SRML and Policies

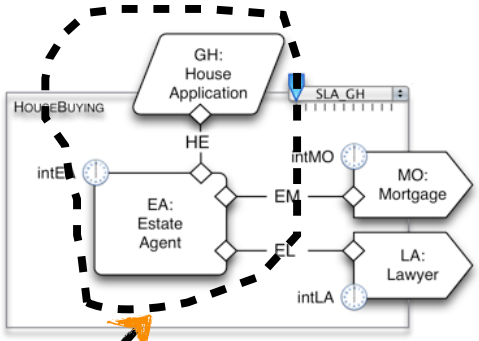
Laura Bocchi
bocchi@mcs.le.ac.uk

Agenda

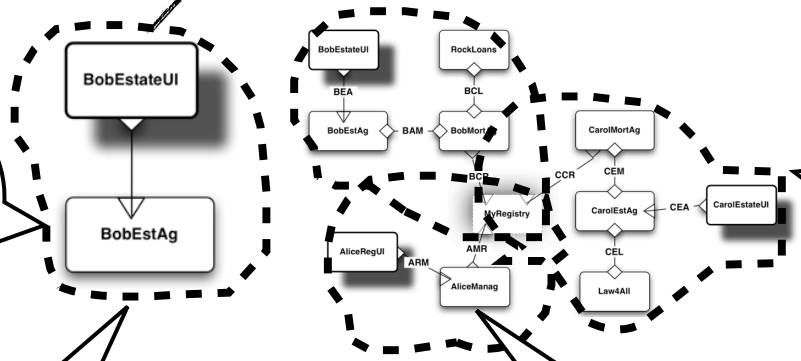
- Internal configuration policies
- External condiguration policies
  - Non-functional properties and SLA

Configuration Policies

- A **SRML module** describes one instance of the session of a **service** or an **activity**



- A configuration describes the active entities in the dimension overlaid by the service layer



The nodes of a configuration represent entities that execute some computation

The nodes of a configuration may derive form different initial moules

The nodes of a configuration correspond to the nodes of a SRML module

The computation of a node in a configuration is modelled by the specification of the node in the SRML module

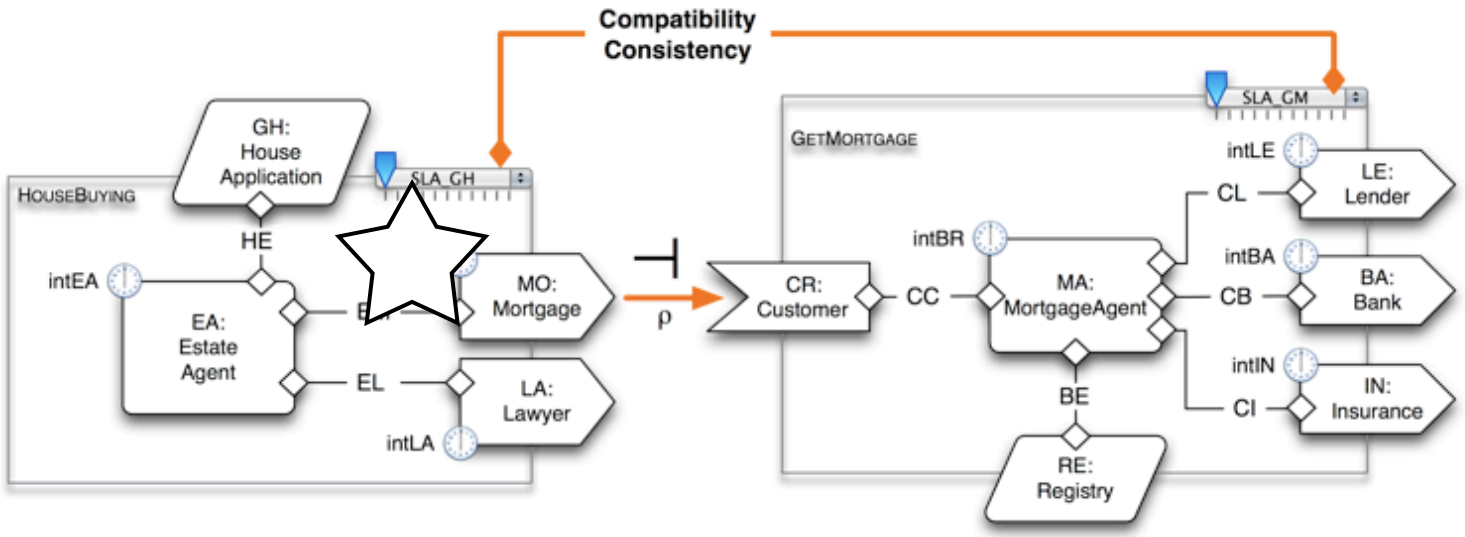
Configuration Policies

- During the execution of a module instance,
  - some user may launch a new activity from the top layer
  - some event happening in an existing node may trigger the a service discovery
- In these cases we have a **dynamic reconfiguration**
- A SRML module specifies configuration policies to model a number of aspects of the dynamic reconfiguration:

Internal configuration policies	when the reconfiguration should happen (for both Activities and Services)
	how a new instance should be initialized (for both Activities and Services)
Business protocols	which functional properties the Activity/Service requires to the discovered services
	which functional properties the Activity/Service requires to the discovered services
Internal configuration policies	which non-functional properties the Activity/Service requires to the discovered services
	which non-functional properties the Activity/Service requires to the discovered services

