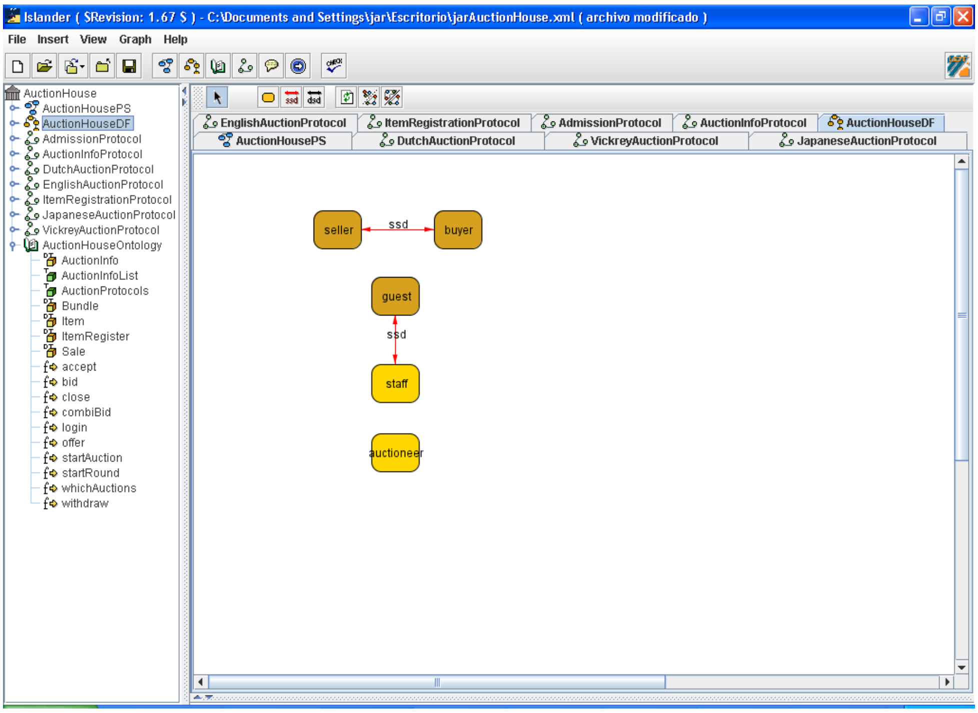
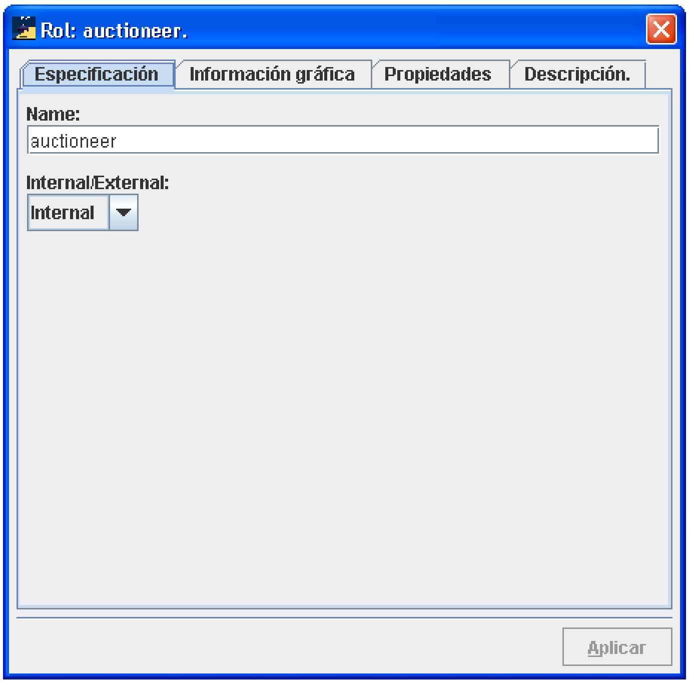
# ISLANDER TUTORIAL

## SOLUTIONS

3) Roles in the DF: staff, guest, auctioneer, buyer, seller



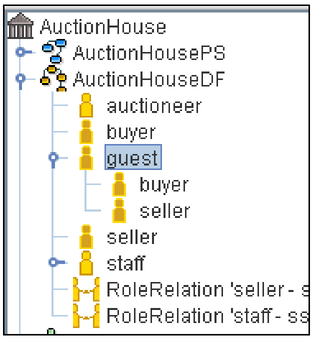
4) Internal: staff, auctioneer. External: guest, buyer, seller.



