Session and Session Type

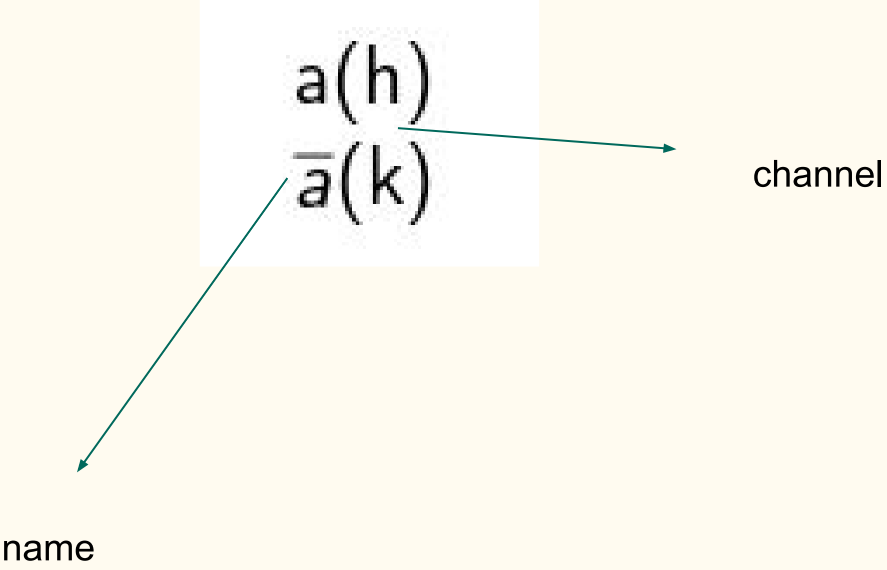
Frans Simanjuntak

Session Type

Notations and Conversion

- !
- ?
- ⊕
- &
- session name
- session channel

Session Initiation



$a(h)$

$\bar{a}(k)$

channel

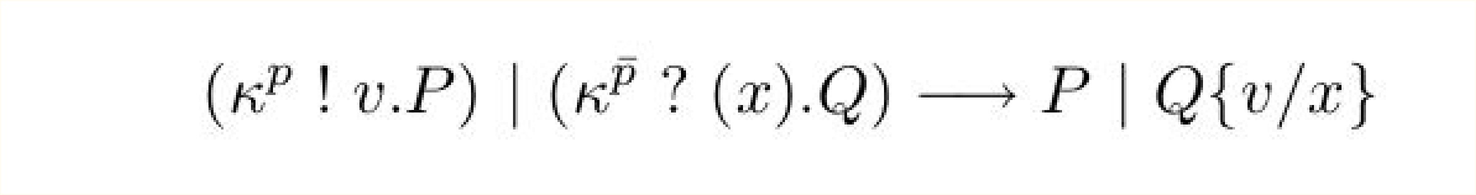
name

Rule - Session Initiation

κ $\quad p \in \{+, -\}$

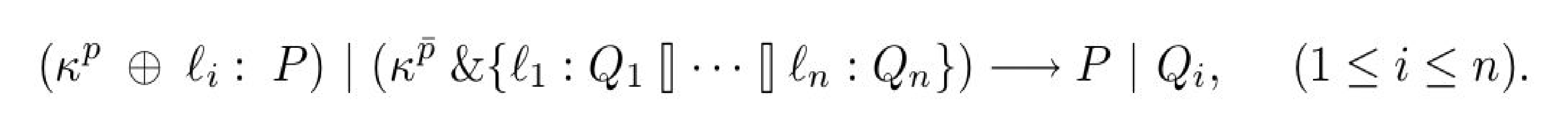
$$(\bar{a}(k).P) \mid (a(h).Q) \longrightarrow (\nu\kappa)(P\{\kappa^{+}/k\} \mid Q\{\kappa^{-}/h\}).$$

