SRML Examples
The orchestration of the trip broker
The Module
BookTrip

![Diagram of BookTrip module with connections between CR (Customer), CB (Registry), intBR (Broker), intPA (PayAgent), intFA (FlightAgent), intHA (HotelAgent), and BE (HotelAgent)].
The States of Broker

**BUSINESS ROLE** Broker is

**INTERACTIONS**

... 

**ORCHESTRATION**

local

s:\{INIT, QUERIED, FLIGHT_OK, HOTEL_OK, CONFIRMED, END_PAID, END_UNBOOKED, END_COMPENSATED\}

```
intBR & init
s=INIT

intBR & term
(s=END_UNBOOKED \lor 
(s=END_PAID \land today=bookTrip.out)) \lor 
END_COMPENSATED
```
One Sequence Diagram...

CR -> bookTrip

bookTrip -> CR

BR -> bookFlight

bookFlight -> FA

RE -> bookHotel

bookHotel -> HA

HA -> payment

payment -> PA

PA

CR

bookTrip

bookTrip

CR

BR

bookFlight

bookFlight

FA

RE

bookHotel

bookHotel

HA

PA

payment

payment

card(userdata)
**BUSINESS ROLE** Broker is

**INTERACTIONS**

```plaintext
r&s bookTrip
   from,to:airport
   out,in:date
   usr:userdata
  fconf:fcode
  hconf:hcode
  amount:moneyData
s&r bookFlight
   from,to:airport
   out,in:date
  fconf:fcode
  amount:moneyData
  beneficiary:accountNo
  payService:serviceId
...```

**transition** Request

- **triggeredBy** bookTrip
- **guardedBy** s=INIT
- **effects**
  - bookTrip.out≤today ⊃ s'=END_UNBOOKED
  - bookTrip.out>today ⊃ s'=QUERIED
  - sends bookTrip.out≤today ⊃ bookTrip
    - bookTrip.Reply=false
    - bookTrip.out>today ⊃ bookFlight
      - bookFlight.from=bookTrip.from
      - bookFlight.to=bookTrip.to
      - bookFlight.out=bookTrip.out
      - bookFlight.in=bookTrip.in
...
**BUSINESS ROLE** Broker

**INTERACTIONS**

r&s *bookTrip*

- from, to: airport
- out, in: date
- usr: userdata
- fconf: fcode
- hconf: hcode
- amount: moneyData

s&r *bookFlight*

- from, to: airport
- out, in: date
- fconf: fcode
- amount: moneyData
- beneficiary: accountNo
- payService: serviceId

s&r *bookHotel*

- checkin, checkout: date
- traveller: usrdata
- amount: moneyData
- hconf: hcode

...
BUSINESS ROLE Broker is

INTERACTIONS

rs bookTrip

tdown from, to: airport
out, in: date
usr: userdata

fconf: fcode
hconf: hcode
amount: moneyData

s&r bookFlight

(down from, to: airport
out, in: date

fconf: fcode
amount: moneyData
beneficiary: accountNo
payService: serviceId

s&r bookHotel

checkin, checkout: date
traveller: userdata

amount: moneyData
hconf: hcode

...
Find the difference ... (there are 5!)

---

**transition** HotelAnswer1

**triggeredBy** bookHotel

**guardedBy** s=FLIGHT_OK

**effects**
- bookHotel.Reply $\Rightarrow$ s'=HOTEL_OK
- $\neg$bookHotel.Reply $\Rightarrow$ s'=END_UNBOOKED

**sends**
- bookHotel.Reply $\Rightarrow$ bookTrip
  - $\land$ bookTrip.Reply=true
  - $\land$ bookTrip.fconf=bookFlight.fconf
  - $\land$ bookTrip.amount=bookFlight.amount+bookHotel.amount
  - $\land$ bookTrip.hconf=bookHotel.hconf
- $\neg$bookHotel.Reply $\Rightarrow$ bookFlight
  - $\land$ bookTrip.Reply=false

---

**transition** HotelAnswer2

**triggeredBy** bookHotel

**guardedBy** s=FLIGHT_OK

**effects**
- bookHotel.Reply=true $\Rightarrow$ s'=HOTEL_OK
- bookHotel.Reply=false $\Rightarrow$ s'=END_UNBOOKED

**sends**
- s'=HOTEL_OK $\Rightarrow$ bookTrip
  - $\land$ bookTrip.Reply=true
  - $\land$ bookTrip.fconf=bookFlight.fconf
  - $\land$ bookTrip.amount=bookFlight.amount+bookHotel.amount
  - $\land$ bookTrip.hconf=bookHotel.hconf
- s'=END_UNBOOKED $\Rightarrow$ bookFlight
  - $\land$ bookTrip.Reply=false

---

**Is the transition on the right correct?**  YES!

**Are the two transitions equivalent?**  YES!
... transition HotelAnswer1
  triggeredBy bookHotel
  guardedBy s=FLIGHT_OK
  effects bookHotel.Reply ⊃ s'=HOTEL_OK
                   ∧ ¬bookHotel.Reply ⊃ s'=END UNBOOKED
  sends bookHotel.Reply ⊃ bookTrip
     ∧ bookTrip.Reply=true
     ∧ bookTrip.fconf=bookFlight.fconf
     ∧ bookTrip.hconf=bookHotel.hconf
     ∧ ¬bookHotel.Reply ⊃ bookTrip
     ∧ bookTrip.Reply=false
...
Find the 5 differences ... (last game!)

... transition HotelAnswer1
  triggeredBy bookHotel
  guardedBy s=FLIGHT_OK
  effects bookHotel.Reply ⊃ s'=HOTEL_OK
      ^ bookHotel.Reply ⊃ s'=END_UNBOOKED
  sends bookHotel.Reply ⊃ bookTrip
      ^ bookTrip.Reply=true
      ^ bookTrip.fconf=bookFlight.fconf
      ^ bookTrip.hconf=bookHotel.hconf
      ^ ~bookHotel.Reply ⊃ bookFlight
