기획

2.1.1 기본 분석기 일반: 집합제약식과 결과탐색식 관련 제외

In Rabbit, user-definable sets and lattices are:

$$
\begin{array}{lcl}
s & ::= & \mathit{int} \mid \{\mathit{elmtidrow}\} \mid \mathit{power}\ s \mid s_1 \times s_2 \mid s_1 + s_2 \mid s_1 \mapsto s_2 \mid \mathit{tylongid}_{nML} \\
\ell & ::= & \mathit{flat}\ s \mid \mathit{ordered}\ s \mid \mathit{power}\ s \mid \ell_1 \times \ell_2 \mid \ell_1 + \ell_2 \mid \ell_1 \mapsto \ell_2 \mid s \mapsto \ell \mid \mathit{strlongid}_{nML}
\end{array}
$$

For a given Rabbit program, let S be the set of user-defined sets and L be the set of user-defined lattices. S and L are finite and fixed. When we write s or ℓ, we mean elements of the fixed sets S or L for a given Rabbit program. We call "domains" for s or ℓ.

Every Rabbit expression has a unique mono-morphic type:

$$
\begin{array}{lcll}
\tau & ::= & \mathit{int} \mid \mathit{bool} & \\
 & \mid & s & \text{user-defined set} \\
 & \mid & \ell & \text{user-defined lattice} \\
 & \mid & \tau \times \tau & \text{product} \\
 & \mid & \tau \to \tau & \text{function} \\
 & \mid & 2^{\tau} & \text{collection} \\
 & \mid & \mathit{ty}_{nML} & \text{nML type}
\end{array}
$$

Above definition imposes the condition that expression's sum-types(+) and map-types($\mapsto$) are restricted to the user-defined sets and lattices.

Simplified core Rabbit syntax is:

$$
\begin{array}{lcl}
a & ::= & \texttt{eqn}\ x = e,\ \cdots,\ x = e \\
e & ::= & c \mid x \mid \lambda x.e \mid \delta x.e \mid e\ e \mid e\texttt{[}e\texttt{=>}e\texttt{]} \\
 & \mid & e : \tau \mid \texttt{case}\ e\ p\ e\ p\ e \mid \texttt{(}e\texttt{,}e\texttt{)} \mid e\texttt{.}i \\
 & \mid & \texttt{\{}e\texttt{,}e\texttt{\}} \mid e\texttt{!}e \mid e\texttt{+}e \mid e\texttt{*}e \mid e\texttt{-}e \mid \texttt{+}e \mid \texttt{*}e \\
 & \mid & e \sqsubseteq e \mid \texttt{if}\ e\ e\ e \mid \texttt{/}\mathit{nexp}\texttt{/} \\
c & ::= & \bot \mid \top \mid z \mid \mathit{elmtid} \mid \texttt{true} \mid \texttt{false} \\
p & ::= & c \mid x \mid \texttt{\_} \mid \texttt{(}p\texttt{,}p\texttt{)} \mid \texttt{\{}p\texttt{,}p\texttt{\}} \mid p : \tau \mid \texttt{/}\mathit{npat}\texttt{/}
\end{array}
$$

- $\lambda x.e$ is a function, has type $\tau_1 \to \tau_2$.
- $\delta x.e$ is a map (an element of a function domain), has type s or ℓ (user-defined domain) of either $s_1 \to s_2$, $s \to \ell$, or $\ell_1 \to \ell_2$.
- $e_1\ e_2$ is overloaded for both function and map application. Hence e_1 can be a function or an element of a function domain.
- $e\texttt{[}e_1\texttt{=>}e_2\texttt{]}$ is for a change to a map, not a computation function, i.e., e's type is a function domain.
- $\texttt{(}e_1\texttt{,}e_2\texttt{)}$ is a pair, has type $\tau_1 \times \tau_2$. The product type is not necessarily a user-defined domain.

- `+`,`*`,`-` are overloaded: `+` for set union, lattice join, and integer addition. `*` for set intersection, lattice meet, and integer multiplication. `-` for set minus and integer subtraction.

- `{`e`,`e`}` is for a collection, has type 2^τ. The collection type is not necessarily a user-defined domain.

- `+`e or `*`e are for folding by `+` and `*` for elements in e.

- e_1`!`e_2 is the collection of the results from applying the function or map e_1 to every elements in collection e_2.

- $e : \tau$ is for type annotation: e's type is τ.