Rule - Receive and Send Value



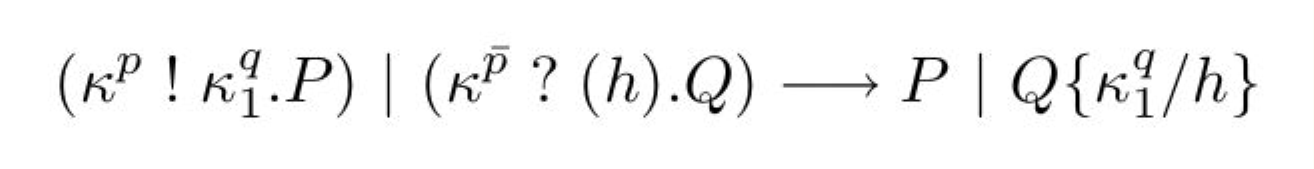
$$(\kappa^{p} \; ! \; v.P) \mid (\kappa^{\bar{p}} \; ? \; (x).Q) \longrightarrow P \mid Q\{v/x\}$$

Rule - Select / Branching



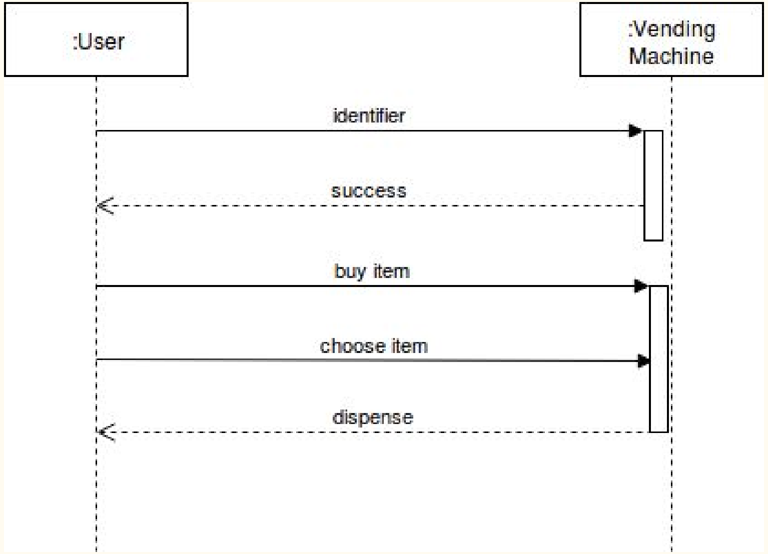
$$(\kappa^{p} \;\oplus\; \ell_i :\; P) \mid (\kappa^{\bar{p}} \,\&\{\ell_1 : Q_1 \,[\!]\, \cdots \,[\!]\, \ell_n : Q_n\}) \longrightarrow P \mid Q_i, \qquad (1 \leq i \leq n).$$

