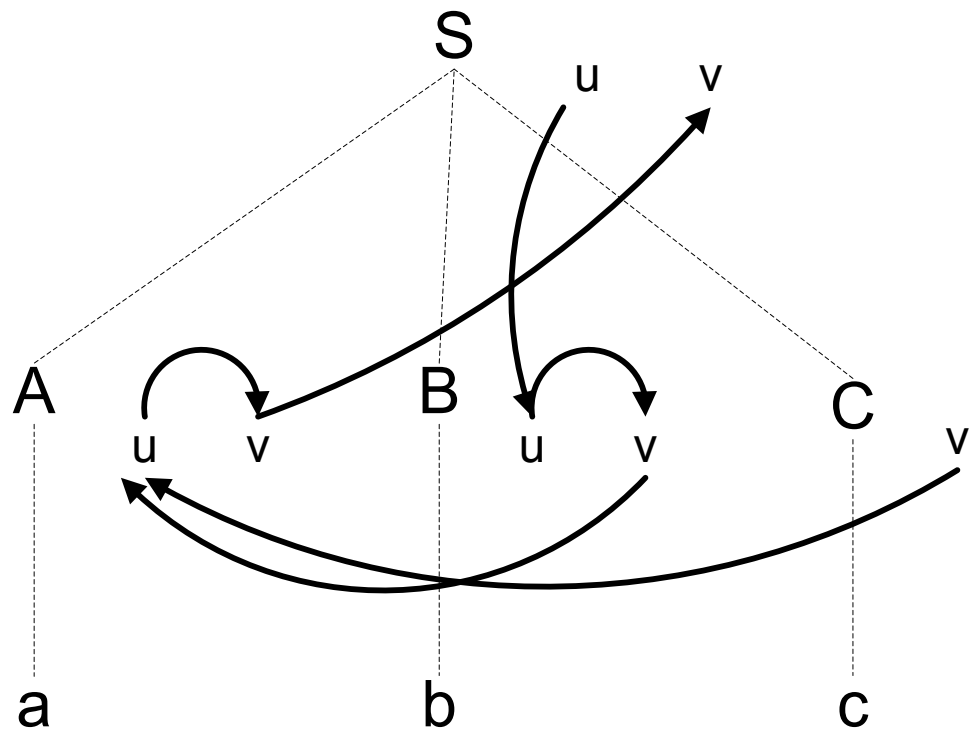
Grammar Rule	Semantic Rule
$exp_1 \rightarrow exp_2$ **+** *term*	exp_1.*postfix* = exp_2.*postfix* || exp_2.*postfix* || +
$exp_1 \rightarrow exp_2$ **-** *term*	exp_1.*postfix* = exp_2.*postfix* || exp_2.*postfix* || -
exp → *term*	*exp.postfix* = *term.postfix*
$term_1 \rightarrow term_2$ ***** *factor*	$term_1$.*postfix* = $term_2$. *postfix* || *factor.postfix* ||*
term → *factor*	*term.postfix* = *factor.postfix*
factor → **(** *exp* **)**	*factor.postfix* = *exp.postfix*
factor → **number**	*factor.postfix* = **number**.*strval*

