Update on OWL-S Process Model for Release 1.1

Mark H. Burstein

BBN Technologies

May 25, 2004

Changes in Process Model

- Compliance with OWL-DL
- ‘bundled’ results include effects and outputs (knowledge effects) associated with a particular result condition
- Data flow model used to relate input, output parameters of different steps in composite processes
- Conditions (pre, result) are SWRL+DRS expressions included as XMLLiterals

Details

- Input, Output, Local parameters now subclass swrl:variable
  - parameterType property (range XMLLiteral)
  - parameterValue – literal representing constant or expression in terms of other parameters (useful in composite processes)
- Process
  - hasParticipant
  - hasPrecondition <swrl expression as XMLLiteral>
  - hasResult
    - Result
      - inCondition <parameterized swrl expression as XMLLiteral>
      - hasEffect <parameterized swrl expression>
      - withOutput – parameterized OWL description representing semantics of response in terms of process parameters, precondition and result variables

OWL-S Surface Syntax

```
define atomic process BuyBook (Client - Agent Title - String Author - String
                                Ccno - String CcExp - MoYr) ;; input parameters
        -> (BuyOutput)                                      ;; output parameters
local (?book - Book ?cc - CreditCard)      ;; defines local parameters
precondition (and (author-of ?book ?Author)(title-of ?book ?Title)
                  (ccNumber ?cc ?Ccno) (ccExpires ?cc ?CcExp))
result when (?bal - Number  ?price - Number ?transID - Number)  ;; result variables
     (and (instock ?book)  (ccBalance ?cc ?bal)                 ;; result condition
          (bookprice ?book ?price)  (> ?price ?bal)))
   => (and (debited ?cc ?price)(owns ?Client ?book)(shipto ?book ?Client) )   ;; effect
   output BuyOutput = (ConfirmedSale :to ?Client :of ?book      ;; output form
                                     :cost ?price :confirmation ?transID);
result when (not (instock ?book))                   ;; exception result conditions
    output BuyOutput = (NotInStock ?book)
result when (exists ... (not (> ?price ?bal)))
   output BuyOutput = (InsufficientFunds ?Ccno)
```

Composite Processes

- **Perform** construct used to describe a composite process reference to another process
  - *process* property refers to performed Process
  - Describe source of inputs (parameters of other Performs in the same composite process)
    - *hasBinding* (a Binding)
    - references to Parameter of the Process to be “set”, OWL expression in terms of dataflow links to other related Performs.
    - Uses same control constructs as OWL-S 1.0

Composite Process

```
define composite process S4tP()
  {
   intake::   receive(openSession());
   initiate:: create_session_key();
              send(dest <= intake.sender,
                   msg <= initia.key);
  first_finishes
     {
      parallel
       { { receive(book_ticket(initiate.key, D3));
           ---book the airline---;
           send(dest <= intake.sender,
                  msg <= confirm_status(initiate.key, ...)) }
         || {iterate
            { receive(book_event(initiate.key, D2));
              ---book the event described by D2---;
              send(dest <= intake.sender,
                     msg <= confirm_status(initiate.key, ...))       } }
         ||  { receive(book_hotel(initiate.key, D1));
               ---book the hotel---;
              send(dest <= intake.sender,
                     msg <= confirm_status(initiate.key, ...)) }  }
      ||   receive(terminate(initiate.key)) } }
```