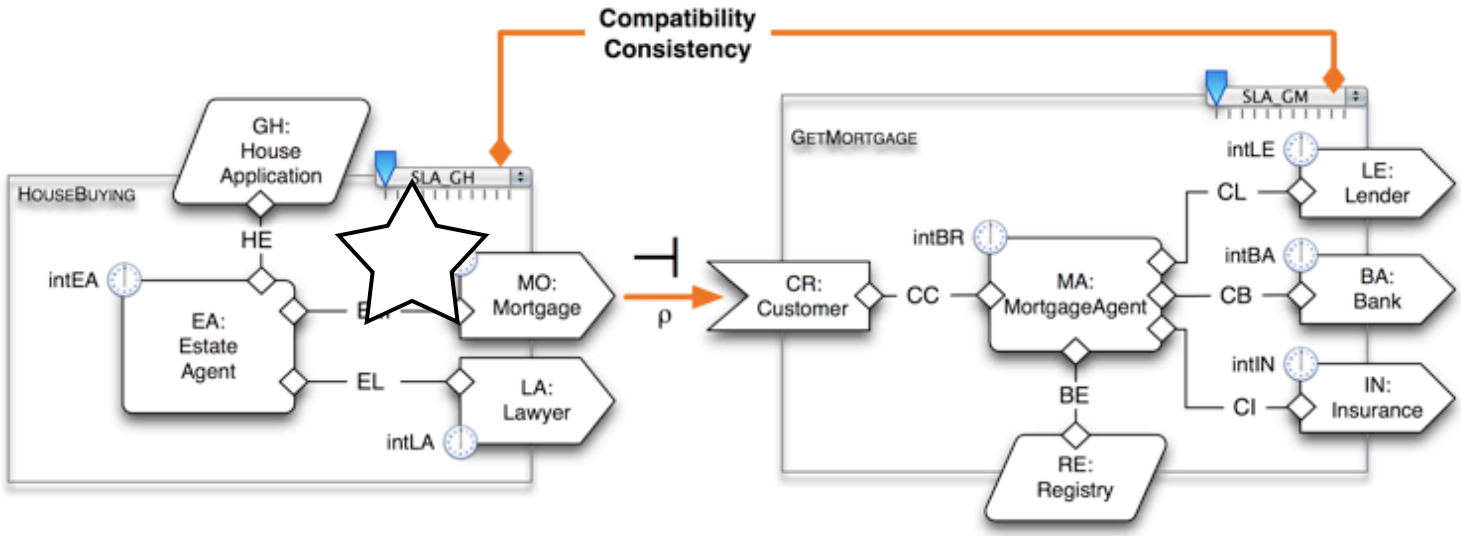
- SRML do not describe the discovery process itself that we assume provided by the middleware

Dynamic Reconfiguration

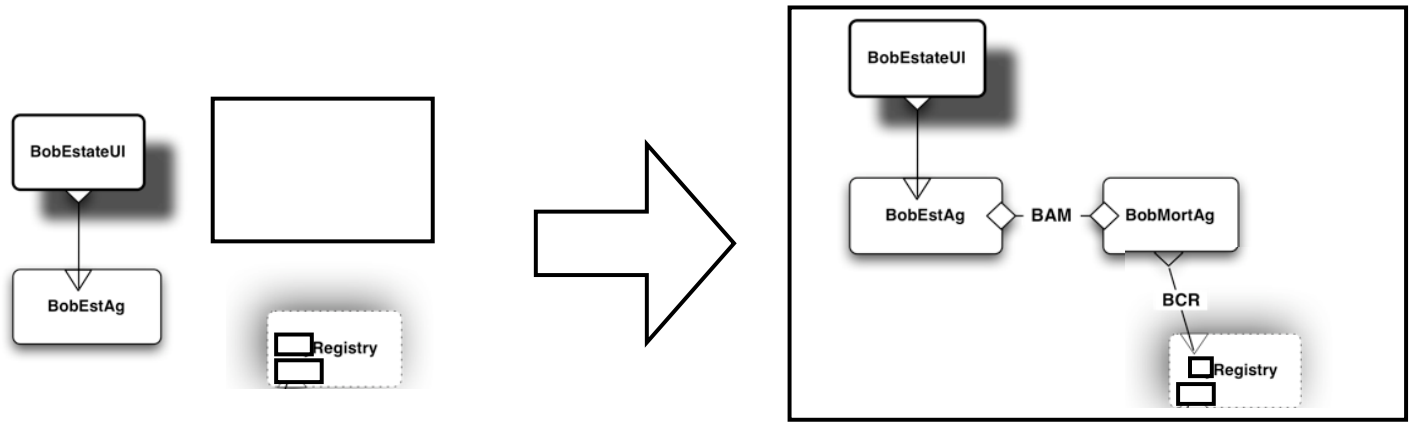


- The trigger (e.g., intMO) becomes true
- Discovery (in a repository, through a broker, etc.)
- Matchmaking (of Business Protocols giving syntactic and behavioural description)
- Ranking (External policies expressing SLA constraints)
- Selection
- Binding (Reconfiguration)

Dynamic Reconfiguration



... and binding



Internal Configuration Policies

- **Internal configuration policies**: concern aspects related with the instantiation of the module or those reconfiguration issues that do not involve negotiation with external parties
  - the initialization of service components (when declared in the module)

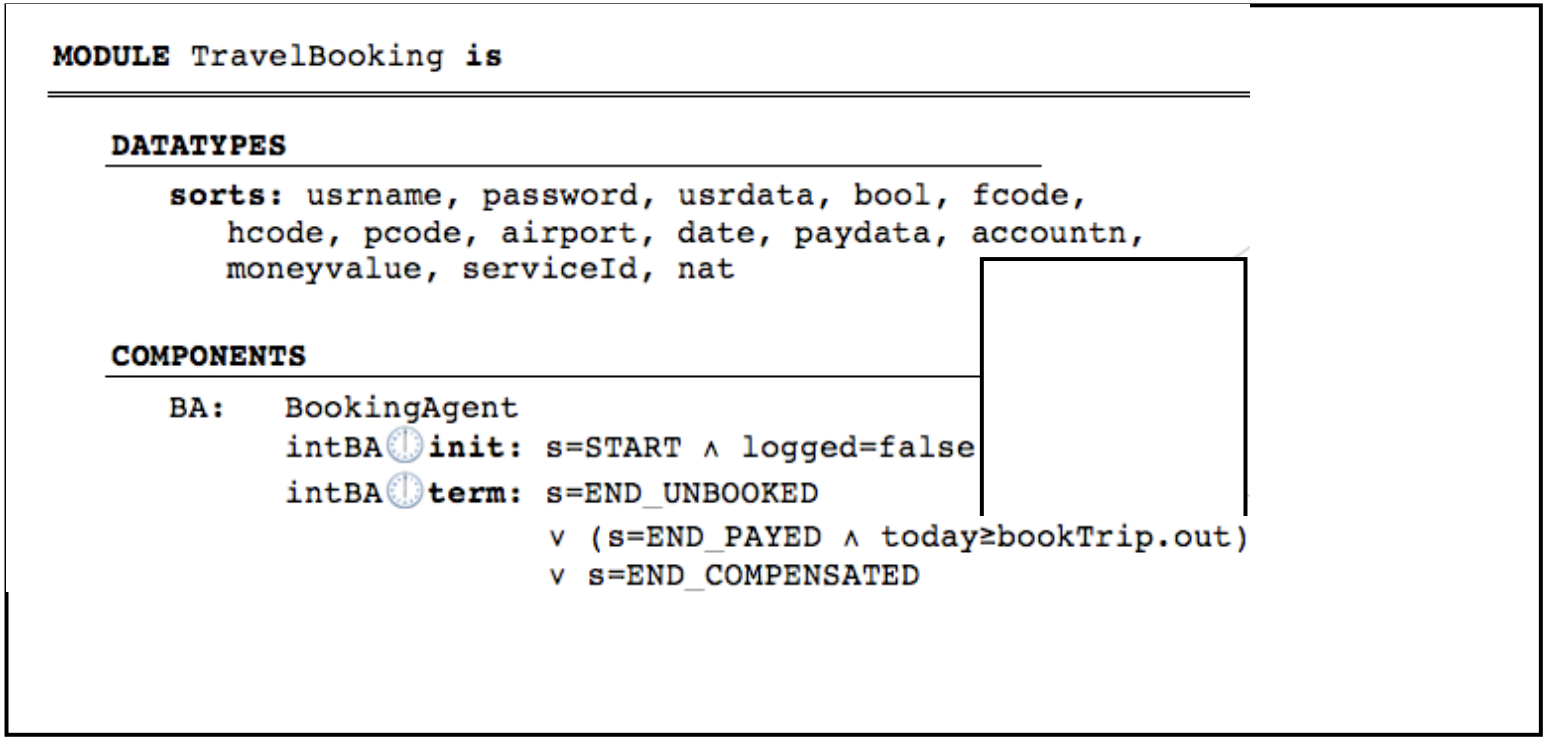
```
BA: BookingAgent
        intBA  🕐  init: S=START ∧ logged=false
        intBA  🕐  term: S=END_UNBOOKED ∨
                        (S=END_PAID ∧ today>=bookTrip.out) ∨
                         S=END_COMPENSATED
```