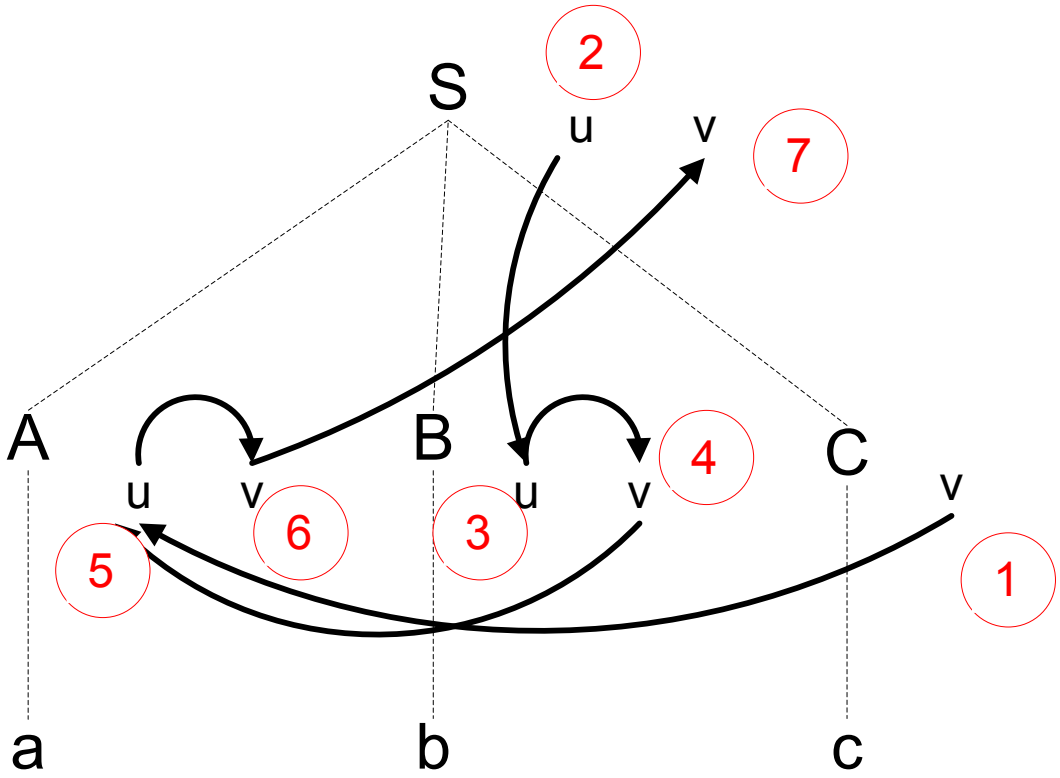
INF 5110 - 2014

Grammar Rule	Semantic Rule
decl → *var-list* : *type*	*var-list.dtype* = *type.dtype*
$var\text{-}list_1$ → $var\text{-}list_2$, **id**	$var\text{-}list_2.dtype$ = $var\text{-}list_1.dtype$ **id**.*dtype* = $var\text{-}list_1.dtype$
var-list → **id**	**id**.*dtype* = *var-list.dtype*
type → **int**	*type.dtype* = *integer*
type → **real**	*type.dtype* = *real*

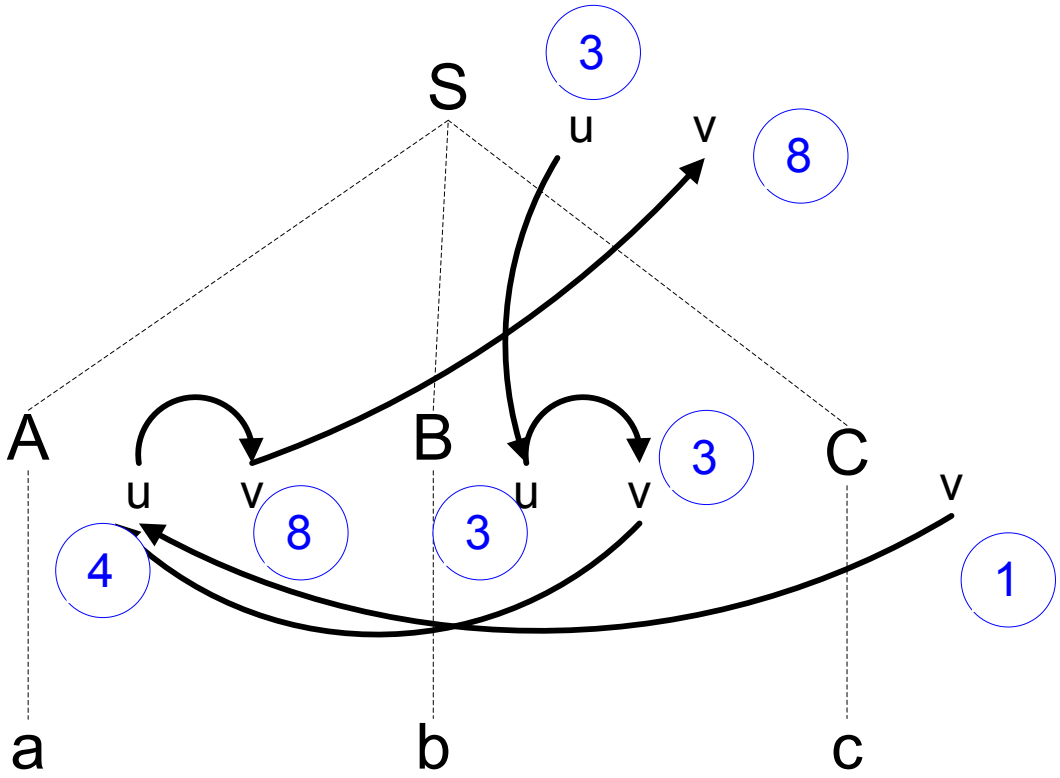
Oppgave 6.13 a 1)