      ^ bookTrip
      ^ bookTrip.Reply=false
...

... transition HotelAnswer1
  triggeredBy bookHotel
  guardedBy FLIGHT_OK
  effects bookHotel.Reply ⊃ HOTEL_OK
      ^ ~bookHotel.Reply ⊃ s'=END_UNBOOKED
  sends bookHotel.Reply ⊃ bookTrip
      ^ bookTrip.Reply
      ^ bookTrip.fconf=bookFlight.fconf
      ^ bookTrip.hconf=bookHotel.hconf
      ^ ~bookHotel.Reply ⊃ bookFlight
      ^ bookTrip
      ^ ~bookTrip.Reply'
...

Is the transitions on the right correct? NO!

Are the two transitions equivalent? NO!
BUSINESS ROLE  Broker is
INTERACTIONS

...  
\texttt{s\&r} bookFlight
\begin{itemize}
  \item from, to: airport
  \item out, in: date
  \item fconf: fcode
  \item amount: moneyData
  \item payService: serviceId
\end{itemize}
\texttt{s\&r} bookHotel
\begin{itemize}
  \item checkin, checkout: date
  \item traveller: usrdData
  \item amount: moneyData
  \item hconf: hcode
\end{itemize}
\texttt{s\&r} payment
\begin{itemize}
  \item amount: moneyValue
  \item originator: usrdData
  \item cardNo: payData
  \item proof: pcode
\end{itemize}
ask card(userdata): payData

Write a transition that...

- Has name Commitment
- It is triggered by the receive of the commit-event of bookTrip occurring in the state \( s=\text{HOTEL\_OK} \)
- Goes in the state CONFIRMED
- Triggers the initiation-event of payment (the amount is the sum of the flight and of the hotel price, the card number is obtained through the synchronous interaction card)
Commitment

transition Commitment

triggeredBy bookTrip

guardedBy s=HOTEL_OK

effects s’=CONFIRMED

sends payment

\^ payment.amount=bookFlight.amount+bookHotel.amount
\^ payment.originator=bookTrip.usr
\^ payAck.cardNo=card(bookTrip.usr)
BUSINESS ROLE Broker is

INTERACTIONS

r&s bookTrip

 from,to:airport
 out,in:date
 usr:userdata

 fconf:fcode
 hconf:hcode
 amount:moneyData

s&r bookFlight

 from,to:airport
 out,in:date

 fconf:fcode
 amount:moneyData
 payService:serviceId

...

Interactions

Write a transition that...

- Has name Timeout
- It is triggered by the event
  now>bookTrip.Useby occurring in the
  state s=HOTEL_OK
- Goes in the state END_UNBOOKED
Timeout

transition Timeout

triggeredBy now>bookTrip.UseBy

guardedBy s=HOTEL_OK

effects s'=END_UNBOOKED
BUSINESS ROLE Broker is INTERACTIONS

s&r payment
  ▶ amount:moneyvalue
  originator:usrdata
  cardNo:paydata
▷ proof:pcode
snd payAck
  ▶ proof:pcode
  status:boolean
s&r bookHotel
  ▶ checkin,checkout:date
  traveller:usrdata
▷ amount:moneyData
  hconf:hcode
ask card(userdata):paydata

Interactions

Write a transition that...

- Has name PaymentAnswer
- It is triggered by the receive-event payment occurring in the state S=CONFIRMED
  - if the parameter Reply is true than the transition goes in the state s=END_PAID
  - otherwise the transition goes in state S=END_UNBOOKED
- In both cases the transition triggers the initiation events for payAck setting the parameter status of payAck as the parameter Reply of payment and
  - if Reply of payment of payment is true then set pcode os payAck as the parameter pcode of payment
transition PaymentAnswer
  triggeredBy payment
  guardedBy s=CONFIRMED
  effects payment.Reply \(\supset\) s'\(\supset\)END_PAID
  \(\land\) \(\neg\)payment.Reply \(\supset\) s'\(\supset\)END_UNBOOKED
  sends payAck
  \(\land\) payAck.status=payment.Reply
  \(\land\) payment.Reply \(\supset\) payAck.proof=payment.proof
BUSINESS ROLE Broker is

INTERACTIONS

r&s bookTrip

根据不同参数
from,to:airport
out,in:date
usr:userdata
fconf:fcode
hconf:hcode
amount:moneyData

s&r bookFlight

根据不同参数
from,to:airport
out,in:date
fconf:fcode
amount:moneyData
beneficiary:accountNo
payService:serviceId

s&r bookHotel

根据不同参数
checkin,checkout:date
traveller:usrdatal
amount:moneyData
hconf:hcode
...

Write a transition that...

- Has name TripCompensate
- It is triggered by the revoke-event bookTrip occurring in the state END_PAID ONLY IF today is prior than the parameter out of bookTrip
- Goes in the state END_COMPENSATED
- Triggers the revoke (compensation) events for bookFlight and bookHotel
TripCompensate

TripCompensate

- **triggeredBy**: `bookTrip`†
- **guardedBy**: $s = \text{END\_PAID} \land \text{today}<\text{bookTrip.out}$
- **effects**: $s' = \text{END\_COMPENSATED}$
- **sends**: `bookFlight`† $\land$ `bookHotel`†