  - the triggering of the discovery of required services

```
FA: FlightAgent
      intFA   🕐   trigger: BA.bookFlight 🔔 ? (or default)
PA: PayAgent
      intPA   🕐   trigger: BA.bookFlight ✔ ?
HA: HotelAgent
      intHA   🕐   trigger: BA.bookFlight ✉ ? ∧ BA.bookFlight.Reply
```

The complete example:

- see notes page 50...



The complete example:

- see notes page 50...

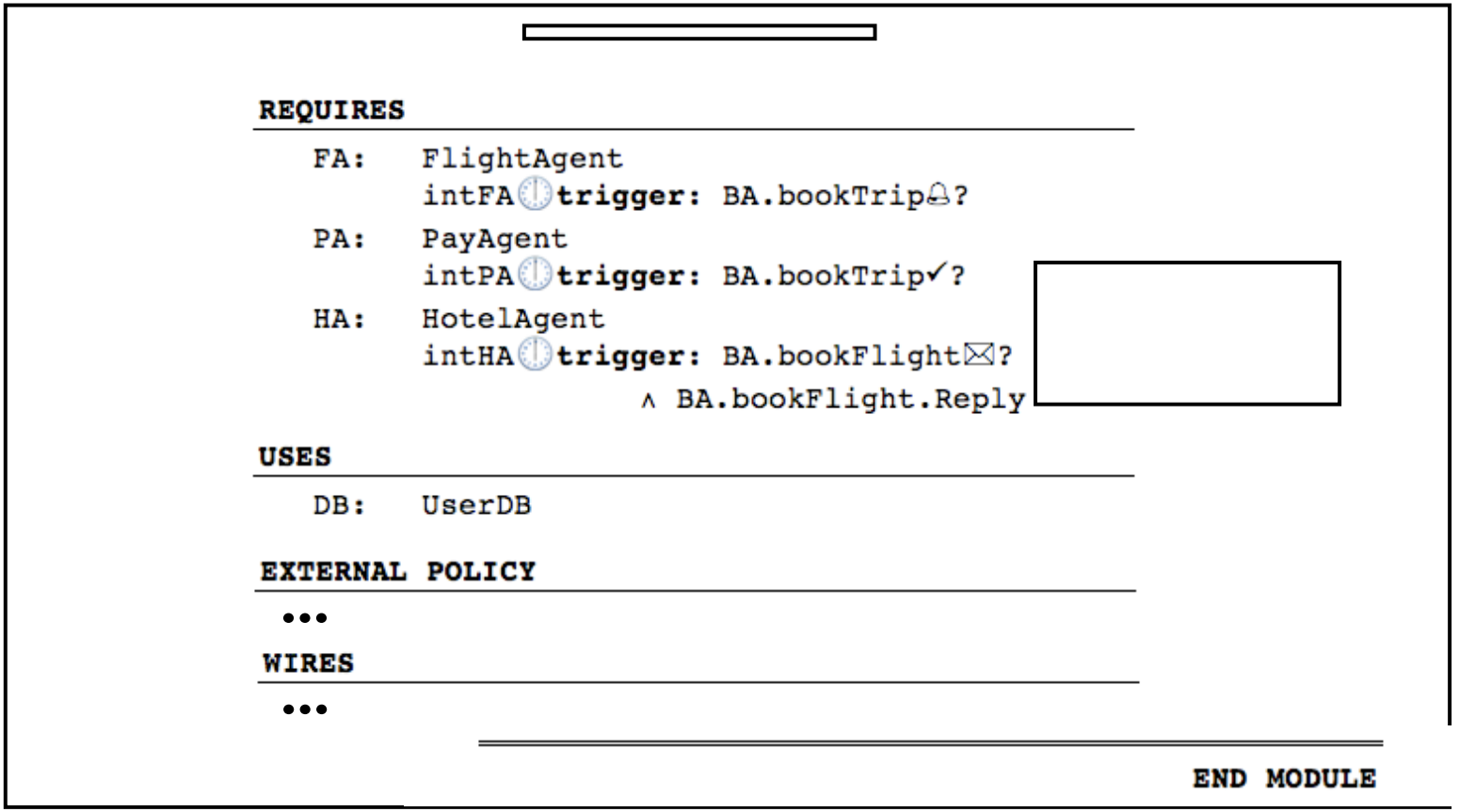
PROVIDES

CR: Customer

CR Customer	**BA** BookingAgent
r&s getProposal 🔔 idData income preferences ✉ proposal cost	**r&s** getProposal 🔔 idData income preferences ✉ proposal cost
r&s login 🔔 usr pwd	**r&s** login 🔔 usr pwd
snd ackRefundSnd 🔔 amount	**snd** ackRefundSnd 🔔 amount

The complete example:

- see notes page 50...



Just two notes about wires...

see notes page 50...

REQUIRES

FA: FlightAgent
intFA **trigger:** BA.bookTrip?

...

