

Code

Logic

Lemmas

Rules

Heuristics

```
let up_hdr ev abv hdr =
| ESend, NoHdr ->
    let origin = getPeer
    if s.recv_credit.(o
up ev abv
| _ -> failwith

∀m:Msg. P(m)⇒∃t:T. Q(mt)
| EFail ->
    for rank = 0 to pred
    if Arrayf.get failed r
| EDump -> ( dump (ls,vs)
| _ -> upnm ev

∃p:Path. ∀t:Time. R(p,t)
| ESend ->
    let dest = getPeer ev
∀p:Path. ∀m:Msg. T(p,m)
    else Queue.add
```

Correct!

Code

Logic

Lemmas

Rules

Heuristics

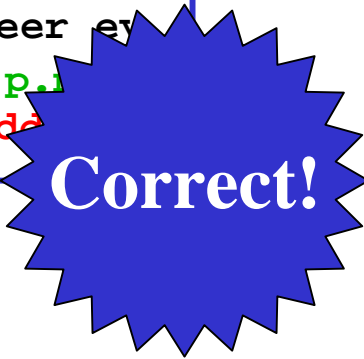
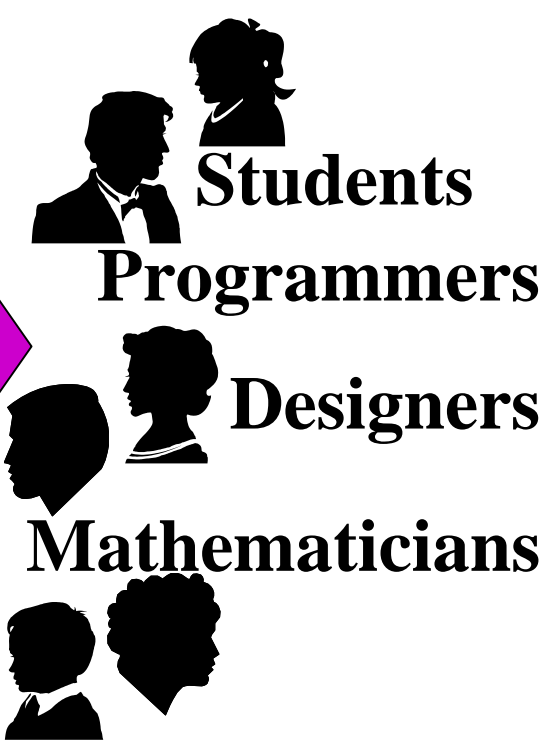
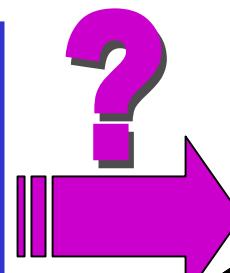
```

let up_hdr ev abv hdr =
| ESend, NoHdr ->
  let origin = getPeer
  if s.recv_credit.(o
up ev abv
| _ -> failwith

∀m:Msg. P(m)⇒∃t:T. Q(mt)
| EFail ->
  for rank = 0 to pred
  if Arrayf.get failed r
| EDump -> ( dump (ls,vs)
| _ -> upnm ev

∃p:Path. ∀t:Time. R(p,t)
| ESend ->
  let dest = getPeer ev
∀p:Path. ∀m:Msg. T(p,m)
  else Queue.add