- Type casting is implicit (rules 2.102,2.103,2.135,2.136) and one-directional. Possible type castings are either

  – from a set to its user-defined flat lattice (s to *flat* s), or

  – from a domain to a sum domain that contains it (τ to $(\cdots+\tau+\cdots)$).

  No implicit casting is possible for the reverse direction: neither from a sum domain into its component domain nor from a flat lattice into its base set. Such castings must be explicitly done only by the case expressions with type annotations in patterns. (See examples below.)

  Given Rabbit programs, optimal number of type castings are automatically derived by Rabbit's type-checking algorithm. Type casting happens in expressions and patterns.

  For example, consider the following part of a Rabbit program:

```
set Age = {0...200}
lattice L = flat Age

fun add ^ = 200
  | add __ = 0
  | add x:Age = x + 1
```

  The `add` function can have two types: $L{\rightarrow}int$ and $L{\rightarrow}L$. It can have $L{\rightarrow}int$ with the following type castings marked by subscripts:

$$\begin{array}{l}\texttt{fun add } \hat{}_L \texttt{ = 200}\\ \quad\texttt{| add } \_\_{}_L \texttt{ = 0}\\ \quad\texttt{| add (x:Age)}_L \texttt{ = x + 1}\end{array}$$

  It can have $L{\rightarrow}L$ as:

$$\begin{array}{l}\texttt{fun add } \hat{}_L \texttt{ = 200}_L\\ \quad\texttt{| add } \_\_{}_L \texttt{ = 0}_L\\ \quad\texttt{| add (x:Age)}_L \texttt{ = (x + 1)}_L\end{array}$$

or as:

$$\begin{array}{l}
\texttt{fun add \^{}}_{L}\texttt{ = 200}_{L}\\
\texttt{\ \ | add -{}-}_{L}\texttt{ = 0}_{L}\\
\texttt{\ \ | add (x:Age)}_{L}\texttt{ = x}_{L}\texttt{ + 1}_{L}
\end{array}$$

Among the multiple typings, the contexts where the function `add` is used in the Rabbit program must determine its unique typing.

For another example, consider:

```
set Int = /int/
set Limit = {--,++}
set Z = Int + Limit

fun widen (x: Int) = if x >= 10 then ++ else x
  | widen (y: Limit) = y

fun add (++,x) = ++
  | add (x,++) = ++
  | add (--,x) = --
  | add (x,--) = --
  | add (x,y) = x+y
```

The `widen` and `add` respectively have type $Z \rightarrow Z$ and $Z \times Z \rightarrow Z$, with the help of type castings marked by subscripts:

$$\begin{array}{l}
\texttt{fun widen (x:Int)}_{Z}\texttt{ = if x >= 10 then ++}_{Z}\texttt{ else x}_{Z}\\
\texttt{\ \ | widen (y:Limit)}_{Z}\texttt{ = y}_{Z}\\
\\
\texttt{fun add (++}_{Z}\texttt{, x}_{Z}\texttt{) = ++}_{Z}\\
\texttt{\ \ | add (x}_{Z}\texttt{, ++}_{Z}\texttt{) = ++}_{Z}\\
\texttt{\ \ | add (-{}-}_{Z}\texttt{, x}_{Z}\texttt{) = -{}-}_{Z}\\
\texttt{\ \ | add (x}_{Z}\texttt{, -{}-}_{Z}\texttt{) = ++}_{Z}\\
\texttt{\ \ | add (x}_{Int}\texttt{, y}_{Int}\texttt{)}_{Z\times Z}\texttt{ = (x+y)}_{Z}
\end{array}$$

2.1.2 Rabbit 프로그램안에 있는 nML

Inside Rabbit, nML terms can be included with delimiting slashes around them: `/` $-$ `/`. nML's type and structure identifiers can be used in set and lattice declarations. nML's expressions and patterns can be used inside Rabbit expressions.

- An nML's type name can be a Rabbit's set s. An nML's structure name can be a Rabbit's lattice ℓ.
- Any nML term can be included inside Rabbit programs and its nML type is the type in Rabbit. Rabbit's type-checking system does not check the type-safety of the embedded nML terms.

- Because *int*, *bool*, and their products and functions are common types in both nML and Rabbit, inter-operation between nML and Rabbit must be via the common-typed values.

3 장

프로그램 분석기로의 변환

Rabbit 프로그램(프로그램 분석식)의 의미는 분석할 프로그램들에서 분석된 결과들을 결정해 주는 함수가 된다.