WIRES

BA BookingAgent	c_4	**BF**	d_4	**FA** FlightAgent
s&r bookFlight from to out in traveller fconf amount beneficiary payService	S_1 i_1 i_2 i_3 i_4 i_5 o_1 o_2 o_3 o_4	Straight. I(airport, airport, date, date, usrdata) O(fcode, moneyvalue, accountn, serviceId)	R_1 i_1 i_2 i_3 i_4 i_5 o_1 o_2 o_3 o_4	**r&s** lockFlight from to out in traveller fconf amount beneficiary payService
snd payAck proof status	S_1 i_1 i_2	Straight. I(pcode, bool)	R_1 i_1 i_2	**rcv** payAck proof status
rcv ackRefundRcv amount	R i_1	Straight. I(moneyvalue)	S i_1	**snd** payRefund amount

...

Just two notes about wires...

see notes page 50...

PROVIDES

CR: Customer

CR Customer	**BA** BookingAgent
r&s getProposal 🔔 idData income preferences ✉ proposal cost	**r&s** getProposal 🔔 idData income preferences ✉ proposal cost
r&s login 🔔 usr pwd	**r&s** login 🔔 usr pwd
snd ackRefundSnd 🔔 amount	**snd** ackRefundSnd 🔔 amount

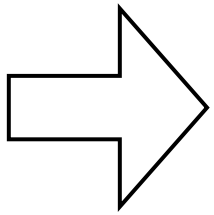
...

WIRES

c_1	**CB**	d_1	**BA** BookingAgent
S_1 i_1 i_2 i_3 i_4 o_1 o_2 o_3	Straight. I(airport, airport, date, date) O(fcode, hcode, moneyvalue)	R_1 i_1 i_2 i_3 i_4 o_1 o_2 o_3	**r&s** bookTrip 🔔 from to out in ✉ fconf hconf amount
S_1 i_1 i_2	Straight. I(username, password)	R_1 i_1 i_2	**r&s** login 🔔 usr pwd
R_1 i_1	Straight. I(moneyvalue)	S_1 i_1	**snd** ackRefundSnd 🔔 amount

...

Agenda



- Internal configuration policies
- External condiguration policies
- Non-functional properties and SLA

SLA in SRML

- SRML supports service selection based on QoS
- Model for non-functional requirements of a dynamically changing configuration
- QoS in SRML relies on
  - c-semirings to model satifiability
  - CSP to model the dynamic reconfiguration of constraints concerning QoS

S.Bistarelli, U. Montanari, F. Rossi (1997)
Semiring-based constraint satisfaction and optimization.
Journal of the ACM (JACM) 44(2): 201-236

External policies in SRML

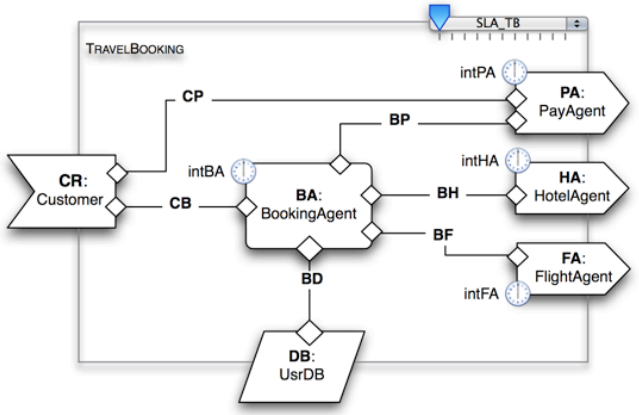
EXTERNAL POLICY

<[0..1],max,min,0,1>

SLA VARIABLES

HA.DIST2CENTRE, HA.DIST2METRO
FA.BOOKFEE, CR.BOOKFEE
CR.PERC, FA.PERC

CONSTRAINTS



C_1: {HA.DIST2CENTRE, HA.DIST2METRO}

$def_1(d,p)$= if d<1000 or p<100 then 1 otherwise 200/p

C_2: {CR.BOOKFEE}

$def_2(n)$= if n>5 then 1 otherwise 0

C_3: {CR.BOOKFEE, FA.BOOKFEE}

$def_3(d,p)$= if d>p the 1- 1/(d-p+1) otherwise 0

C_3: {CR.PERC, FA.PERC}

$def_3(a,b)$= if a=b then 1 otherwise 0

What is a c-semiring?

- A c-semiring is an algebraic structure $\langle A,+,\times,0,1\rangle$ where:
  - A is a set of values such that $\{0,1\}\in A$
  - + is a binary operation on A that is commutative, associative, idempotent and with unit element 0
  - $\times$ is another binary operation on A that is commutative, associative with unit element 1 and absorbing element 0
  - $\times$ distributes over +

A is the domain of the degree of satisfaction
- {0,1} for yes/no
- [0,1] for intermediate degrees

What is a c-semiring?

- A c-semiring is an algebraic structure $\langle A,+,\times,0,1\rangle$ where:
  - A is a set of values such that $\{0,1\}\in A$
  - $+$ is a binary operation on A that is commutative, associative, idempotent and with unit element 0
  - $\times$ is another binary operation on A that is commutative, associative with unit element 1 and absorbing element 0
  - $\times$ distributes over $+$

$+$ is a comparison primitive

$a<b \Leftrightarrow a+b=b$ (b is better than a)

$\langle\{0,1\},\vee,\wedge,0,1\rangle$
$\langle[0,1],max,min,0,1\rangle$

What is a c-semiring?

- A c-semiring is an algebraic structure $\langle A,+,\times,0,1\rangle$ where:
  - A is a set of values such that $\{0,1\}\in A$
  - + is a binary operation on A that is commutative, associative, idempotent and with unit element 0
  - $\times$ is another binary operation on A that is commutative, associative with unit element 1 and absorbing element 0
  - $\times$ distributes over +

$\times$ is a composition primitive

$\langle[0,1],\vee,\wedge,0,1\rangle$
$\langle\{0,1\},\max,\min,0,1\rangle$

SLA in SRML

- A constraint system is a triple < S, D, V > where
  - S is a C-semiring
  - D is a finite set (domain of possible elements taken by the variables)
  - V is a totally ordered set (of variables)
- A constraint is a pair < def, con > where
  - $con \subseteq V$ is called the type of the constraint
  - $def : D^{|con|} \rightarrow A$ is the value (mapping) of the constraint

$$\langle a_1, a_2, .., a_{|con|} \rangle$$

$$\downarrow$$

degree of satisfaction

SLA in SRML

- def : $D_{|con|} \rightarrow A$ is the value (mapping) of the constraint

$$\langle a_1, a_2, .., a_{|con|} \rangle$$

$$\downarrow$$

degree of satisfaction

- For example, if we have V = { cost , waitingTime} then def could map:

<50,1> -> 1 (if I pay 50 and I wait 1 day I am happy)
<50,2> -> 0.5 (If I wait 2 days buy then I pay only 50 I am happy)
<100,1> -> 0.5 (if I pay 100 and I wait 1 day I am half happy)
<0,7> -> 0 (if I have to wait more than 7 days I am unhappy, even if it is free)

- Defining **def** through enumeration could be not practical as there may be infinite values for the variables to consider
- We use functions

External policies in SRML

EXTERNAL POLICY

<[0..1],max,min,0,1>

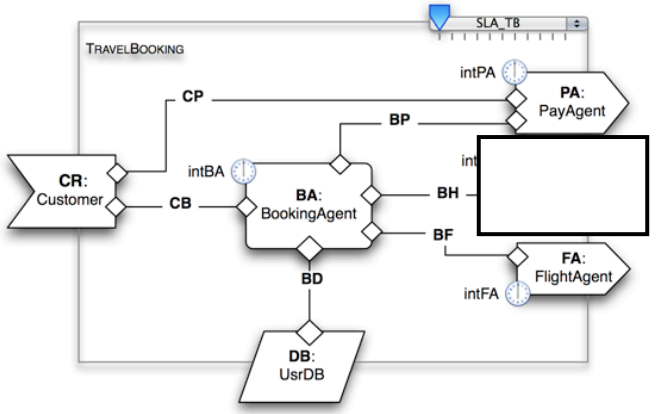
SLA VARIABLES

FA.BOOKFEE, CR.BOOKFEE

CR.PERC, FA.PERC

CONSTRAINTS

{HA.DIST2CENTRE, HA.DIST2METRO}



$def_1(d,p)$= if d<1000 or p<100 then 1 otherwise 200/p

If the hotel is less than one Km from the centre or less than 100 m from the metro station then the degree of satisfaction is 1 (maximal)

Otherwise the degree of satisfaction is inversely proportional to the distance from the metro station

External policies in SRML

EXTERNAL POLICY

<[0..1],max,min,0,1>

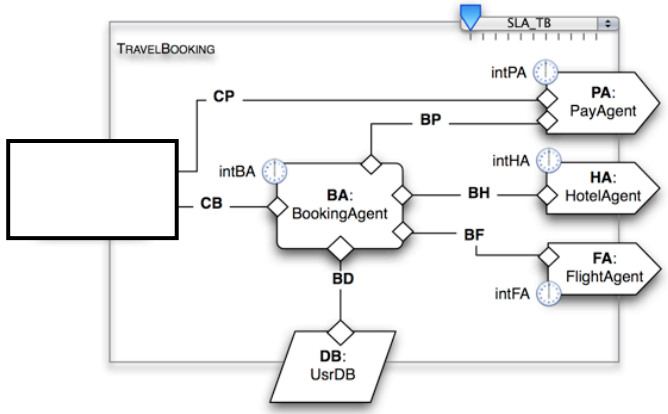
SLA VARIABLES

HA.DIST2CENTRE, HA.DIST2METRO

FA.BOOKFEE, [ ]

CR.PERC, FA.PERC

CONSTRAINTS



C_1: {HA.DIST2CENTRE, HA.DIST2METRO}

$def_1(d,p)$= if d<1000 or p<100 then 1 otherwise 200/p

[ ] {CR.BOOKFEE}

$def_2(n)$= if n>5 then 1 otherwise 0

The booking fee that the customer will agree to pay to TravelBooking must be greater than 5£

External policies in SRML

EXTERNAL POLICY

<[0..1],max,min,0,1>

SLA VARIABLES

HA.DIST2CENTRE, HA.DIST2METRO

CR.PERC, FA.PERC

CONSTRAINTS

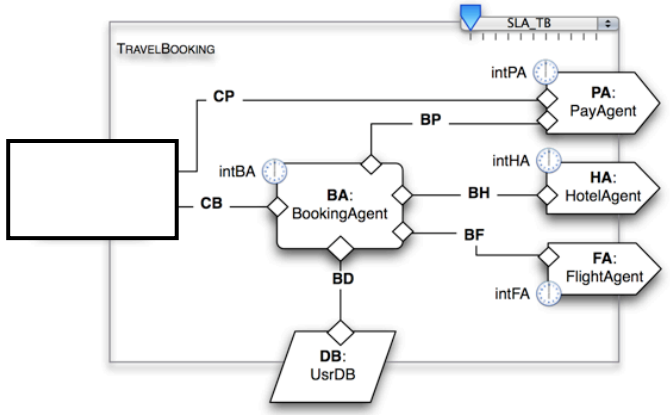
...

C_2: {CR.BOOKFEE}

$def_2(n)$= if n>5 then 1 otherwise 0

{CR.BOOKFEE, FA.BOOKFEE}

$def_3(d,p)$= if d>p the 1- 1/(d-p+1) otherwise 0



The booking fee of asked by the flight agent must be lower than the booking fee asked by TravelBooking to the customer.

Specifically, the degree of satisfaction of TravelBooking is directly proportional to the difference of the fee gained (from Customer) and the fee paid (fo FlightAgent).

External policies in SRML

EXTERNAL POLICY

<[0..1],max,min,0,1>

SLA VARIABLES

HA.DIST2CENTRE, HA.DIST2METRO

FA.BOOKFEE, CR.BOOKFEE

[ ]

CONSTRAINTS

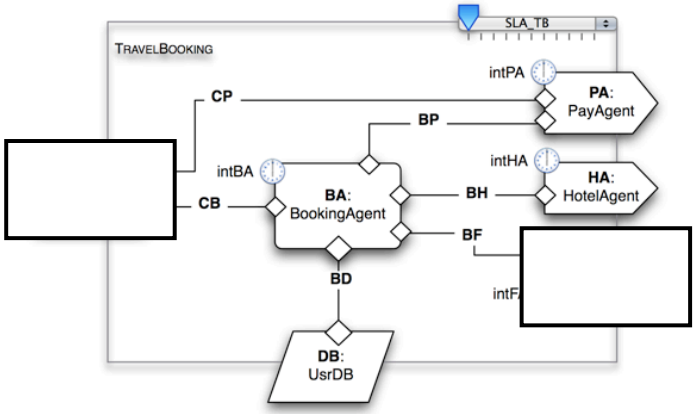
...