Name:
auctioneer

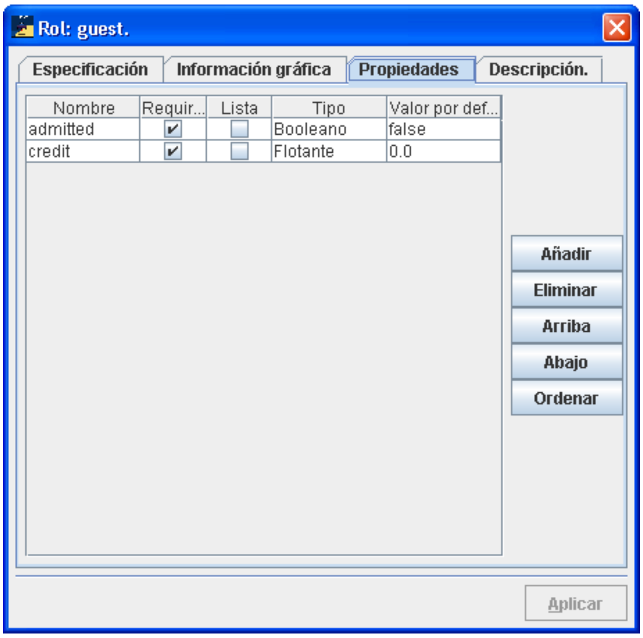
Internal/External:
Internal

5) Ssd between buyer and seller. Ssd between guest and staff. Improvements: The first relationship could be dsd.
6) Seller is a sub-role of guest.



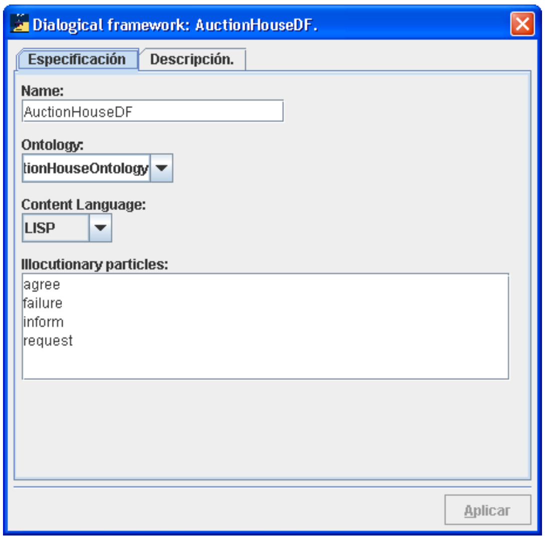
AuctionHouse
AuctionHousePS
AuctionHouseDF
auctioneer
buyer
guest
buyer
seller
seller
staff
RoleRelation 'seller - s
RoleRelation 'staff - ss

7) The guest role has de *admiited* and *credit* properties that are inherited by the buyer and seller roles. Further properties could be the list of goods to sell (for sellers) and the purchase list (for buyers).



| Nombre | Requir... | Lista | Tipo | Valor por def... |
|---|---|---|---|---|
| admitted | ☑ | □ | Booleano | false |
| credit | ☑ | □ | Flotante | 0.0 |

8) The dialogic framework is composed of an ontology, a set of illocutionary particles, a content language, and a set of roles along with their relationships.



