# Oppgave 6.5

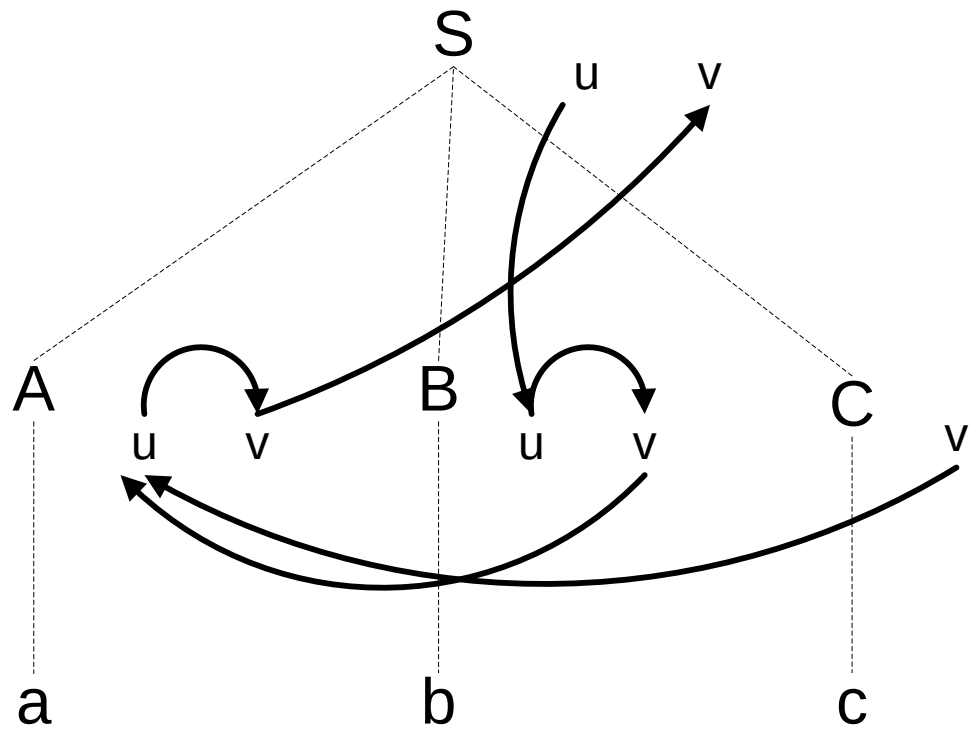
| Grammar Rule | Semantic Rule |
|---|---|
| $exp_1 \rightarrow exp_2$ **+** $term$ | $exp_1.postfix =$ $exp_2.postfix \mid\mid exp_2.postfix \mid\mid +$ |
| $exp_1 \rightarrow exp_2$ **-** $term$ | $exp_1.postfix =$ $exp_2.postfix \mid\mid exp_2.postfix \mid\mid -$ |
| $exp \rightarrow term$ | $exp.postfix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

