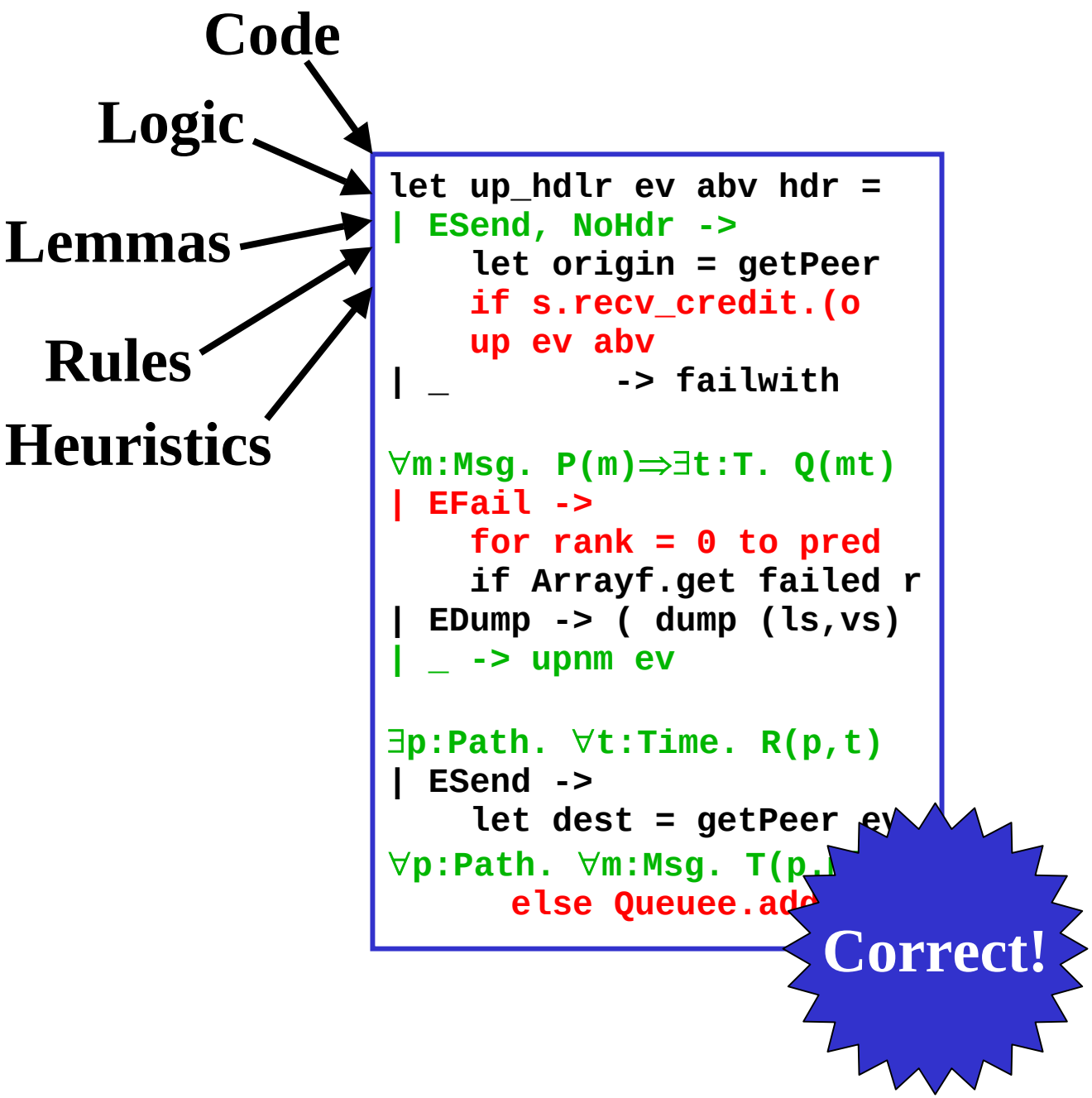
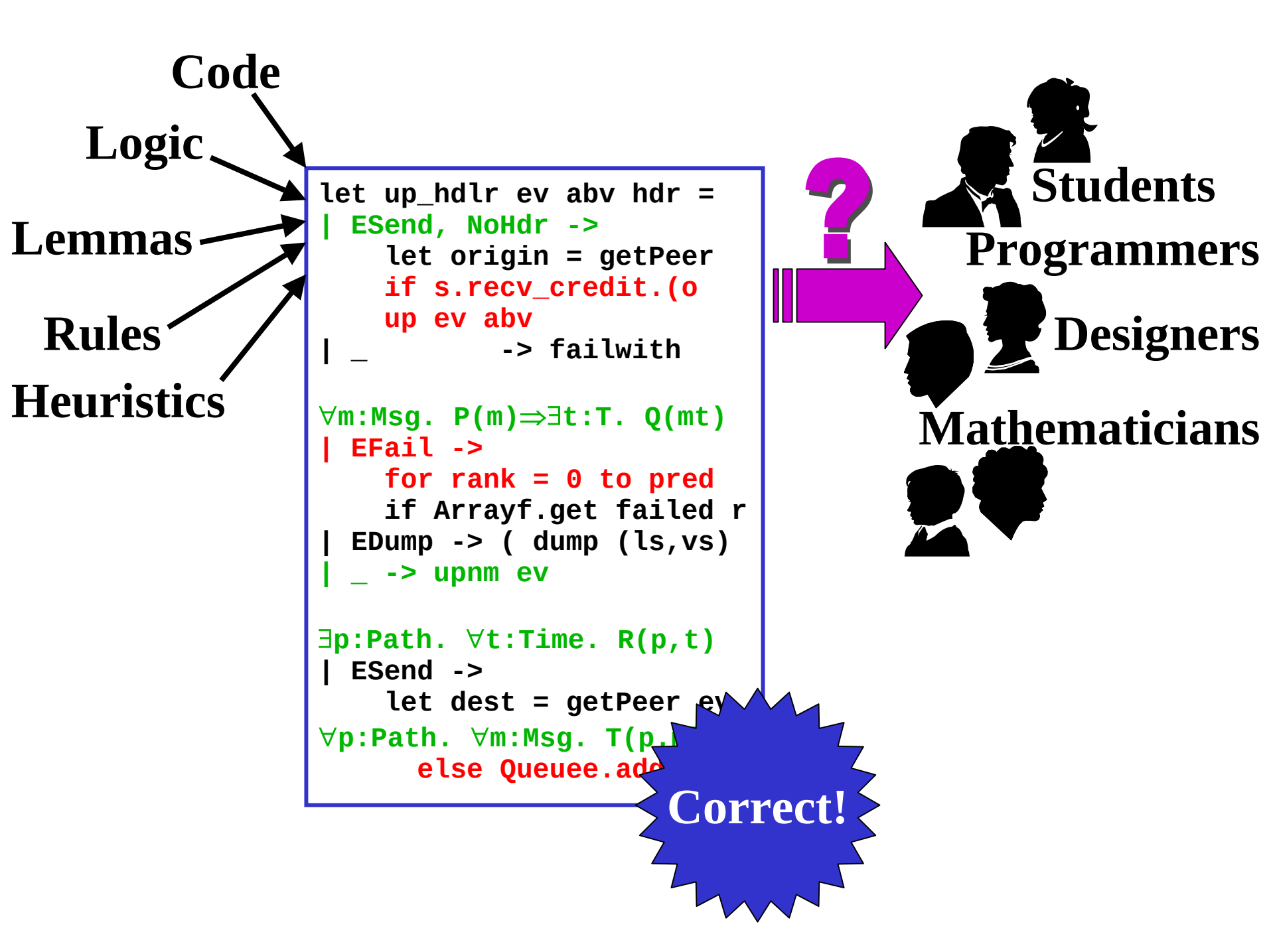