C_3: {CR.BOOKFEE, FA.BOOKFEE}

$def_3(d,p)$= if d>p the 1- 1/(d-p+1) otherwise 0

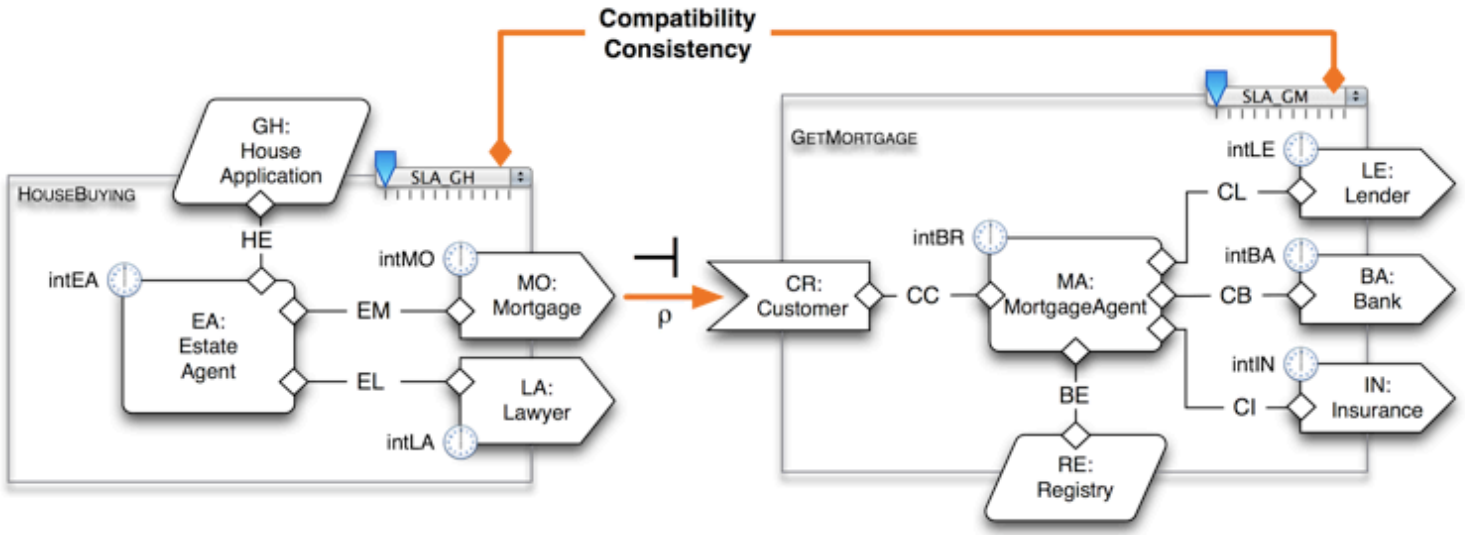
[ ] {CR.PERC, FA.PERC}

$def_3(a,b)$= if a=b then 1 otherwise 0



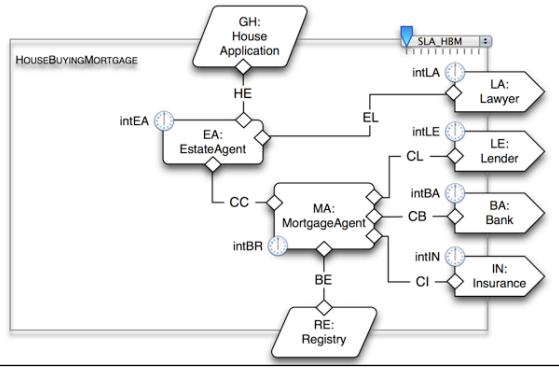
The percentage of refund promised to the customer must be the same as the one offered by the flight agent

Reconfiguration and SLA



$P1=\langle C_1, con_1 \rangle$ $P2=\langle C_2, con_2 \rangle$

- Compatibility checked by combining
  - the projection of P1 on the attributes of HA
  - the projection of P2 on the attributes of CS



$\langle C_1 \otimes C_2, con_1 \cup con_2 \rangle$

The complete example:

- **see notes page 50...**

```
BUSINESS PROTOCOL FlightAgent is
   INTERACTIONS
      r&s lockFlight
         🔔 from,to:airport,
            out,in:date,
            traveller:usrdata
         ✉ fconf:fcode
            amount:moneyvalue,
            beneficiary:accountn,
            payService:serviceId
      rcv payAck
         🔔 proof:pcode
            status:bool
      snd payRefund
         🔔 amount:moneyvalue
   BEHAVIOUR
      initiallyEnabled lockFlight🔔?
      (lockFlight✉! ∧ lockFlight.Reply) enables payAck🔔?
      (payAck🔔? ∧ payAck.status) enables lockFlight↟?
         until today≥lockFlight.out
      (lockFlight↟? ∧ today<lockFlight.out)
         ensures payRefund🔔!
```

- How should we change (restrict) the behaviour of FlightAgent to allow the compensation of lockFlight to be accepted only until 5 days before the trip?
- And to ensure the refund only if the compensation occurs more than 5 days before the trip?

The complete example:

- **see notes page 50...**

```
BUSINESS PROTOCOL FlightAgent is
INTERACTIONS
   r&s lockFlight
      🔔 from,to:airport,
         out,in:date,
         traveller:usrdata
      ✉ fconf:fcode
         amount:moneyvalue,
         beneficiary:accountn,
         payService:serviceId
   rcv payAck
      🔔 proof:pcode
         status:bool
   snd payRefund
      🔔 amount:moneyvalue

BEHAVIOUR
   initiallyEnabled lockFlight🔔?
   (lockFlight✉! ∧ lockFlight.Reply) enables payAck🔔?
   (payAck🔔? ∧ payAck.status) enables lockFlight✞?
      until today+5≥lockFlight.out
   (lockFlight✞? ∧ today+5<lockFlight.out)
      ensures payRefund🔔!
```

- Remember that we were negotiating the parameter FA.KD in the SLA ?
- And to ensure the refund only if the compensation occurs more than 5 days before the trip?

The complete example:

- **see notes page 50...**

BUSINESS PROTOCOL Customer **is**

INTERACTIONS

s&r login
- 🔔 usr:username, pwd:password

s&r bookTrip
- 🔔 from,to:airport, out,in:date
- ✉ fconf:fcode, hconf:hcode, amount:moneyvalue

rcv payNotify
- 🔔 status:bool

rcv refund
- 🔔 amount:moneyvalue

BEHAVIOUR

initiallyEnabled login🔔?
(login✉! ∧ login.Reply) **enables** bookTrip🔔?
(bookTrip⌛ ∧ bookTrip✓?) **ensures** payNotify🔔!
(payNotify🔔! ∧ payNotify.status) **enables** bookTrip✞?
until today≥bookTrip.out
(bookTrip✞? ∧ today<bookTrip.out) **ensures** refund🔔!