```



Code

Logic

Lemmas

Rules

Heuristics

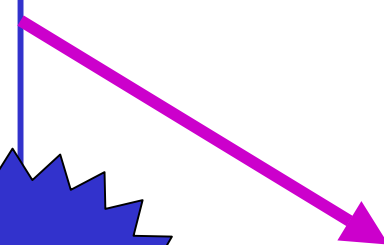
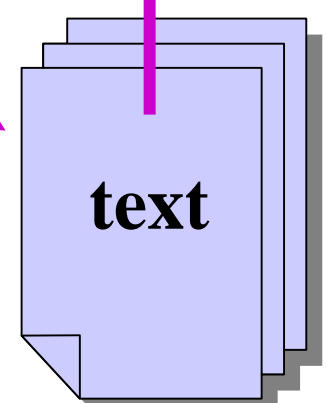
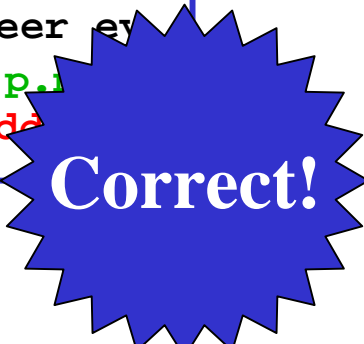
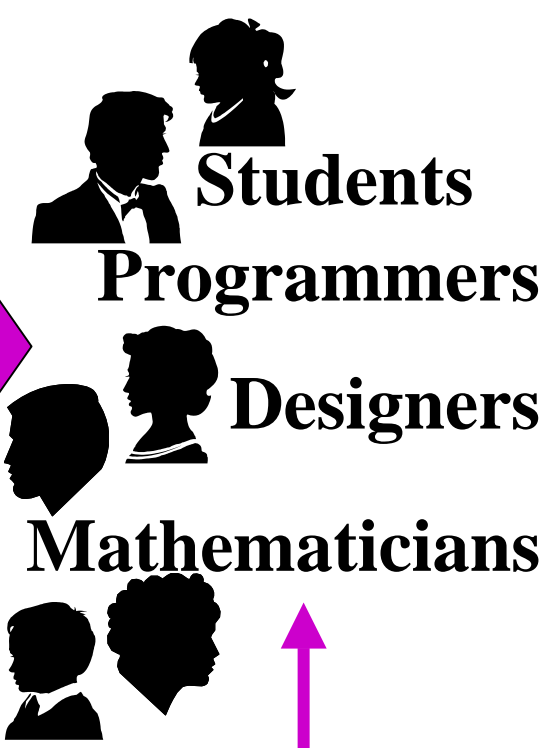
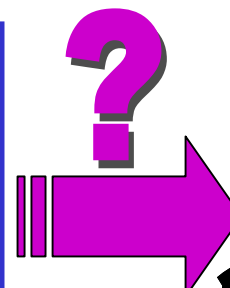
```

let up_hdr ev abv hdr =
| ESend, NoHdr ->
  let origin = getPeer
  if s.recv_credit.(o
up ev abv
| _ -> failwith

∀m:Msg. P(m)⇒∃t:T. Q(mt)
| EFail ->
  for rank = 0 to pred
  if Arrayf.get failed r
| EDump -> ( dump (ls,vs)
| _ -> upnm ev

∃p:Path. ∀t:Time. R(p,t)
| ESend ->
  let dest = getPeer ev
∀p:Path. ∀m:Msg. T(p,
  else Queue.add

```

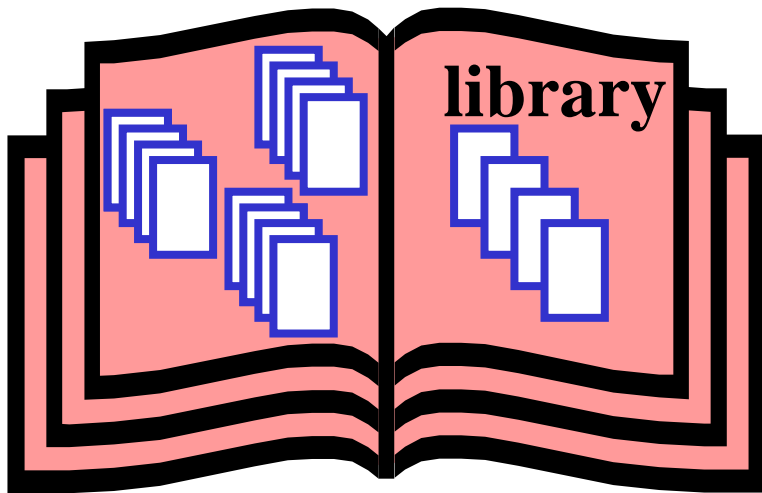


```
let up_hdlr ev abv hdr =
| ESend, NoHdr ->
    let origin = getPeer
    if s.recv_credit.(o
up ev abv
| _ -> failwith

∀m:Msg. P(m)⇒∃t:T. Q(mt)
| EFail ->
    for rank = 0 to pred
    if Arrayf.get failed r
| EDump -> ( dump (ls,vs)
| _ -> upnm ev

∃p:Path. ∀t:Time. R(p,t)
| ESend ->
    let dest = getPeer ev
∀p:Path. ∀m:Msg. T(p,m)
    else Queue.add (ev,
```