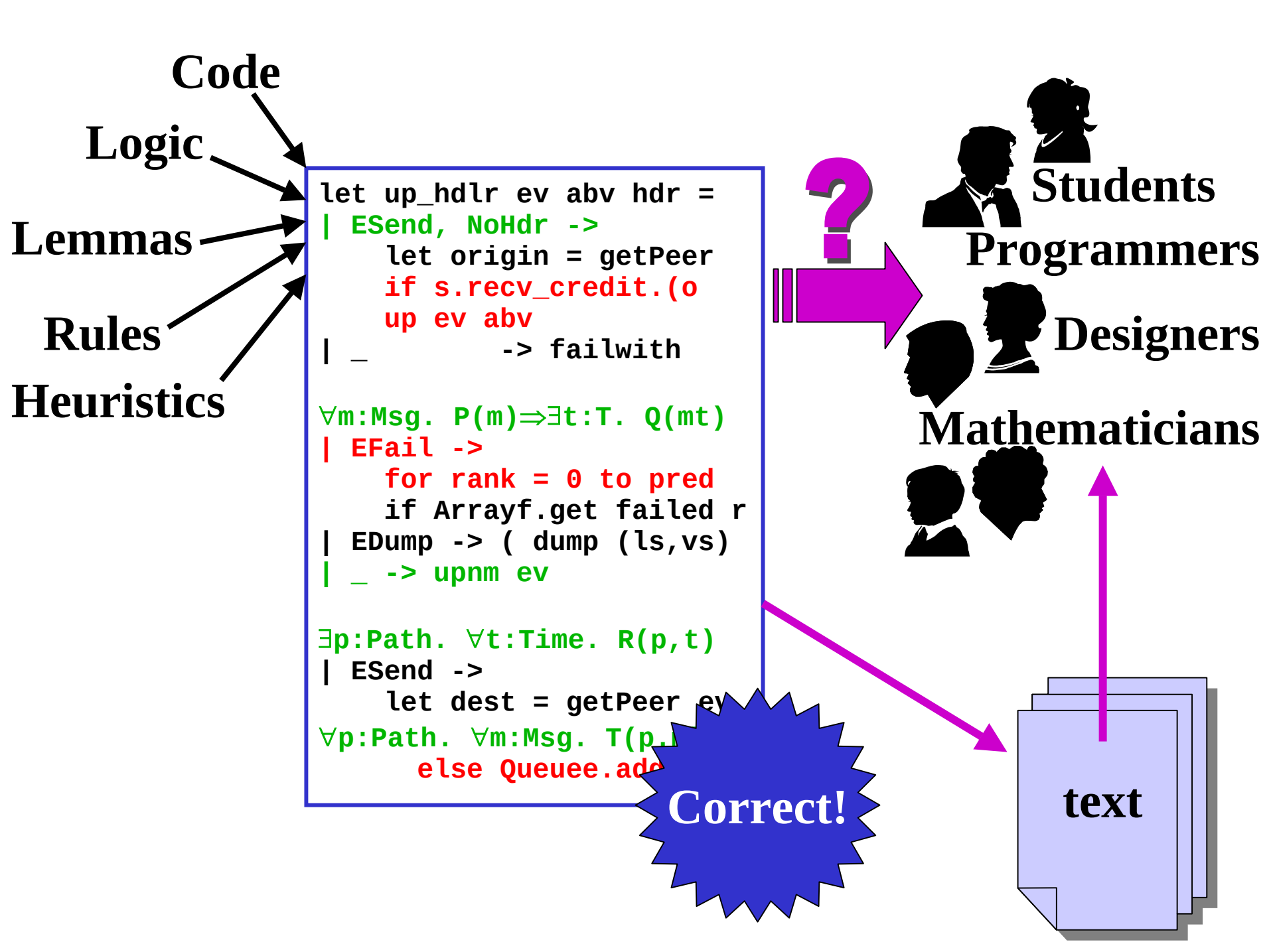