INF 5110 - 2011

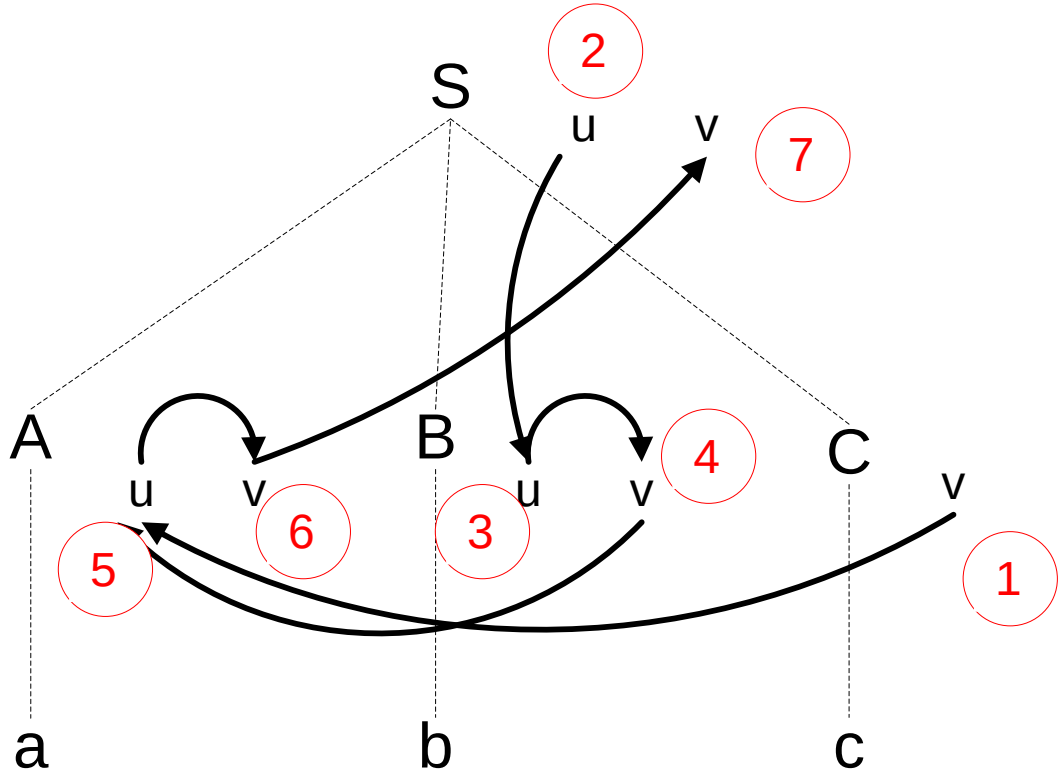
# Oppgave 6.7

| Grammar Rule | Semantic Rule |
|---|---|
| *decl* → *var-list* : *type* | *var-list.dtype* = *type.dtype* |
| $var\text{-}list_1$ → $var\text{-}list_2$ , **id** | $var\text{-}list_2.dtype = var\text{-}list_1.dtype$<br>**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* |

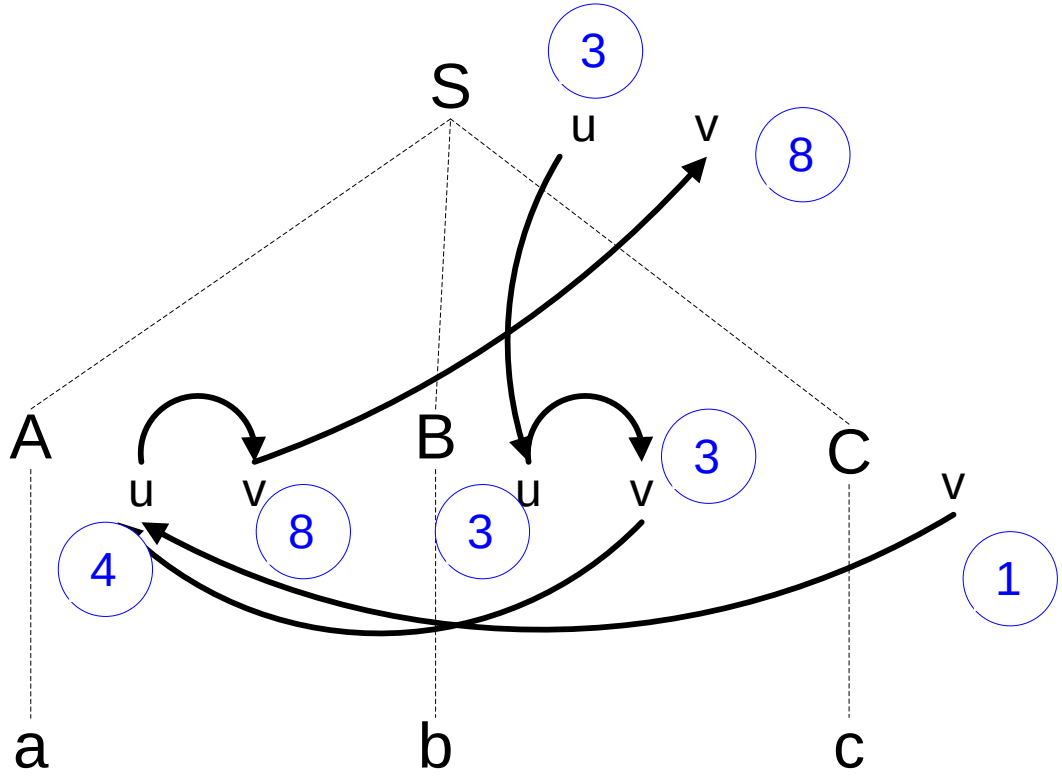
# Oppgave 6.13 a  1)