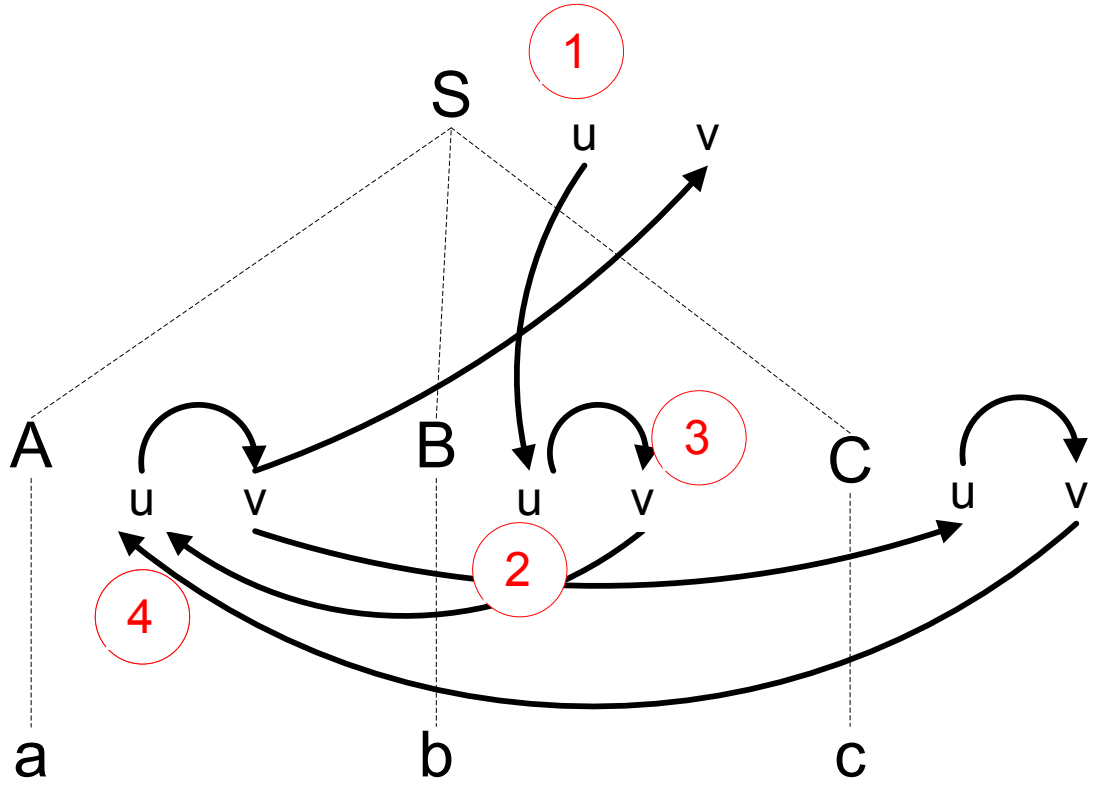
Oppgave 6.13 a 2)