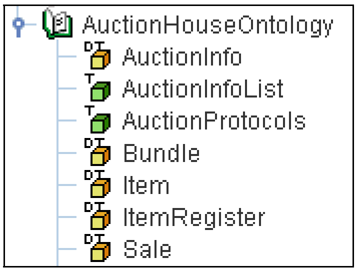
Name: AuctionHouseDF

Ontology: tionHouseOntology

Content Language: LISP

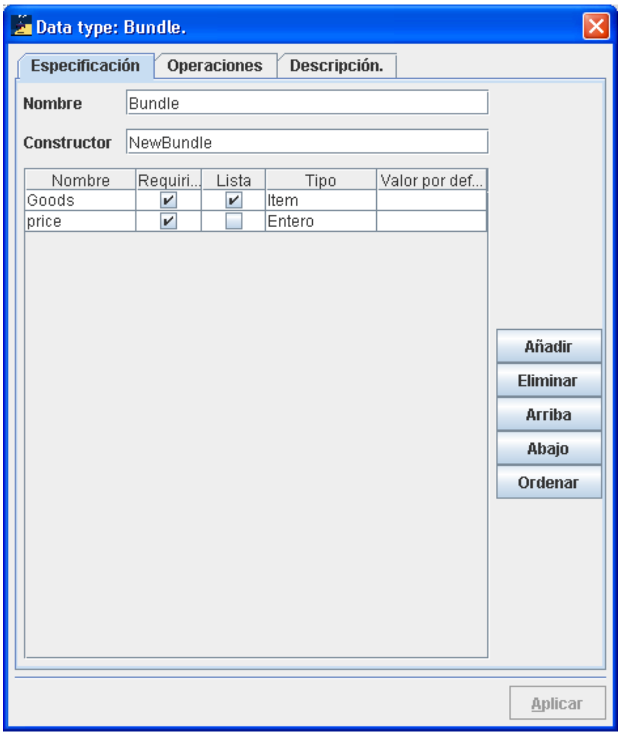
Illocutionary particles:
agree
failure
inform
request

9) The *AuctionHouseOntology* contains the following data types: *AuctionInfo*, *Item*, *Sale*, *ItemRegister*.



AuctionHouseOntology
- AuctionInfo
- AuctionInfoList
- AuctionProtocols
- Bundle
- Item
- ItemRegister
- Sale

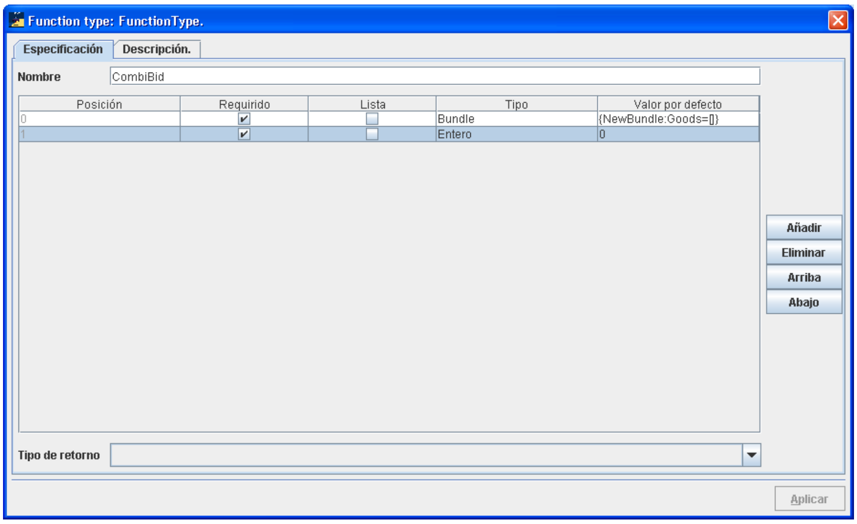
10) No, they are not. Define the *Bundle* data type as follows.



Nombre: Bundle

Constructor: NewBundle

| Nombre | Requiri... | Lista | Tipo | Valor por def... |
|---|---|---|---|---|
| Goods | ☑ | ☑ | Item | |
| price | ☑ | ☐ | Entero | |

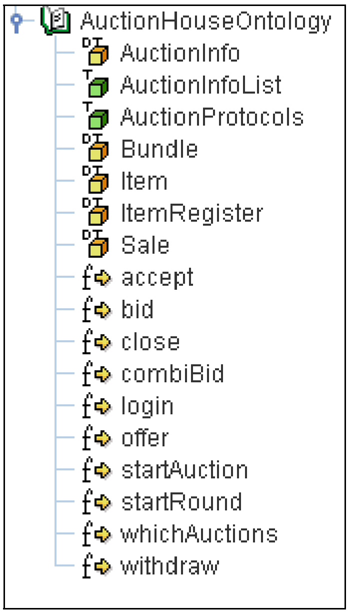


Nombre: CombiBid

| Posición | Requirido | Lista | Tipo | Valor por defecto |
|---|---|---|---|---|
| 0 | ☑ | ☐ | Bundle | {NewBundle:Goods=[]} |
| 1 | ☑ | ☐ | Entero | 0 |