Library Collection

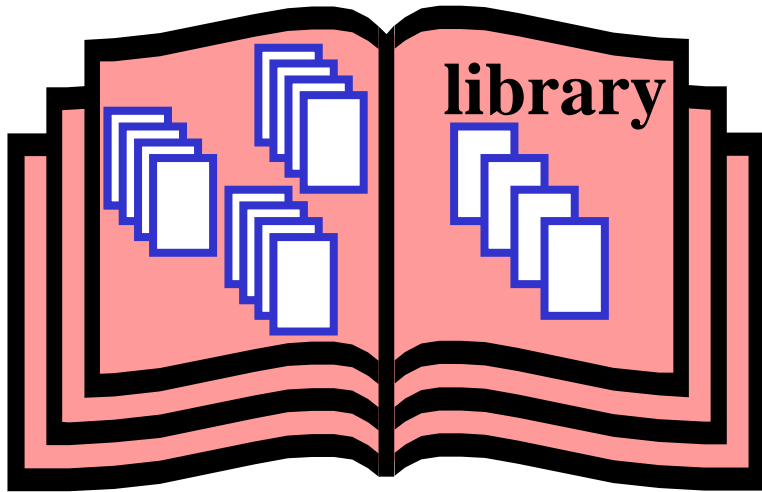


```
let up_hdlr ev abv hdr =
| ESend, NoHdr ->
    let origin = getPeer
    if s.recv_credit.(o
up ev abv
| _ -> failwith

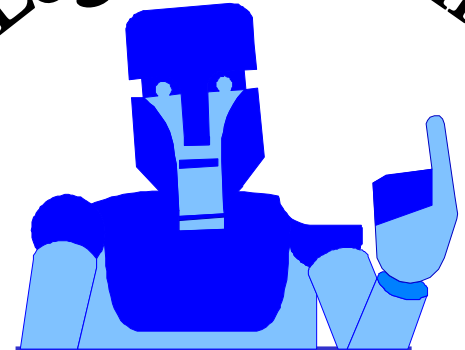
Vm:Msg. P(m)⇒∃t:T. Q(mt)
| EFail ->
    for rank = 0 to pred
    if Arrayf.get failed r
| EDump -> ( dump (ls,vs)
| _ -> upnm ev

Ep:Path. Vt:Time. R(p,t)
| ESend ->
    let dest = getPeer ev
Vp:Path. Vm:Msg. T(p,m)
    else Queue.add (ev,
```

Library Collection

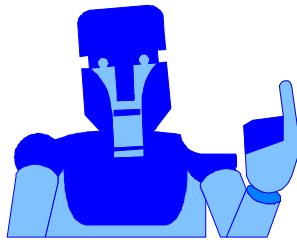
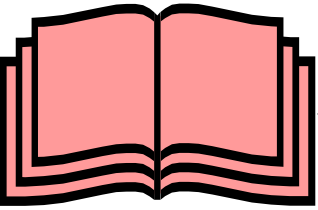


Logical System



heuristics
logic rules
typing

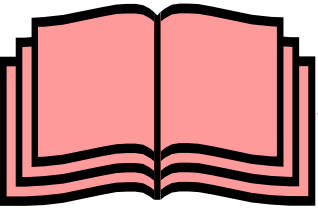
```
let up_hdr ev abv hdr =  
| ESend, NoHdr ->  
  let origin = getPeer  
  if s.recv_credit.(o  
  up ev abv  
| _ -> failwith  
  
Vm:Msg. P(m)⇒∃t:T. Q(mt)  
| EFail ->  
  for rank = 0 to pred  
  if Arrayf.get failed r  
| EDump -> ( dump (ls,vs)  
| _ -> upnm ev  
  
Ep:Path. Vt:Time. R(p,t)  
| ESend ->  
  let dest = getPeer ev  
Vp:Path. Vm:Msg. T(p,m)  
  else Queue.add (ev,
```