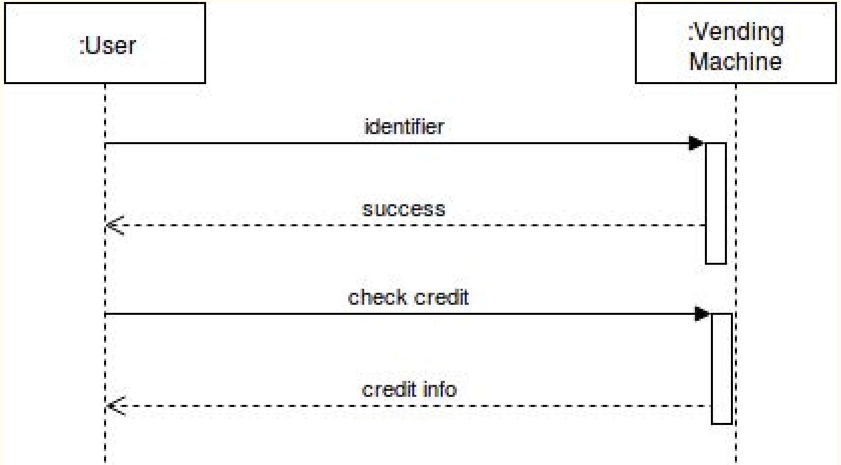
Delegation



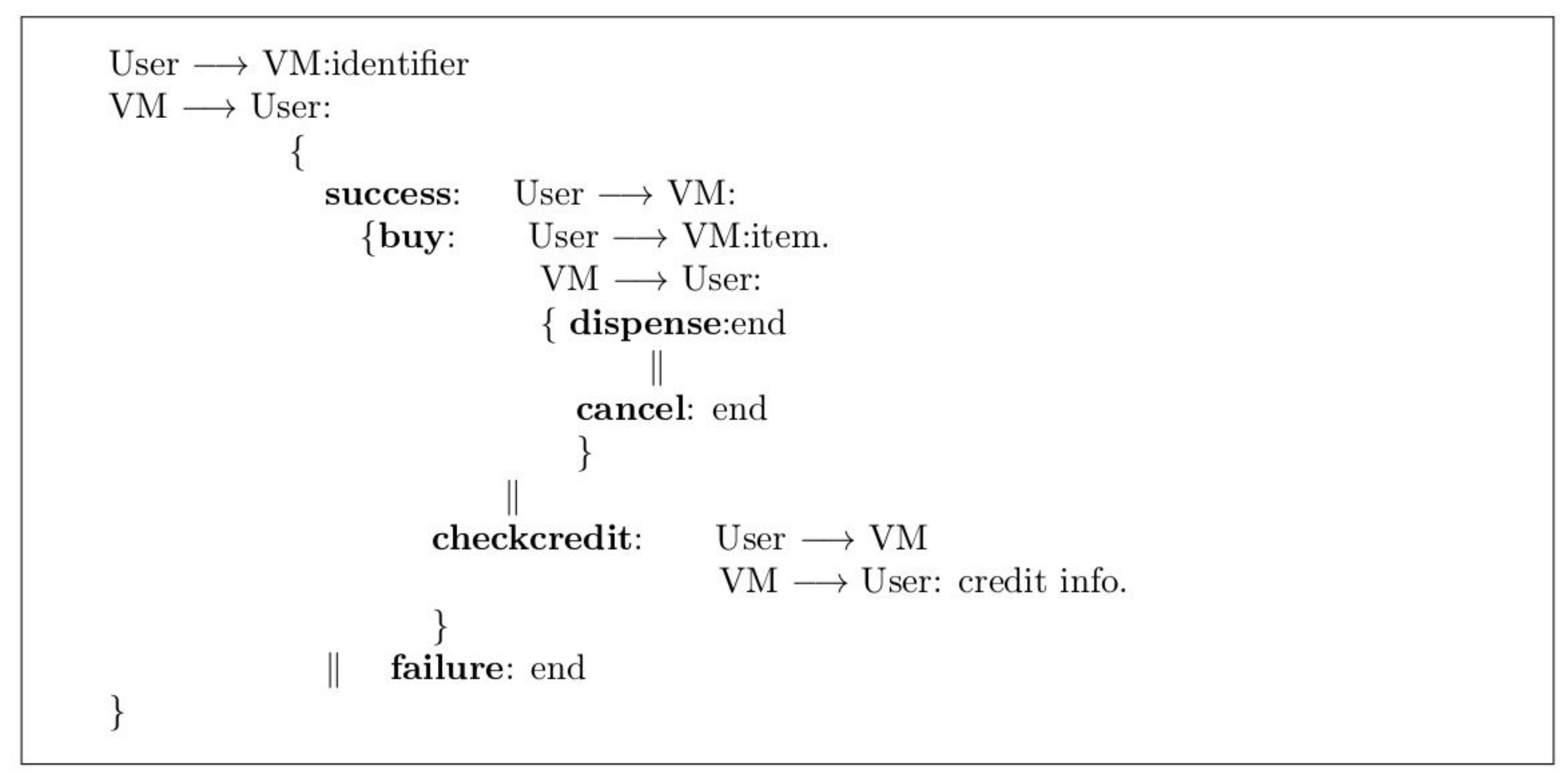
$$(\kappa^{p} \; ! \; \kappa_1^{q}.P) \mid (\kappa^{\bar{p}} \; ? \; (h).Q) \longrightarrow P \mid Q\{\kappa_1^{q}/h\}$$