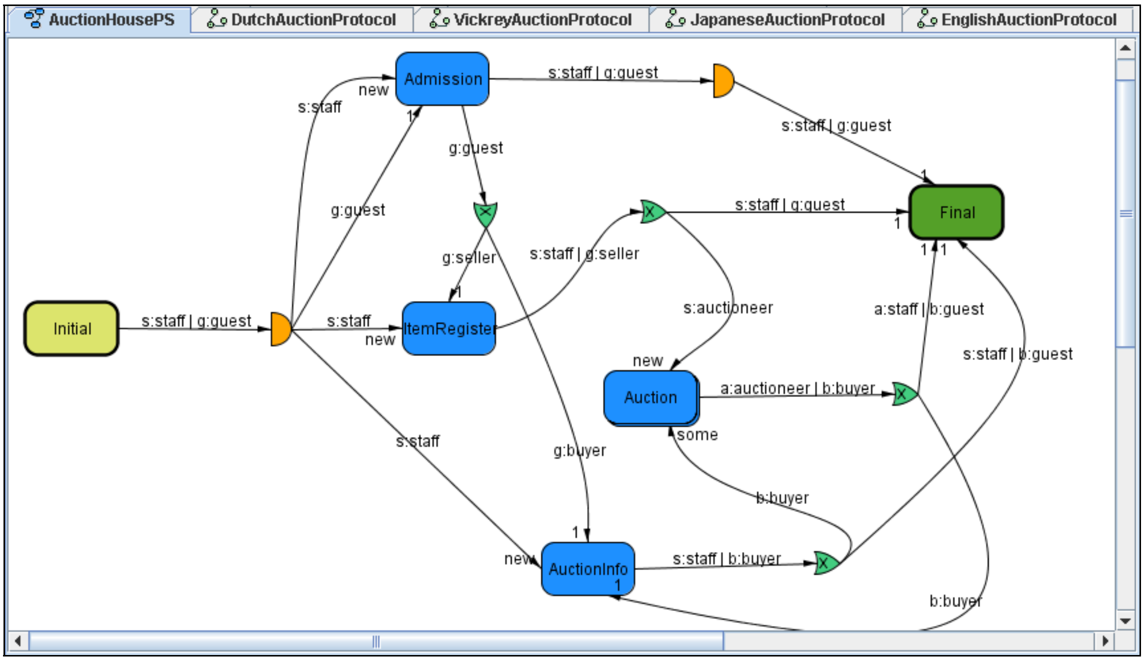
Tipo de retorno:

11) The functions are: *accept*, *bid*, *close*, *login*, *offer*, *startAuction*, *startRound*, *whichAuctions*, *withdraw*. They stand for the content language employed in the institution.