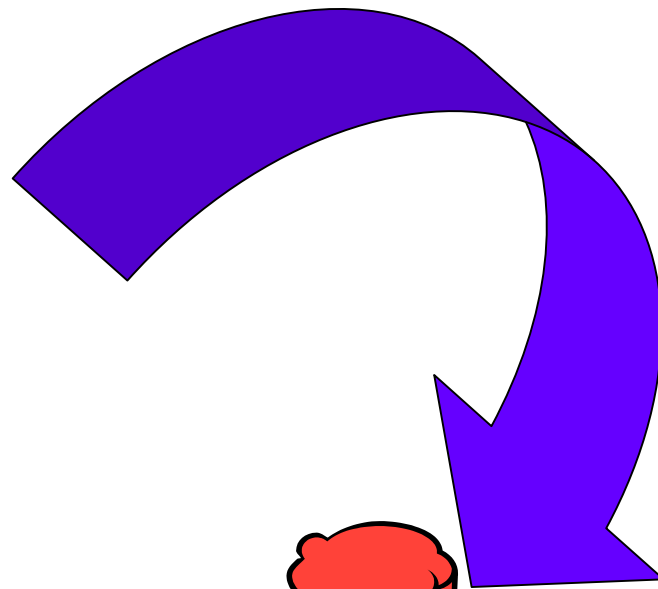
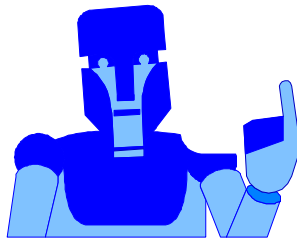
```
let up_hdr ev abv_hdr =
| ESend, NoHdr ->
  let origin = getPeer
  if s.recv_credit.(o
  up ev abv
| _ -> failwith

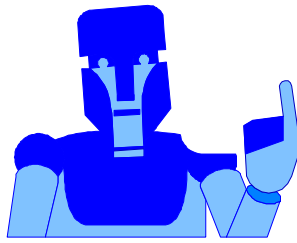
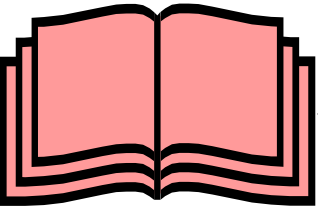
∀m:Msg. P(m)⇒∃t:T. Q(mt)
| EFail ->
  for rank = 0 to pred
  if Arrayf.get failed r
| EDump -> ( dump (ls,vs)
| _ -> upnm ev

∃p:Path. ∀t:Time. R(p,t)
| ESend ->
  let dest = getPeer ev
∀p:Path. ∀m:Msg. T(p,m)
  else Queue.add (ev,
```