User and Vending Machine



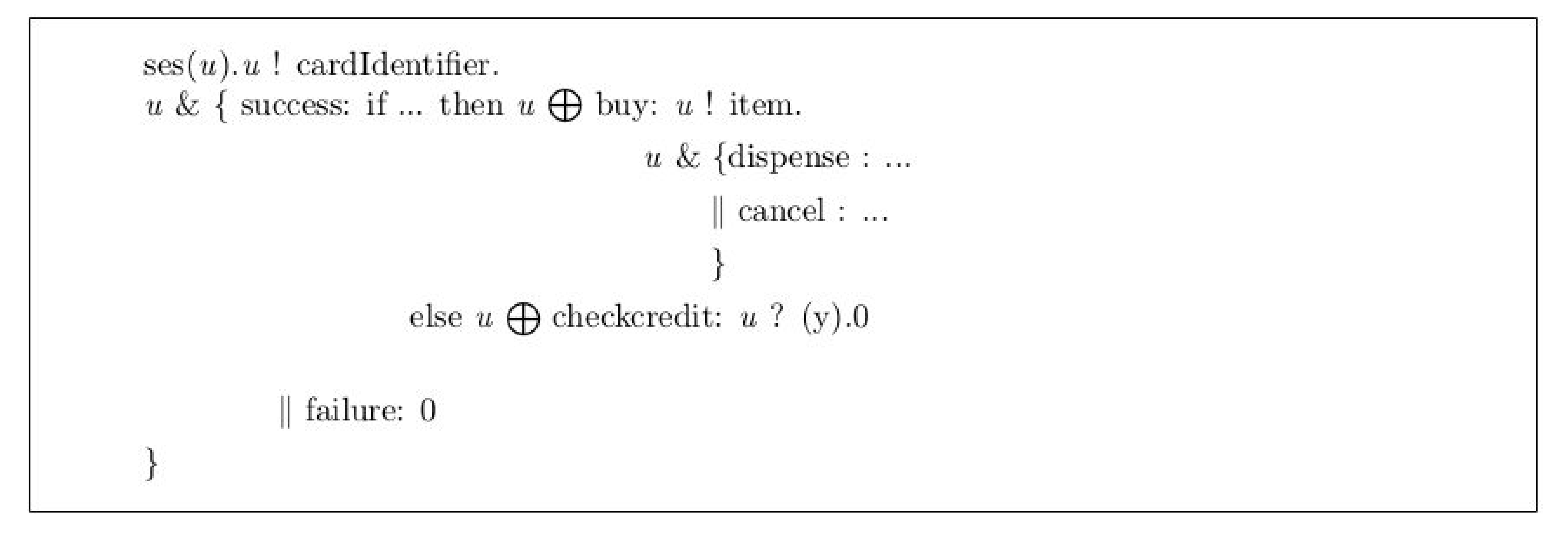


Global Description



```
User ⟶ VM:identifier
VM ⟶ User:
            {
              success:   User ⟶ VM:
                {buy:     User ⟶ VM:item.
                           VM ⟶ User:
                           { dispense:end
                                   ||
                             cancel: end
                             }
                     ||
                  checkcredit:     User ⟶ VM
                                   VM ⟶ User: credit info.
                  }
            ||   failure: end
}
```