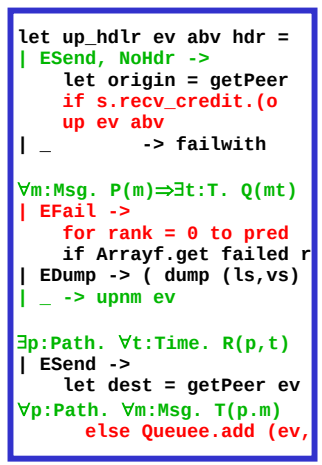
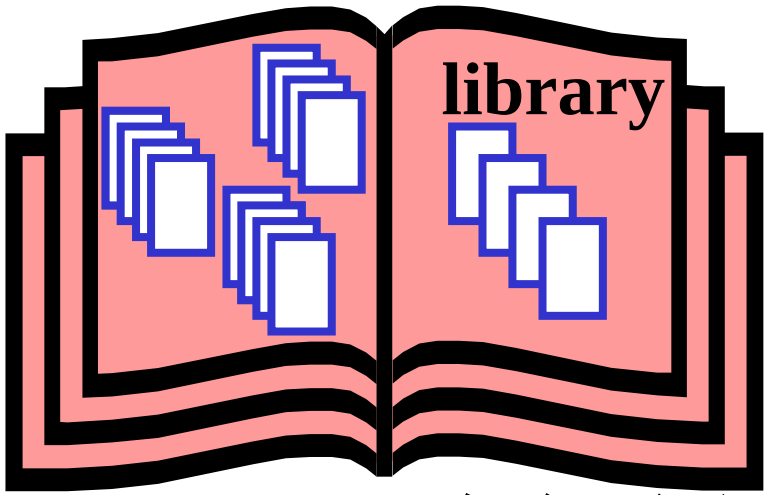