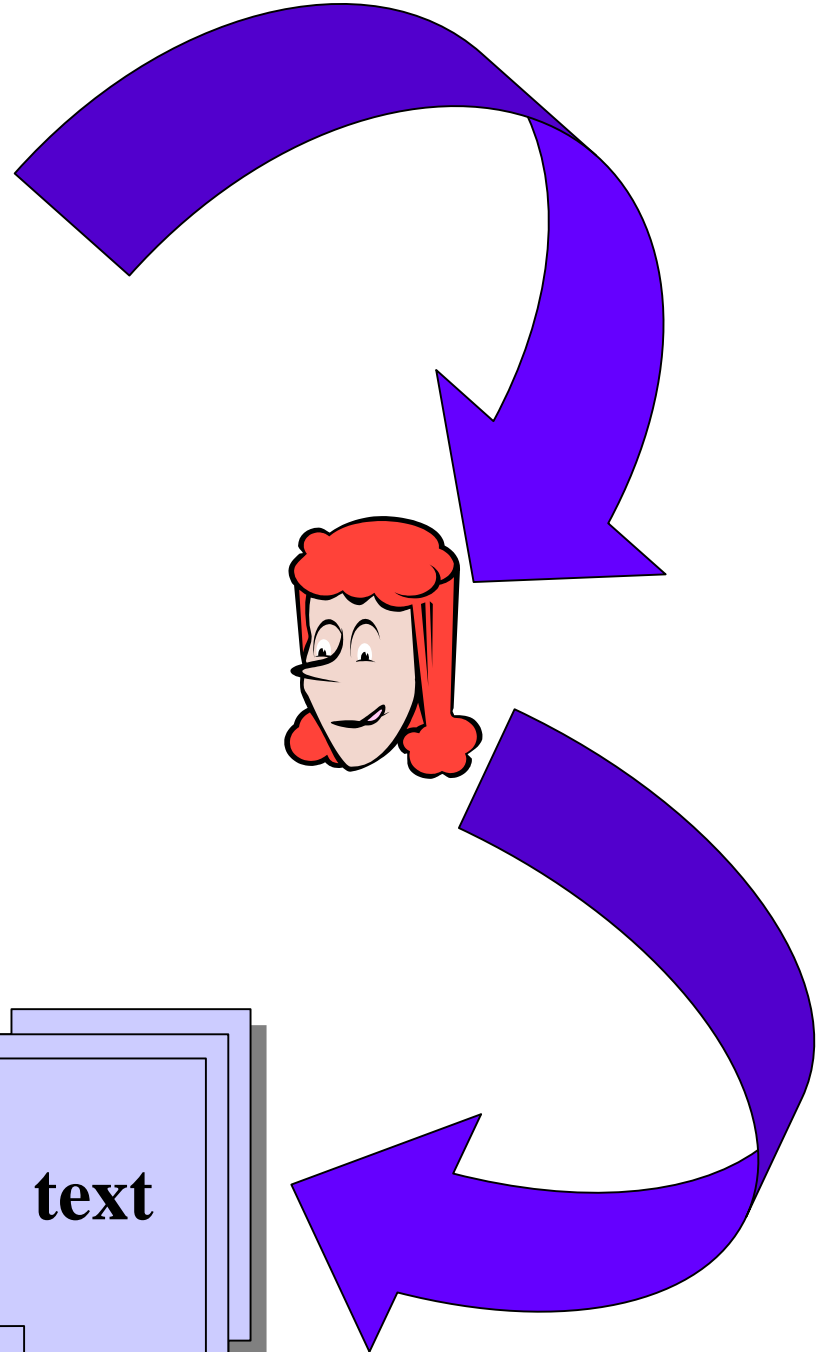
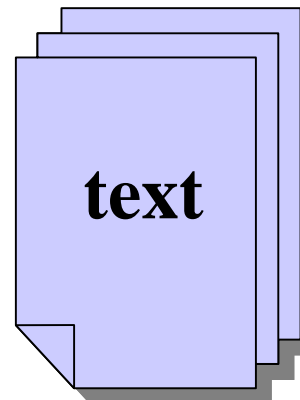


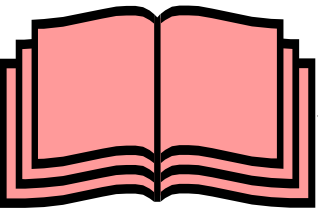
```
let up_hdr ev abv_hdr =  
| ESend, NoHdr ->  
  let origin = getPeer  
  if s.recv_credit.(o  
  up ev abv  
| _ -> failwith  
  
∀m:Msg. P(m)⇒∃t:T. Q(mt)  
| EFail ->  
  for rank = 0 to pred  
  if Arrayf.get failed r  
| EDump -> ( dump (ls,vs)  
| _ -> upnm ev  
  
∃p:Path. ∀t:Time. R(p,t)  
| ESend ->  
  let dest = getPeer ev  
∀p:Path. ∀m:Msg. T(p,m)  
  else Queue.add (ev,
```





```
let up_hdr ev abv_hdr =  
| ESend, NoHdr ->  
  let origin = getPeer  
  if s.recv_credit.(o  
  up ev abv  
| _ -> failwith  
  
∀m:Msg. P(m)⇒∃t:T. Q(mt)  
| EFail ->  
  for rank = 0 to pred  
  if Arrayf.get failed r  
| EDump -> ( dump (ls,vs)  
| _ -> upnm ev  
  
∃p:Path. ∀t:Time. R(p,t)  
| ESend ->  
  let dest = getPeer ev  
∀p:Path. ∀m:Msg. T(p,m)  
  else Queue.add (ev,
```