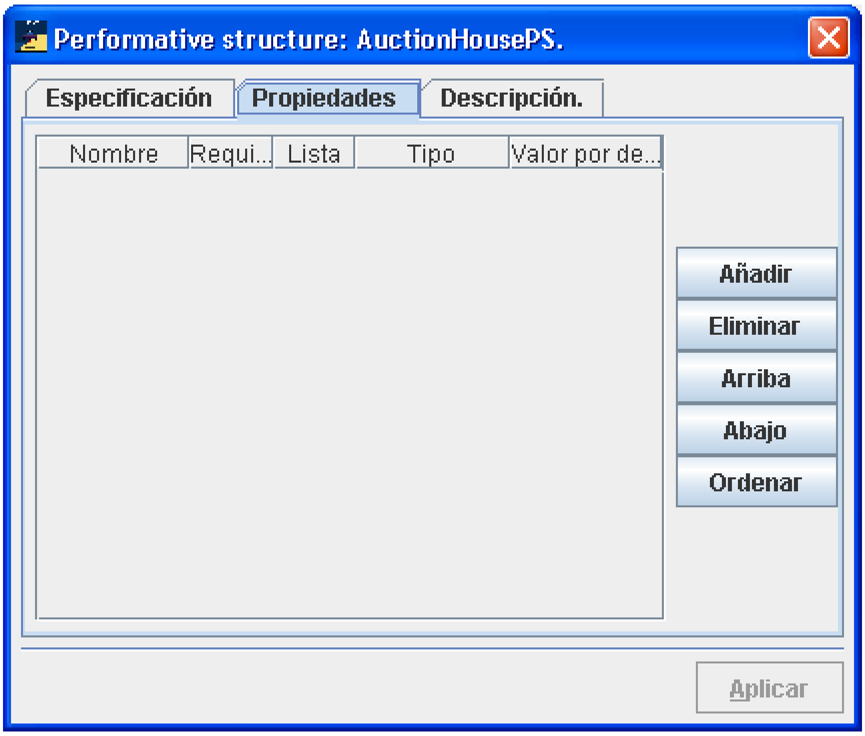


AuctionHouseOntology
- AuctionInfo
- AuctionInfoList
- AuctionProtocols
- Bundle
- Item
- ItemRegister
- Sale
- accept
- bid
- close
- combiBid
- login
- offer
- startAuction
- startRound
- whichAuctions
- withdraw

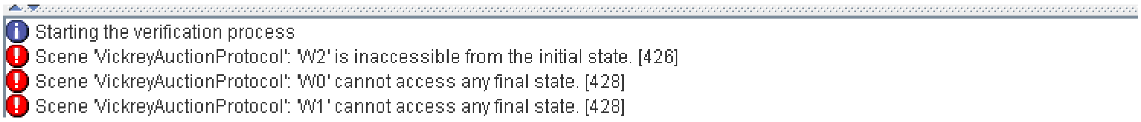
12) It is the network of scenes and transitions describing the activities in the institution.



13) None according to the current definition.

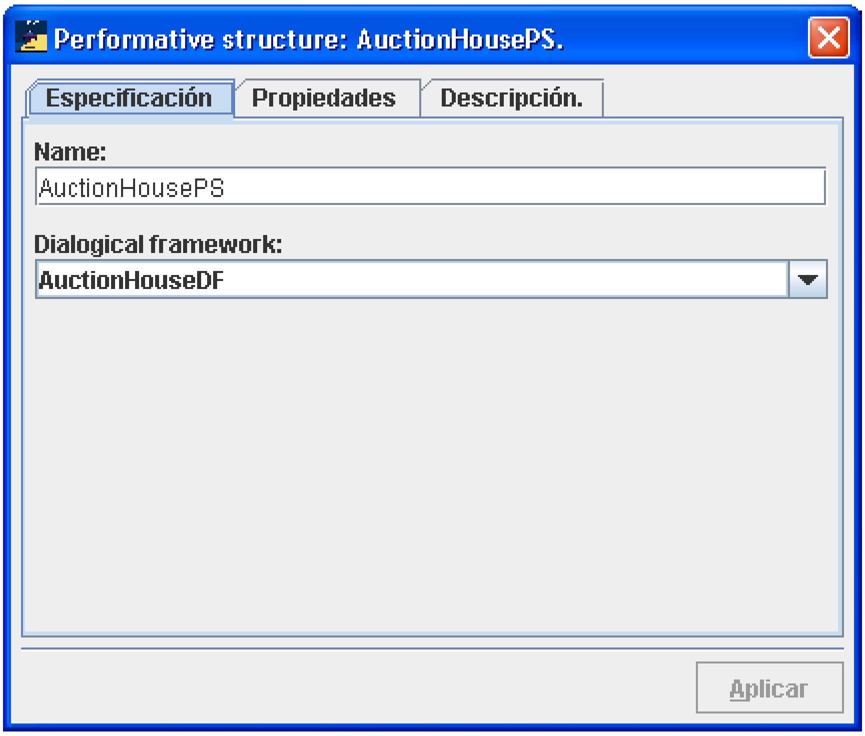


14) It is correct indeed. Try to remove elements in the specification and run the checker again. If you come up with any errors, click on them to precisely locate them.



```
Starting the verification process
Scene 'VickreyAuctionProtocol': 'W2' is inaccessible from the initial state. [426]
Scene 'VickreyAuctionProtocol': 'W0' cannot access any final state. [428]
Scene 'VickreyAuctionProtocol': 'W1' cannot access any final state. [428]
```

15) AuctionHouseDF.