Code
Logic
Lemmas
Rules
Heuristics
let up_hdlr ev abv hdr =
| ESend, NoHdr ->
let origin = getPeer
if s.recv_credit.(o
up ev abv
| _ -> failwith
∀m:Msg. P(m)⇒∃t:T. Q(mt)
| EFail ->
for rank = 0 to pred
if Arrayf.get failed r
| EDump -> (dump (ls,vs)
| _ -> upnm ev
∃p:Path. ∀t:Time. R(p,t)
| ESend ->
let dest = getPeer ev
∀p:Path. ∀m:Msg. T(p,
else Queuee.ad
Correct!


Code
Logic
Lemmas
Rules
Heuristics
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,
else Queuee.ad
?
Students
Programmers
Designers
Mathematicians
Correct!


Code
Logic
Lemmas
Rules
Heuristics
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,
else Queuee.ad
?
Students
Programmers
Designers
Mathematicians
Correct!
text

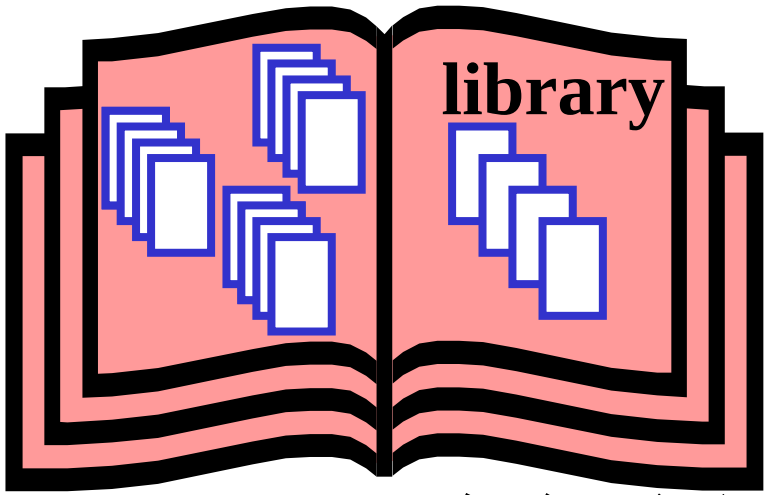

let up_hdlr ev abv hdr =
| ESend, NoHdr ->
let origin = getPeer
if s.recv_credit.(o
up ev abv
| _ -> failwith
∀m:Msg. P(m)⇒∃t:T. Q(mt)
| EFail ->
for rank = 0 to pred
if Arrayf.get failed r
| EDump -> (dump (ls,vs)
| _ -> upnm ev
∃p:Path. ∀t:Time. R(p,t)
| ESend ->
let dest = getPeer ev
∀p:Path. ∀m:Msg.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