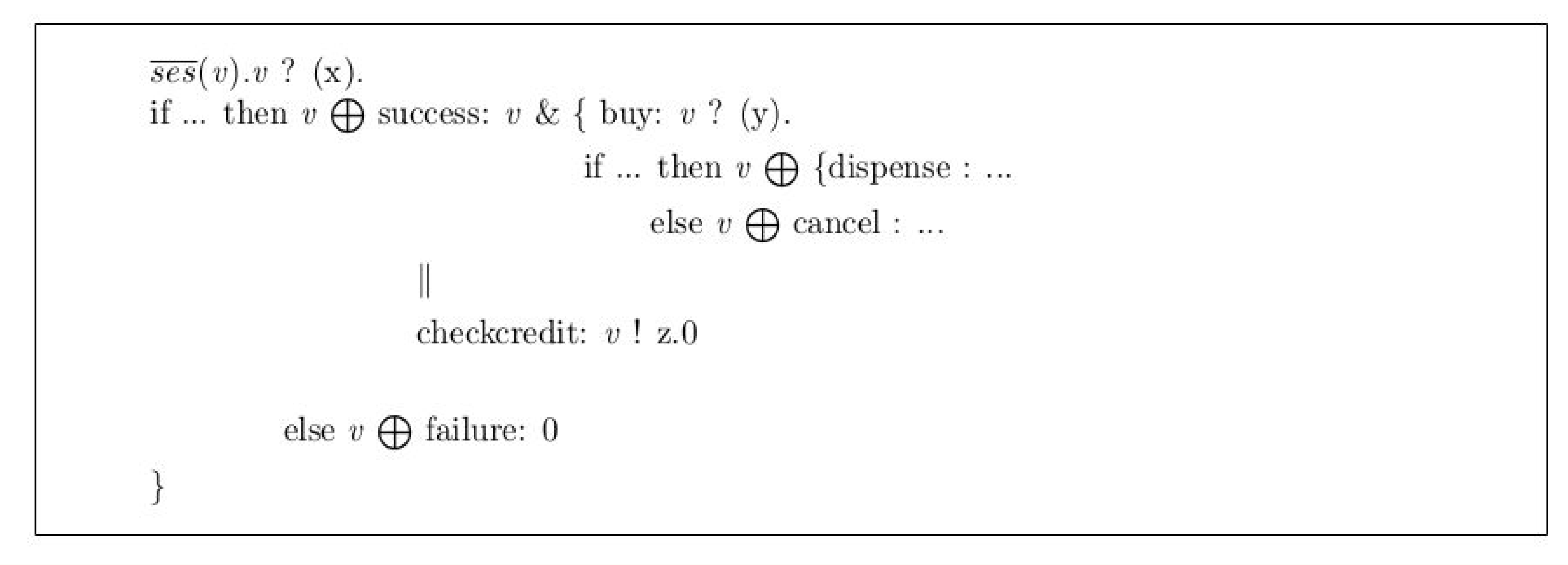
User Agent



```
ses(u).u ! cardIdentifier.
u & { success: if ... then u ⊕ buy: u ! item.
                                     u & {dispense : ...
                                          || cancel : ...
                                          }
                     else u ⊕ checkcredit: u ? (y).0

          || failure: 0
}
```