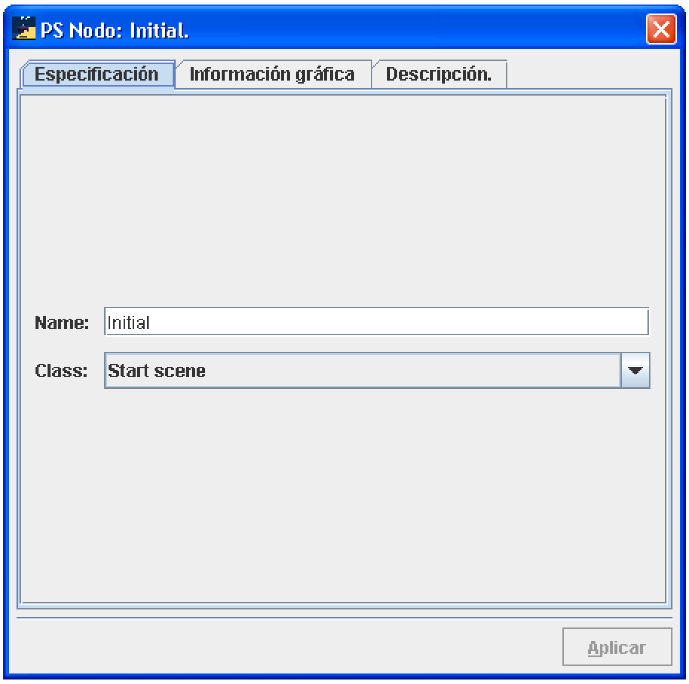


Name:
AuctionHousePS

Dialogical framework:
AuctionHouseDF

16) The scenes are: *Initial*, *Final*, *Admission*, *ItemRegister*, *AuctionInfo*, *Auction*.

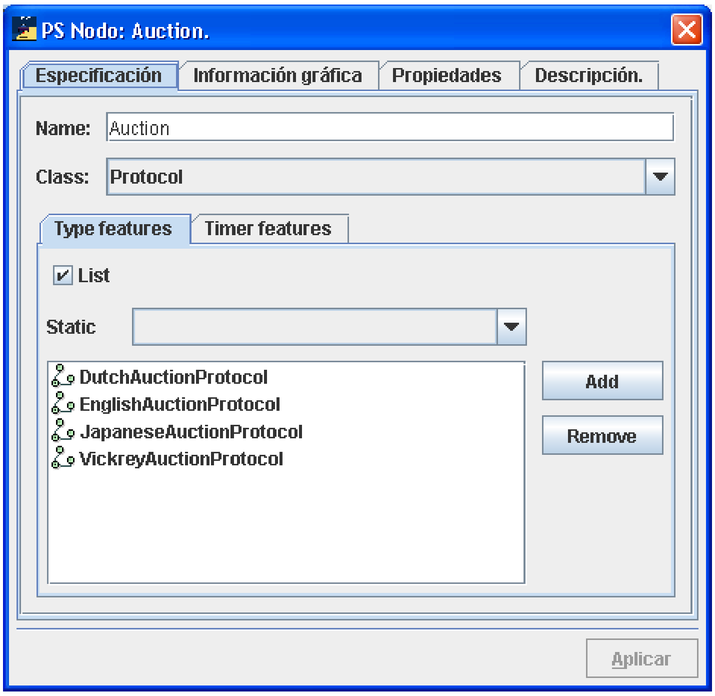
17) There is no protocol (no activity) for the initial and final scenes as shown below.



Name: Initial

Class: Start scene

18) Indeed, notice that several auction protocol specifications can be employed to run an *Auction* scene as shown below.