Library Collection

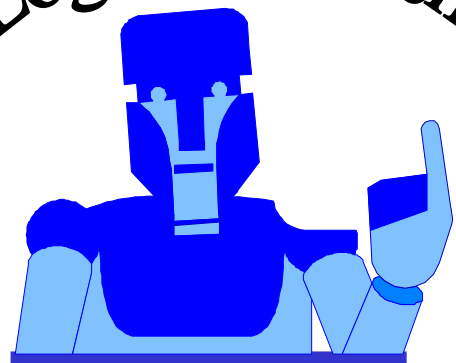



```
let up_hdlr ev abv hdr =
| ESend, NoHdr ->
    let origin = getPeer
    if s.recv_credit.(o
    up ev abv
| _         -> failwith

∀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 Queuee.add (ev,
```

Library Collection

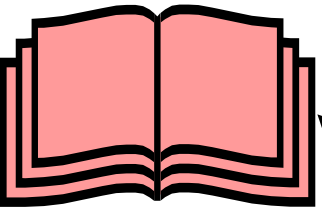


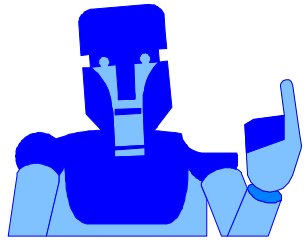
Logical System



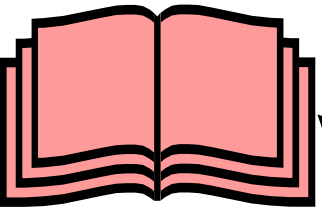
heuristics
logic rules
typing

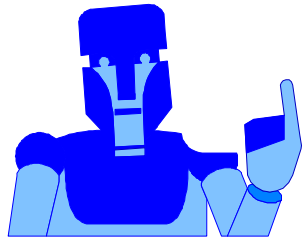
```
let up_hdlr ev abv hdr =
| ESend, NoHdr ->
    let origin = getPeer
    if s.recv_credit.(o
    up ev abv
| _          -> failwith

∀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 Queuee.add (ev,
```

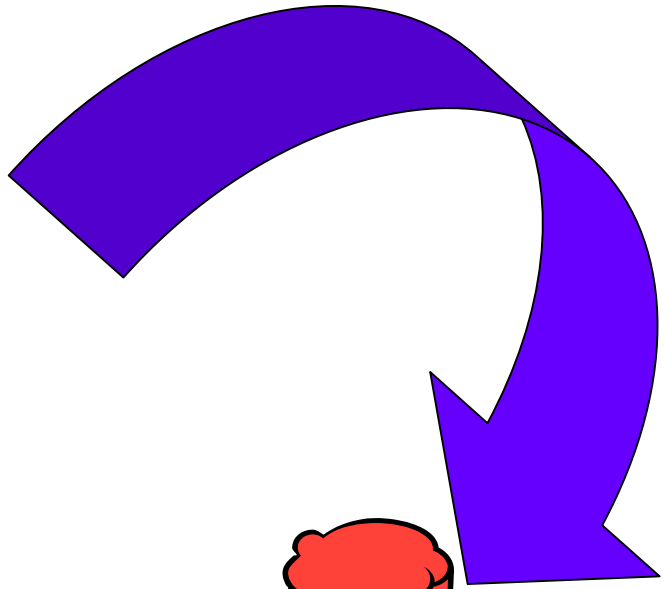


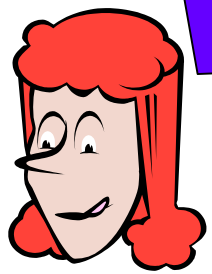


```
let up_hdlr ev abv hdr =
| ESend, NoHdr ->
    let origin = getPeer
    if s.recv_credit.(o
    up ev abv
| _        -> failwith

∀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 Queuee.add (ev,
```