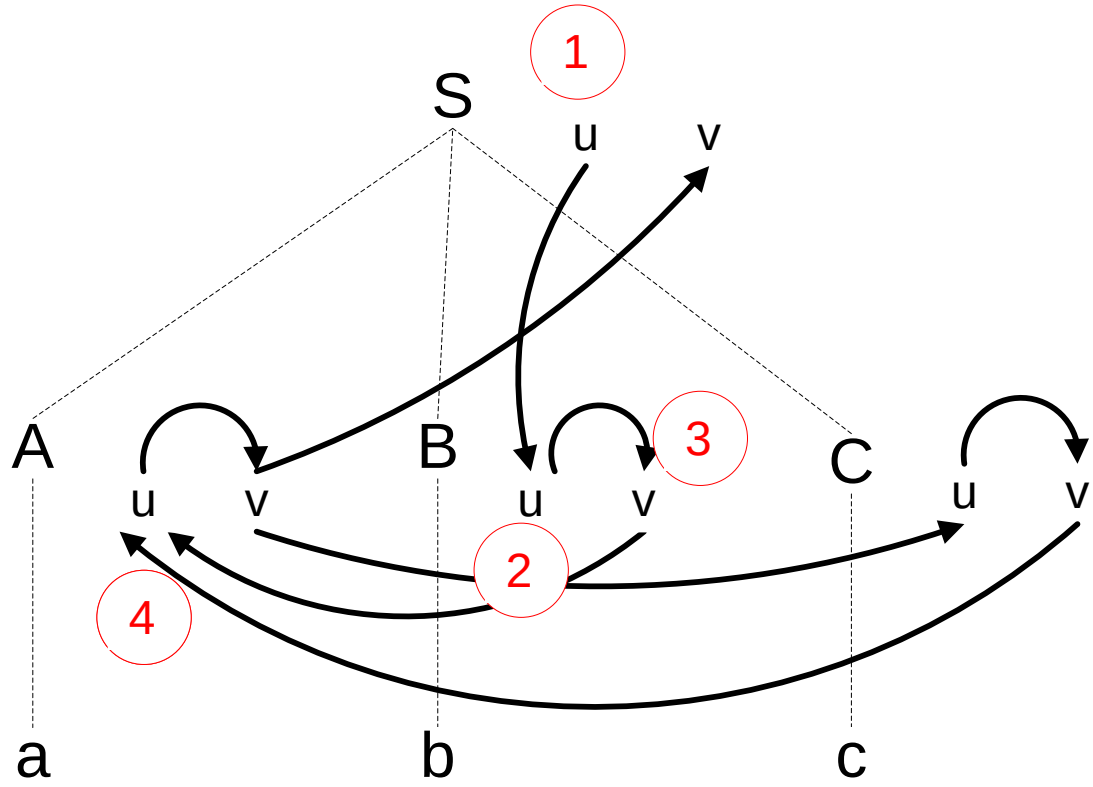
# Oppgave 6.13 a  2)



# 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*<br>*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 →*<br>*type* ***name*** ***(*** *params* ***)*** *body* | ***if*** *(****name****.*name =<br>*method-decl.enclosingClassName)*<br>***then if*** *(****not****(type.*type =<br>void*))****then*** *error(“constructor not of type void”)*<br><br>Eller<br><br>***if*** *(****name****.*name =<br>*method-decl.enclosingClassName)*<br>***and*** *(****not****(type.*type =<br>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*<br>*function-decl.param-kind = parameter.kind*<br>*function-decl.param-type = parameter.type* |
| *parameter* → *type* ***id*** | *parameter.kind = var*<br>*parameter.type = type.type* |
| *parameter* → *type* **func** ***id*** | *parameter.kind = func*<br>*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 |
| $stmt\text{-}list_1$ → *stmt*-$list_2$ *stmt* | $stmt\text{-}list_2$.type = $stmt\text{-}list_1$.type<br>*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 =<br>(*return-stmt*.type = *exp*.type) |

| Grammar Rule | Semantic Rule |
|---|---|
| $exp \rightarrow id$ | $exp$.type = lookup($id$.$name$) |
| $exp \rightarrow id_1$ **+** $id_2$ | $exp$.type =<br>if lookup($id_1$.$name$)= Integer and<br>lookup($id_2$.$name$)= Integer<br>then Integer else ErrorType |
| $exp \rightarrow$ **true** | $exp$.type = Boolean |
| $exp \rightarrow$ **false** | $exp$.type = Boolean |