Name: Auction

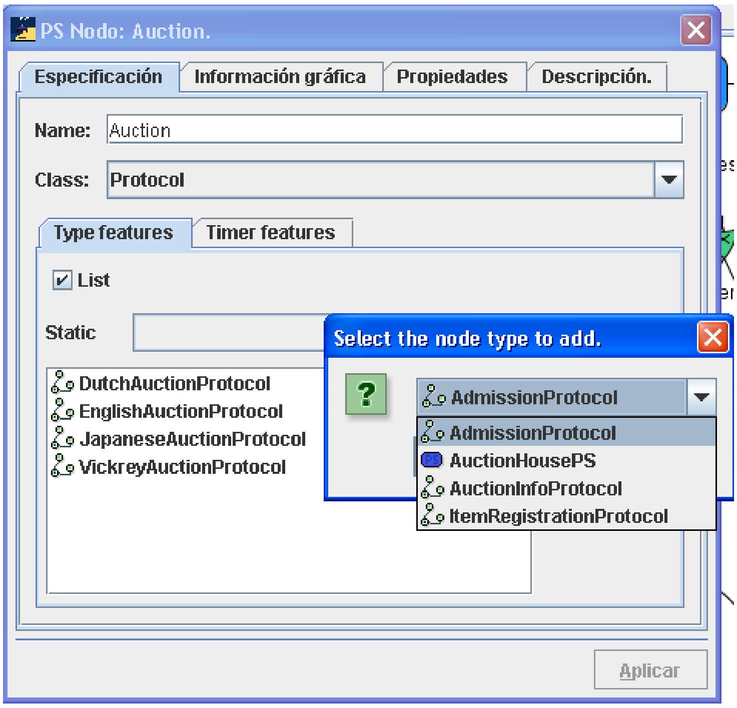
Class: Protocol

☑ List

Static

DutchAuctionProtocol
EnglishAuctionProtocol
JapaneseAuctionProtocol
VickreyAuctionProtocol

Inded, performative structures can be nested. Nodes in the peformative structure can be either specified as protocols or performative structures or both.



PS Nodo: Auction.

Name: Auction

Class: Protocol

☑ List

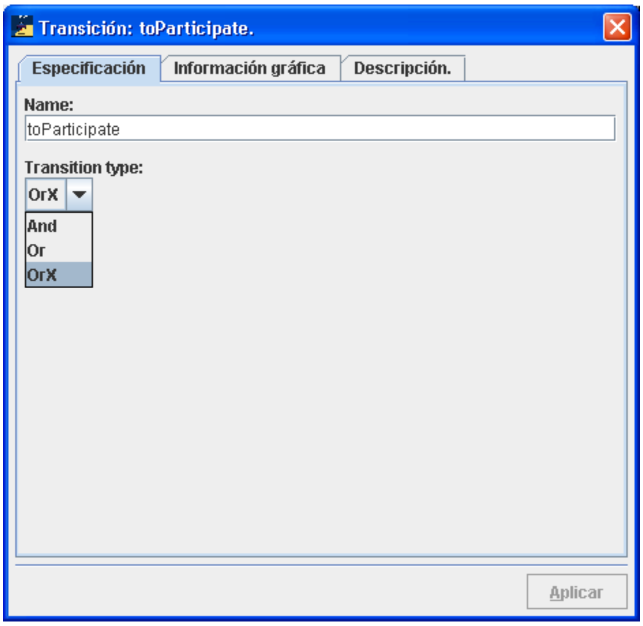
Static

DutchAuctionProtocol
EnglishAuctionProtocol
JapaneseAuctionProtocol
VickreyAuctionProtocol

Select the node type to add.

AdmissionProtocol

19) A transition in a performative structure can be of one out of three types: *And*, *Or*, *OrX*. In this particular specification, we have only employed *And* and *OrX* transitions.

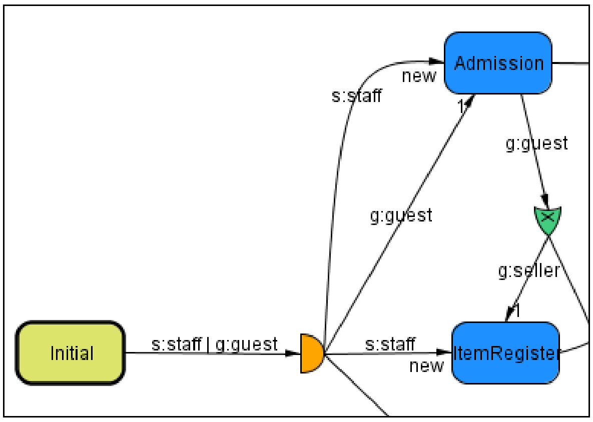


Transición: toParticipate.