BUSINESS PROTOCOL FlightAgent **is**

INTERACTIONS

r&s lockFlight
- 🔔 from,to:airport, out,in:date, traveller:usrdata
- ✉ fconf:fcode amount:moneyvalue, beneficiary:accountn, payService:serviceId

rcv payAck
- 🔔 proof:pcode status:bool

snd payRefund
- 🔔 amount:moneyvalue

BEHAVIOUR

initiallyEnabled lockFlight🔔?
(lockFlight✉! ∧ lockFlight.Reply) **enables** payAck🔔?
(payAck🔔? ∧ payAck.status) **enables** lockFlight✞?
until today+5≥lockFlight.out
(lockFlight✞? ∧ today+5<lockFlight.out)
ensures payRefund🔔!

- Can BookTrip provide the business protocol Customer, relying on FlightAgent?
- Customer is allowed to compensate the flight, e.g., the day before the trip but the flight agent does not allow this and will not provide any refund
- In theory BookTrip could provide Customer BUT it should implement the orchestration accordingly (and pay the refund by itself if the customer compensates, e.g., the day before the trip!)

The complete example:

- see notes page 50...

BUSINESS PROTOCOL Customer **is**

INTERACTIONS

s&r login
🔔 usr:username, pwd:password
s&r bookTrip
🔔 from,to:airport,
out,in:date
✉ fconf:fcode,
hconf:hcode,
amount:moneyvalue
rcv payNotify
🔔 status:bool
rcv refund
🔔 amount:moneyvalue

BEHAVIOUR

initiallyEnabled login🔔?
(login✉! ∧ login.Reply) **enables** bookTrip🔔?
(bookTrip⌛ ∧ bookTrip✓?) **ensures** payNotify🔔!
(payNotify🔔! ∧ payNotify.status) **enables** bookTrip✞?
until today≥bookTrip.out+10
(bookTrip✞? ∧ today+10<bookTrip.out) **ensures** refund🔔!

BUSINESS PROTOCOL FlightAgent **is**

INTERACTIONS

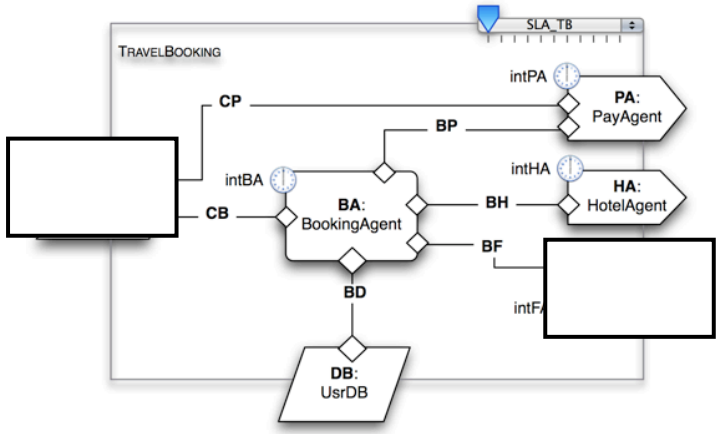
r&s lockFlight
🔔 from,to:airport,
out,in:date,
traveller:usrdata
✉ fconf:fcode
amount:moneyvalue,
beneficiary:accountn,
payService:serviceId
rcv payAck
🔔 proof:pcode
status:bool
snd payRefund
🔔 amount:moneyvalue

BEHAVIOUR

initiallyEnabled lockFlight🔔?
(lockFlight✉! ∧ lockFlight.Reply) **enables** payAck🔔?
(payAck🔔? ∧ payAck.status) **enables** lockFlight✞?
until today+5≥lockFlight.out
(lockFlight✞? ∧ today+5<lockFlight.out)
ensures payRefund🔔!

- Can BookTrip provide the business protocol Customer, relying on FlightAgent?
- Yes, because we provide a more restrictive condition to customer than the one we can rely on
- On the down side, we should not restrict Customer more than what if necessary otherwise the customer may choose another service that provides better conditions
- The properties (functional and non functional) provided and required should be well tuned.

External policies in SRML



EXTERNAL POLICY

<[0..1],max,min,0,1>

SLA VARIABLES

CR.KD, FA.KD,

...

CONSTRAINTS

C_3: {CR.KD,FA.KD}

$def_3(a,b)$= 1 if a=c and 0 otherwise

The values of **KD** provided to the customer has to be the same as the one agreed with the flight agent

The complete example:

- see notes page 50...

```
BUSINESS PROTOCOL Customer is
  INTERACTIONS
    s&r login
      🔔 usr:username, pwd:password
    s&r bookTrip
      🔔 from,to:airport,
         out,in:date
      ✉ fconf:fcode,
         hconf:hcode,
         amount:moneyvalue
    rcv payNotify
      🔔 status:bool
    rcv refund
      🔔 amount:moneyvalue
  SLA VARIABLES
    KD:[1..30]
  BEHAVIOUR
    initiallyEnabled login🔔?
    (login✉! ∧ login.Reply) enables bookTrip🔔?
    (bookTrip⌛ ∧ bookTrip✓?) ensures payNotify🔔!
    (payNotify🔔! ∧ payNotify.status) enables bookTrip✞?
      until
    (bookTrip✞? ∧                    ) ensures refund🔔!
```

```
BUSINESS PROTOCOL FlightAgent is
  INTERACTIONS
    r&s lockFlight
      🔔 from,to:airport,
         out,in:date,
         traveller:usrdata
      ✉ fconf:fcode
         amount:moneyvalue,
         beneficiary:accountn,
         payService:serviceId
    rcv payAck
      🔔 proof:pcode
         status:bool
    snd payRefund
      🔔 amount:moneyvalue
  SLA VARIABLES
    KD:[1..30]
  BEHAVIOUR
    initiallyEnabled lockFlight🔔?
    (lockFlight✉! ∧ lockFlight.Reply) enables payAck🔔?
    (payAck🔔? ∧ payAck.status) enables lockFlight✞?
      until
    (lockFlight✞? ∧                    )
      ensures payRefund🔔!
```

- We can use the SLA variables in the business protocols.
- The value of CR.KD is defined when the instance of BookTrip is created (and bound to the service/ activity of the customer)
- The value of FA.KD is defined when a flight service is discovered, selected and instantiated
  - as soon as BookTrip is intantiated (if the trigger of FA is defined as "true")
  - at the first attempt of interaction of BookTrip with the flight agent (if the trigger is "default")
  - whenever the trigger of FA becomes true...