Oppgave 6.13 b)



Oppgave 6.13 c



Grammar Rule	Semantic Rule
class → ***class name*** **{** *decls* **}**	*decls*.*enclosingClassName* = ***name***.name
$decls_1$ → $decls_2$ **;** *decl*	$decls_2$.*enclosingClassName* = $decls_1$.*enclosingClassName* *decl*.*enclosingClassName* = $decls_1$.*enclosingClassName*
decls → *decl*	*decl*.*enclosingClassName* = *decls*.*enclosingClassName*
decl → *variable-decl*	
decl → *method-decl*	*method-decl*.*enclosingClassName* = *decl*.*enclosingClassName*
type → **int**	*type*.type = int
type → **bool**	*type*.type = bool
type → **void**	*type*.type = void

Grammar Rule	Semantic Rule
method-decl → *type* **name** **(** *params* **)** *body*	**if** (**name**.name = *method-decl.enclosingClassName*) **then if** (**not**(*type*.type = void))**then** *error*(“*constructor not of type void*”) Eller **if** (**name**.name = *method-decl.enclosingClassName*) **and** (**not**(*type*.type = void))**then** *error*(“*constructor not of type void*”)

Grammar Rule	Semantic Rule
function-decl → *type* ***id*** **(** **)** *body*	*function-decl.has_parameter = no*
function-decl → *type* ***id*** **(** *parameter* **)** *body*	*function-decl.has_parameter = yes* *function-decl.param-kind = parameter.kind* *function-decl.param-type = parameter.type*
parameter → *type* ***id***	*parameter.kind = var* *parameter.type = type.type*
parameter → *type* **func** ***id***	*parameter.kind = func* *parameter.type = type.type*
type → **int**	*type.type = integer*
type → **bool**	*type.type = boolean*
type → **void**	*type.type = void*

Grammar Rule	Semantic Rule
call → ***id*** *()*	*call.ok =* *(lookup(**id**.name).has_parameter=no)*
call → $\mathbf{id}_1$*(*$\mathbf{id}_2$*)*	*call.ok =* *(lookup(*$\mathbf{id}_1$*.name).has_parameter=yes)* *and* *(lookup(*$\mathbf{id}_2$*.name).kind=* *(lookup(*$\mathbf{id}_1$*.name).param-kind)* *and* *(lookup(*$\mathbf{id}_2$*.name).type=* *(lookup(*$\mathbf{id}_1$*.name).param-type)* *and* *(if lookup(*$\mathbf{id}_2$*.name).kind=func then* *(lookup(*$\mathbf{id}_2$*.name).has_parameter=no) else true)*

Grammar Rule	Semantic Rule
func → *type* **func** *id signature stmt-list*	*stmt-list*.type = *type*.type
type → **int**	*type*.type = Integer
type → **bool**	*type*.type = Boolean
$stmt\text{-}list_1 \rightarrow stmt\text{-}list_2\ stmt$	$stmt\text{-}list_2$.type = $stmt\text{-}list_1$.type *stmt*.type = $stmt\text{-}list_1$.type
stmt-list → *stmt*	*stmt*.type = *stmt-list*.type
stmt → *return-stmt*	*return-stmt*.type = *stmt*.type
return-stmt → **return** *exp*	*return-stmt*.ok = (*return-stmt*.type = *exp*.type)

Grammar Rule	Semantic Rule
exp → *id*	*exp*.type = lookup(*id*.*name*)
exp → id_1 **+** id_2	*exp*.type = if lookup(id_1.*name*)= Integer and lookup(id_2.*name*)= Integer then Integer else ErrorType
exp → **true**	*exp*.type = Boolean
exp → **false**	*exp*.type = Boolean