Name: toParticipate

Transition type: OrX

20) We find the possible paths each role may take by following the arcs, labelled by agent variables and roles, connecting scenes with transitions at the performative structure level. The figure below shows that a *staff* agent can go from the *initial* scene in parallel towards the *ItemRegister* and *Admission* scenes.

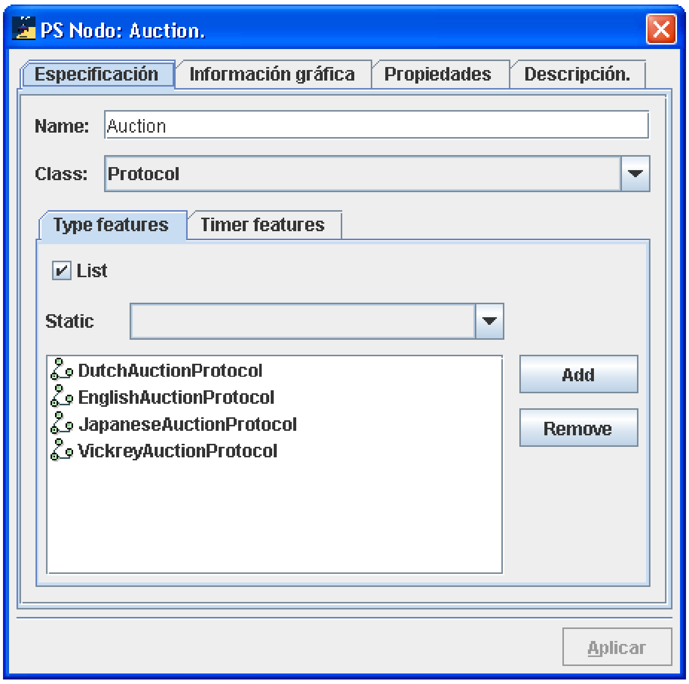


21) In what follows, we limit our discussion to the eventual improvements to the current specifications of the auction protocols in the auction market:
    - Dutch
        - Edge 5. The price can be lowered by the auctioneer whenever it is above the reserve price. This condition is not actually checked. Therefore the condition “?price > !ItemRegister.reservePrice”. Notice though that the “reservePrice” field has not been defined for “ItemRegister”. Notice too that the current specification employs “Item” instead of “ItemRegister”.
        - Edge 6. The auctioneer can withdraw a good whenever “!price – decrement > !ItemRegister.reservePrice”. This condition is not specified and thus should be added.
        - Edge 4. According to its specification, the first bidder submitting a bid would be the winner. Therefore, collisions cannot occur. However, collisions are interesting because the auctioneer raises the price after a collision to generate more competition since at least two bidders were interested for the good at the very same price. Therefore, in order to support collisions we can add a loop from w5 to w5 to collect more bids till “!waitTime” expires. If the number of bidders is greater than two (“?winner(w3,w5) >= 2”) a new transition from w5 to w3 is added to declare that a collision happened, and the round is subsequently restarted at a higher price.
    - English
        - Edge 2. The price increase is hardwired. We could let it free to the auctioneer discretion, or even constrain it under a certain value.
        - Notice that the typical knocking is not specified. It can be easily specified by adding transitions from w7 that successively allow the auctioneer to “knock” and wait for a given amount of time till the next knocking.
    - Japanese
        - Edge 10. The good is sold at a wrong price since one of the preconditions requires that “?sale.price == (!price – increment)”. It should be “?sale.price == !price” since the last remaining bidder did accept the last offer called by the auctioneer.
        - Edge 11. Although there were no bidders at the last price called by the auctioneer, there were bidders in a former round. Therefore, the auction should not be closed without a winner! Thus we must replace the current withdrawal with a sale along with the following

preconditions “?sale.price == (!price – increment)”, “?sale.buyer in lastBidBuyers”, “?sale.item == !item”.

- Vickrey
  - Notice that the winning bidder pays the price he offered. Therefore, the specification in this example is a variation of a first-price sealed bid auction protocol, instead of a Vickrey auction protocol as expected.

22) The fact that the node is specified as a *List* as shown below.



Name: Auction

Class: Protocol

☑ List

Static

DutchAuctionProtocol
EnglishAuctionProtocol
JapaneseAuctionProtocol
VickreyAuctionProtocol