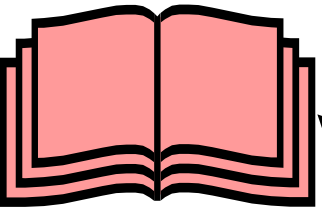


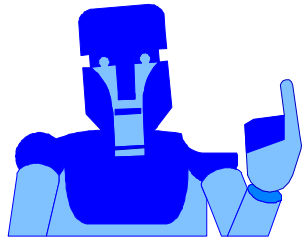




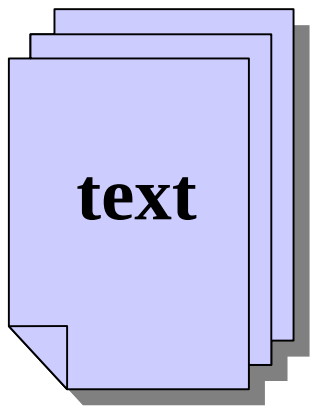


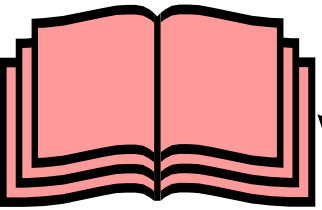
```
let up_hdlr ev abv hdr =
| ESend, NoHdr ->
    let origin = getPeer
    if s.recv_credit.(o
    up ev abv
| _         -> failwith

∀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 Queuee.add (ev,
```

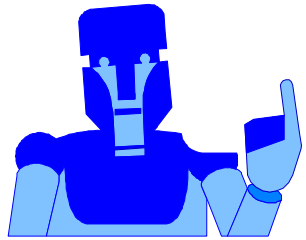




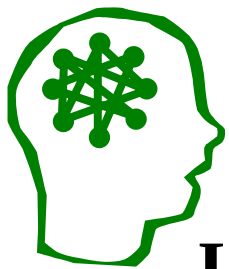
```
let up_hdlr ev abv hdr =
| ESend, NoHdr ->
    let origin = getPeer
    if s.recv_credit.(o
    up ev abv
| _         -> failwith

∀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 Queuee.add (ev,
```

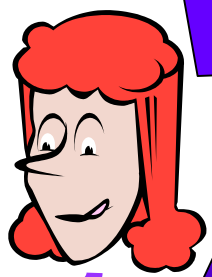




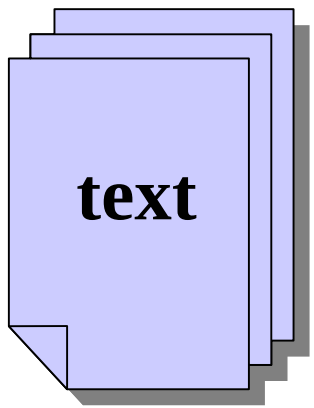


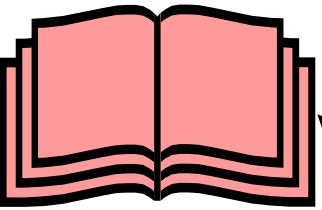
```
let up_hdlr ev abv hdr =
| ESend, NoHdr ->
    let origin = getPeer
    if s.recv_credit.(o
    up ev abv
| _         -> failwith

∀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 Queuee.add (ev,
```

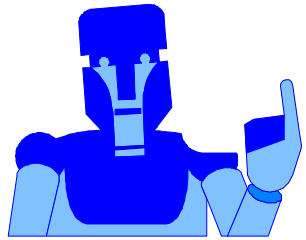




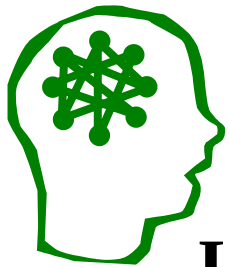
User Model

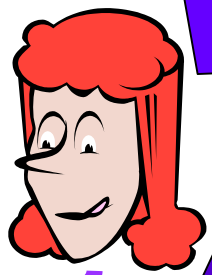






```
let up_hdlr ev abv hdr =
| ESend, NoHdr ->
    let origin = getPeer
    if s.recv_credit.(o
    up ev abv
| _         -> failwith

∀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 Queuee.add (ev,
```





User Model