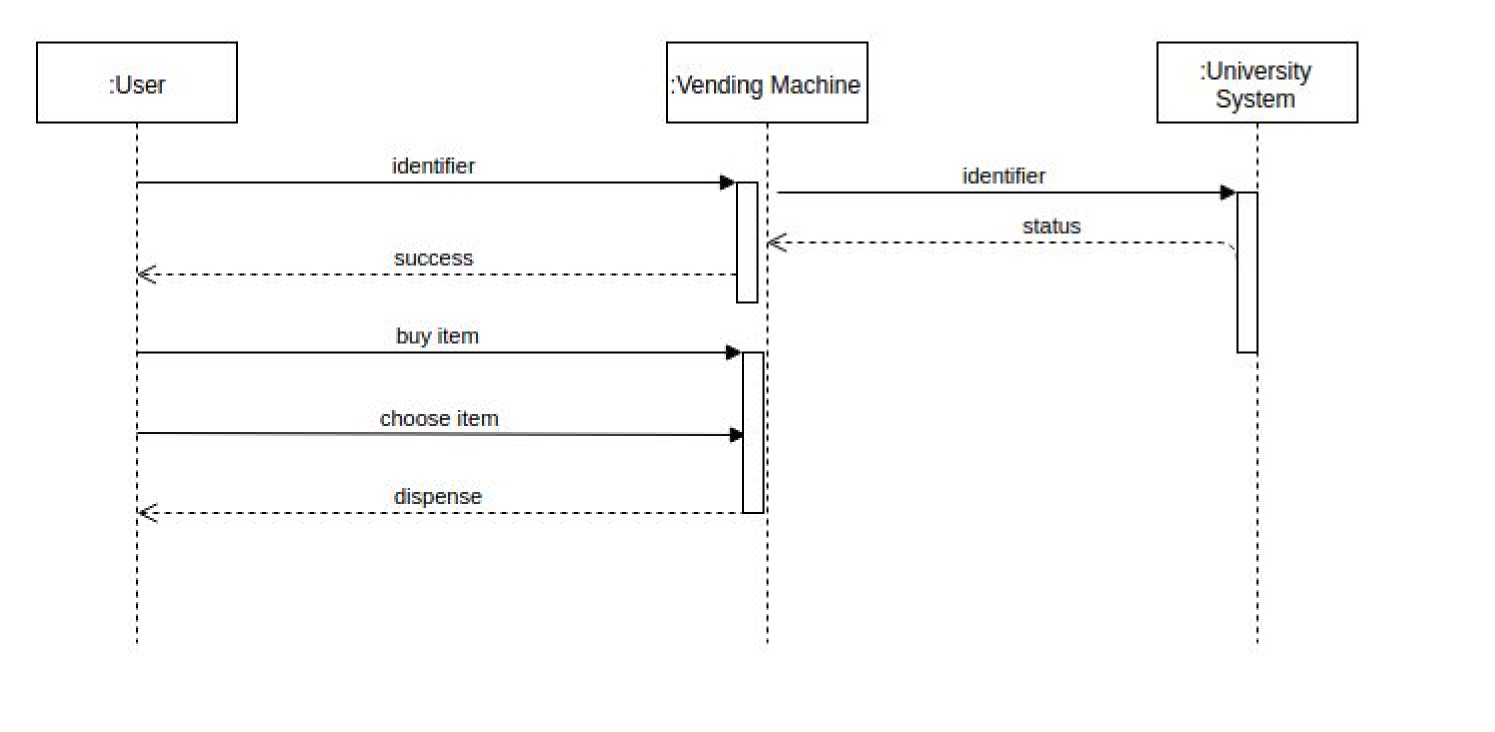
Vending Machine

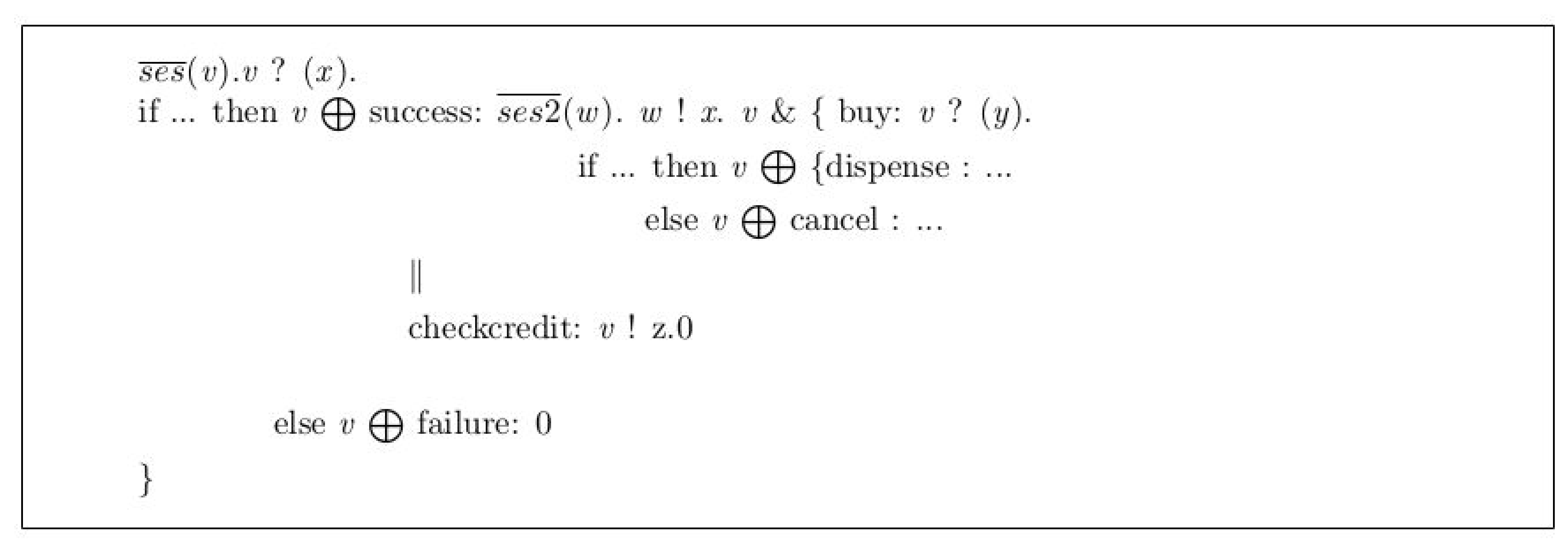


```
$\overline{ses}(v).v$ ? (x).
if ... then $v \oplus$ success: $v$ & { buy: $v$ ? (y).
                                  if ... then $v \oplus$ {dispense : ...
                                       else $v \oplus$ cancel : ...
                    ||
                    checkcredit: $v$ ! z.0

          else $v \oplus$ failure: 0
}
```

Delegation

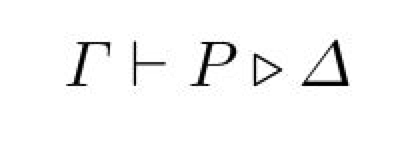

:User
:Vending Machine
:University System
identifier
identifier
status
success
buy item
choose item
dispense



```
$\overline{ses}(v).v\ ?\ (x).$
if ... then $v \bigoplus$ success: $\overline{ses2}(w).\ w\ !\ x.\ v\ \&\ \{$ buy: $v\ ?\ (y).$
                                    if ... then $v \bigoplus$ {dispense : ...
                                         else $v \bigoplus$ cancel : ...
                      ||
                      checkcredit: $v$ ! z.0

         else $v \bigoplus$ failure: 0
}
```