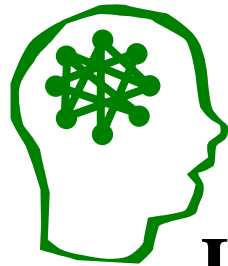
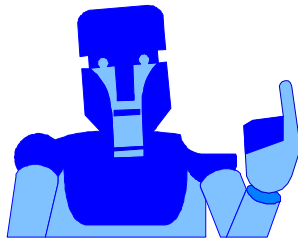
```

let up_hdr ev abv hdr =
| ESend, NoHdr ->
  let origin = getPeer
  if s.recv_credit.(o
  up ev abv
| _ -> failwith

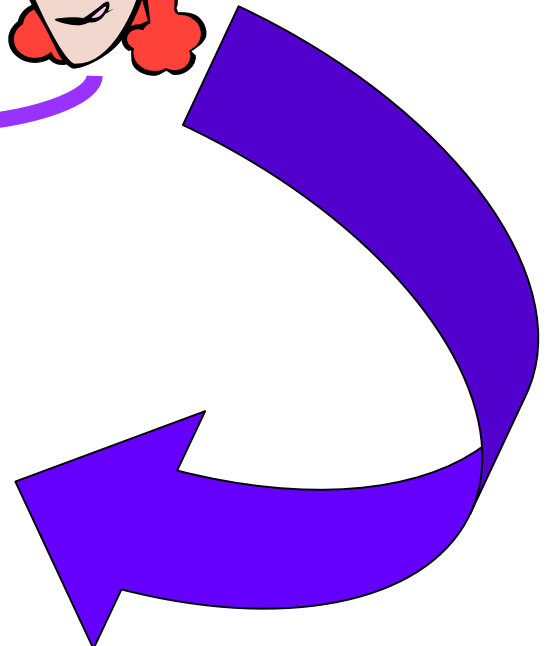
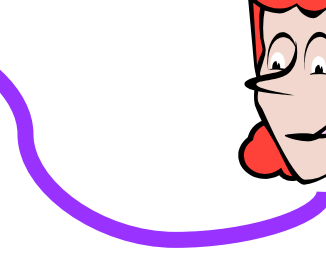
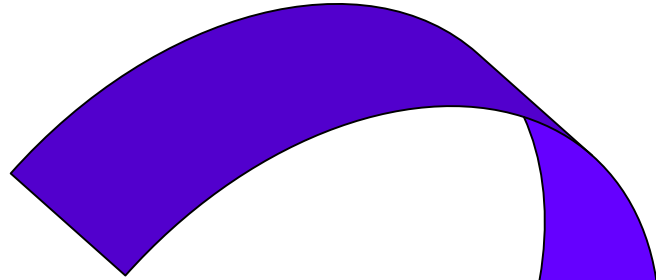
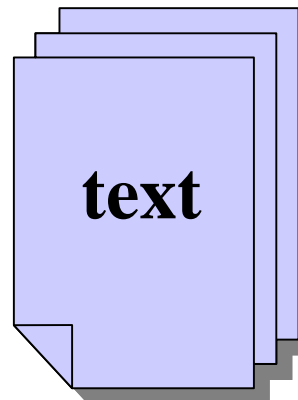
Vm:Msg. P(m)⇒T:Q(mt)
| EFail ->
  for rank = 0 to pred
  if Arrayf.get failed r
| EDump -> ( dump (ls,vs)
| _ -> upnm ev

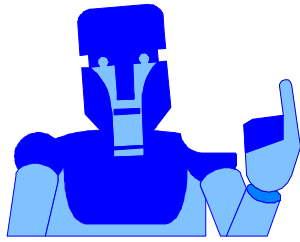
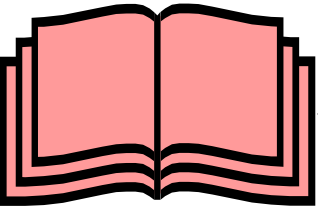
Ep:Path. Vt:Time. R(p,t)
| ESend ->
  let dest = getPeer ev
Vp:Path. Vm:Msg. T(p,m)
  else Queue.add (ev,

```



**User
Model**





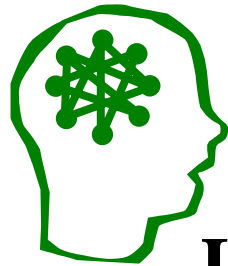
```

let up_hdr ev abv_hdr =
| ESend, NoHdr ->
  let origin = getPeer
  if s.recv_credit.(o
  up ev abv
| _ -> failwith

Vm:Msg. P(m) => T:Time. Q(mt)
| EFail ->
  for rank = 0 to pred
  if Arrayf.get failed r
| EDump -> ( dump (ls,vs)
| _ -> upnm ev

Ep:Path. Vt:Time. R(p,t)
| ESend ->
  let dest = getPeer ev
Vp:Path. Vm:Msg. T(p,m)
  else Queue.add (ev,

```



**User
Model**