The complete example:

- **see notes page 50...**

```
BUSINESS ROLE BookingAgent is
  INTERACTIONS
    …
  SLA VARIABLES
    KD:[1..30]
  ORCHESTRATION
    local s:[START, LOGGED, QUERIED, FLIGHT_OK, HOTEL_OK, CON-
    FIRMED, END_PAID, END_UNBOOKED, COMPENSATING, END_COMPENSATED],
    logged:bool, traveller:usrdata, travcard:paydata

    …
    transition TripCompensate
      triggeredBy bookTrip↑
      guardedBy s=END_PAID ∧ today<bookTrip.out+KD
      effects s'=COMPENSATING
      sends bookFlight↑ ∧ bookHotel↑
    transition TripRefund
      triggeredBy ackRefundRcv🔔
      guardedBy s=COMPENSATING
      effects s'=END_COMPENSATED
      sends ackRefundSnd🔔
      ∧ ackRefundSnd.amount=ackRefundRcv.amount
```

- The business role of BookingAgent, that orchestrate the interactions between the customer and the flight agent has to be "tuned" with the business protocols
- Also business role can depend on SLA variables

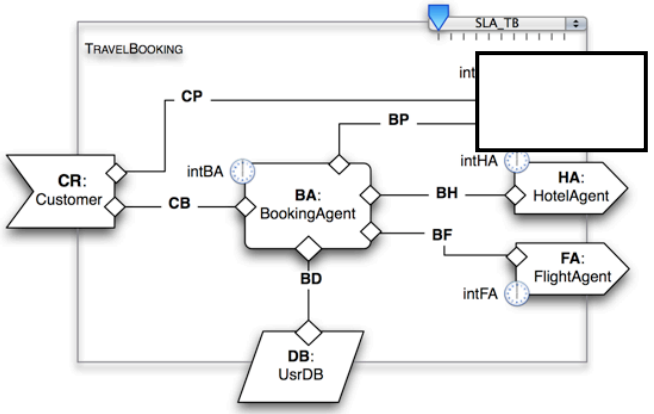
External policies in SRML

```
BUSINESS PROTOCOL FlightAgent is
  INTERACTIONS
    r&s lockFlight
      🔔 from,to:airport,
         out,in:date,
         traveller:usrdata
      ✉ fconf:fcode
         amount:moneyvalue,
         beneficiary:accountn,
         payService:serviceId
    ...
```

ServiceId is the service identifier of PA (similar to an URI)

The value of ServceId is communicated by FA during the orchestration

BookTrip does not perform an actual discovery but binds to the pay agent specified by the flight agent



EXTERNAL POLICY

<[0..1],max,min,0,1>

SLA VARIABLES

PA.ServiceID

CONSTRAINTS

C_2: {PA.ServiceId}

$def_2(n)$= 1 if n=FA.lockFlight.payService and 0 otherwise

Problems

EXTERNAL POLICY

<[0..1],max,min,0,1>

SLA VARIABLES

HA.DIST2CENTRE, HA.DIST2METRO, FA.BOOKFEE, CR.BOOKFEE, CR.PERC, FA.PERC

CONSTRAINTS

...

C_2: {CR.BOOKFEE} $def_2(n)$= if n>5 then 1 otherwise 0

C_3: {CR.BOOKFEE, FA.BOOKFEE} $def_3(d,p)$= if d>p the 1- 1/(d-p+1) otherwise 0

- **Change the business role BookingAgent to let customer pay CR.BOOKFEE to the payagent (ignore the problem of "distributing" the amount between TravelBooking and FlightAgent)**

Problems

EXTERNAL POLICY

<[0..1],max,min,0,1>

SLA VARIABLES

HA.DIST2CENTRE, HA.DIST2METRO, FA.BOOKFEE, CR.BOOKFEE, CR.PERC, FA.PERC

CONSTRAINTS

...

C_2: {CR.BOOKFEE} $def_2(n)$= if n>5 then 1 otherwise 0

C_3: {CR.BOOKFEE, FA.BOOKFEE} $def_3(d,p)$= if d>p the 1- 1/(d-p+1) otherwise 0

- **Change the business protocol Customer to ensure that the parameter amount of refund is equal to the amount of the trip minus the booking fee (which is not refunded)**

Problems

EXTERNAL POLICY

<[0..1],max,min,0,1>

SLA VARIABLES

HA.DIST2CENTRE, HA.DIST2METRO, FA.BOOKFEE, CR.BOOKFEE, CR.PERC, FA.PERC

CONSTRAINTS

...

C_2: {CR.BOOKFEE} $def_2(n)$= if n>5 then 1 otherwise 0

C_3: {CR.BOOKFEE, FA.BOOKFEE} $def_3(d,p)$= if d>p the 1- 1/(d-p+1) otherwise 0

- Which value is negotiated/ defined first: CR.BOOKFEE or FA.BOOKFEE?
- How to change to internal reconfiguration policies to negotiate them together?

REQUIRES

FA: FlightAgent
intFA **trigger:** default

PA: PayAgent
intPA **trigger:** BA.bookTrip✓?

HA: HotelAgent
intHA **trigger:** BA.bookFlight✉?
∧ BA.bookFlight.Reply

Problems

EXTERNAL POLICY

<[0..1],max,min,0,1>

SLA VARIABLES

HA.PETS, FA.PETS, HA.PETS, FA.MILESPROGRAM, CR.MILESPROGRAM, HA.MILESPROGRAM

CONSTRAINTS

Problems

EXTERNAL POLICY

<[0..1],max,min,0,1>

SLA VARIABLES

HA.PETS, FA.PETS, HA.PETS, FA.MILESPROGRAM, CR.MILESPROGRAM, HA.MILESPROGRAM, HA.BOOKINGFEE

CONSTRAINTS

- if the customer has pets it is mandatory that FA and HA accept pets (FA.PETS=true and HA.PETS=true). If the customer does not have pets then the satisfaction is maximal in either case.
- the miles program of the customer must be the same as the miles program of the flight agent
- if the miles program of the hotel is not as the miles program of the satisfaction is inversely proportional to the booking fee (HO.BOOKINGFEE)

Problems

EXTERNAL POLICY

<[0..1],max,min,0,1>

SLA VARIABLES

HA.PETS, FA.PETS, HA.PETS, FA.MILESPROGRAM, CR.MILESPROGRAM, HA.MILESPROGRAM, HA.BOOKINGFEE

CONSTRAINTS

C_1: {CR.PETS, FA.PETS, HA.PETS}

$def_1(a,b,c)$= if a=true then (if b=c=true then 1 otherwise 0)
otherwise 1

C_2: {CR.MILESPROGRAM, FA.MILESPROGRAM}

$def_2(a,b)$= if a=b then 1
otherwise 0

C_3: {CR.MILESPROGRAM, HA.MILESPROGRAM, HA.BOOKINGFEE}

$def_2(a,b,c)$= if a=b then 1
otherwise 1/c+1