Typing System

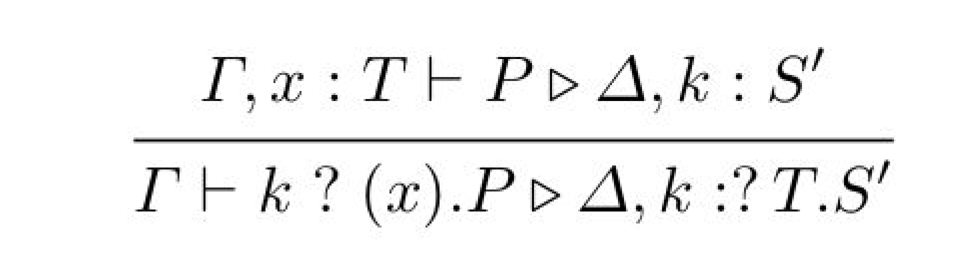


$$\Gamma \vdash P \triangleright \Delta$$

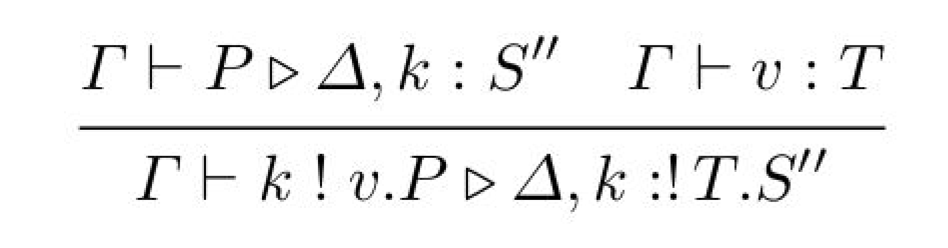
Rules - Session Initiation

$$\frac{\mathit{\Gamma}, a : [S] \vdash P \triangleright \mathit{\Delta}, k : S}{\mathit{\Gamma}, a : [S] \vdash a(k).P \triangleright \mathit{\Delta}} \qquad \frac{\mathit{\Gamma}, a : [S] \vdash P \triangleright \mathit{\Delta}, k : \overline{S}}{\mathit{\Gamma}, a : [S] \vdash \bar{a}(k).P \triangleright \mathit{\Delta}}$$

Receive and Send Value

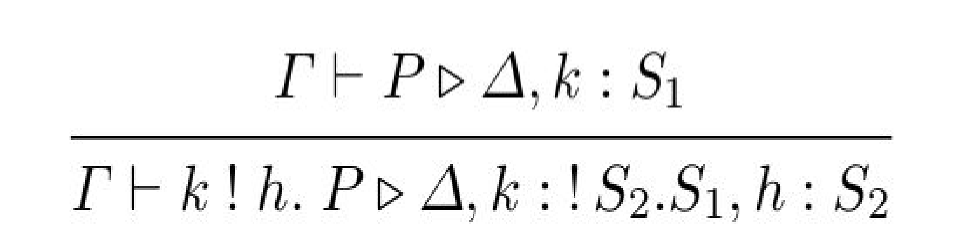


$$\frac{\Gamma, x : T \vdash P \triangleright \Delta, k : S'}{\Gamma \vdash k\ ?\ (x).P \triangleright \Delta, k :?\, T.S'}$$

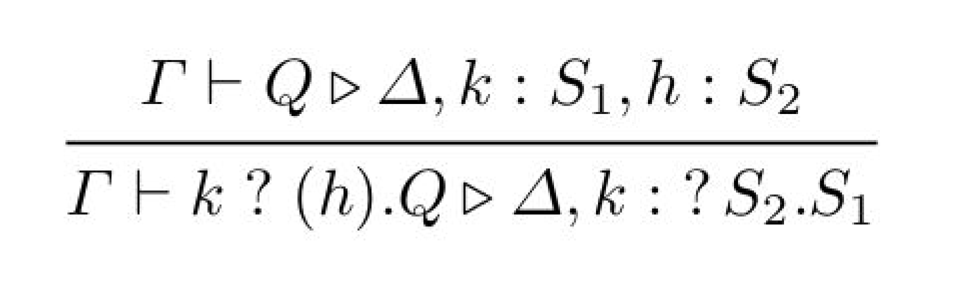


$$\frac{\Gamma \vdash P \triangleright \Delta, k : S'' \qquad \Gamma \vdash v : T}{\Gamma \vdash k\ !\ v.P \triangleright \Delta, k :!\, T.S''}$$

Delegation



$$\frac{\Gamma \vdash P \triangleright \Delta, k : S_1}{\Gamma \vdash k\,!\,h.\,P \triangleright \Delta, k : !\,S_2.S_1, h : S_2}$$



$$\frac{\Gamma \vdash Q \triangleright \Delta, k : S_1, h : S_2}{\Gamma \vdash k\,?\,(h).Q \triangleright \Delta, k : ?\,S_2.S_1}$$

Extension

- Extension of the Calculus
  - Correspondence Assertion
  - Multiparty Sessions
  - Concurrent Constraint
  - Code Mobility
  - Exception
- Extension of Typing
  - Subtyping
  - Bounded Polymorphism
  - Progress
  - Action Permutation

Implementation

- Functional Paradigm
  - Haskell
- Object Oriented Programming
  - Sing#
  - SJ
  - Scribbe
  - Bica

Questions?