Multirôle Session Types

Sophia Drossopoulou
Imperial College London

very preliminary, and ongoing, joint work with
Azalea Raad and Susan Eisenbach

based on earlier work with
Elena Giachino, Matthew Sackman, and Susan Eisenbach

Relationships, Objects, Roles, and Queries in Modern Programming

- Researchers in programming language design wish to bridge the gap between implementation and design. ...
- Researchers in program analysis are interested in raising the level of abstraction in programming languages.....
- Researchers in databases are keen to bridge the ``impedance mismatch'' between programs and databases...
- In the absence of language support for querying, many commonly used operations need to be expressed using nested loops and complex conditional statements. ...
- Researchers in modelling languages and knowledge representation want to clarify whether roles and relationships are dual notions,....

Relationships, Objects, Roles, and Queries in Modern Programming

- Researchers in programming language design wish to bridge the gap between implementation and design. ...
- Researchers in program analysis are interested in raising the level of abstraction in programming languages.....
- Researchers in databases are keen to bridge the ``impedance mismatch'' between programs and databases...
- In the absence of language support for querying, many commonly used operations need to be expressed using nested loops and complex conditional statements. ...
- Researchers in modelling languages and knowledge representation want to clarify whether roles and relationships are dual notions,....

Overview

- Session Types, and multiparty session types
- Multirole session types
- Design Issues & Further Work

Session Types and Multiparty Session Types

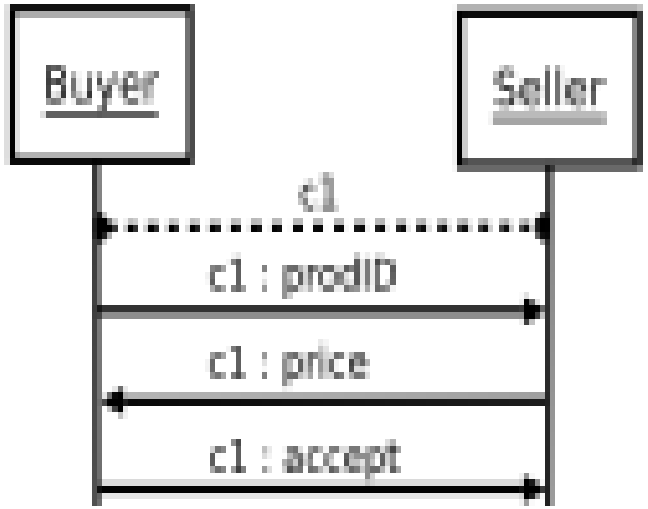
Session Types - 1

K. Honda. Types for dyadic interaction. CONCUR'93, volume 715 of LNCS

- A session is a sequence of interactions between two parties.
- It starts after a connection has been established.
- During the session, each party may execute its own local computation, interleaved with communications with the other party.
- Communications take the form of sending and receiving values over a channel.
- Throughout interaction between the two parties, there should be a perfect matching of sending actions in one with receiving actions in the other, and vice versa.

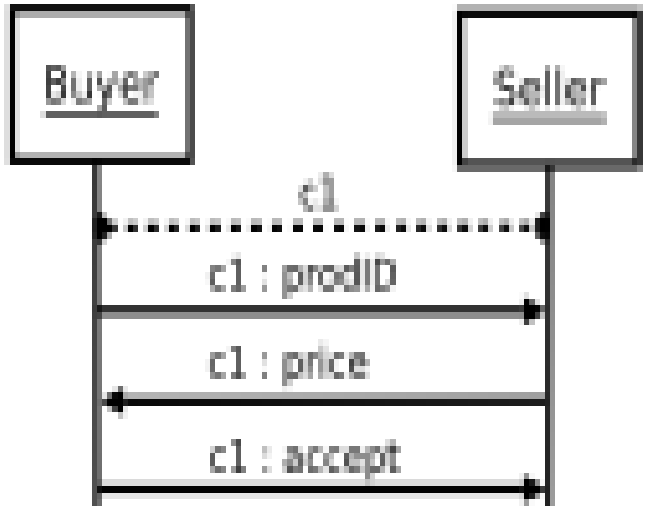
Session Types - 3

- sequence of interactions between two parties.
- … connection has been established.
- each party may execute its own local computation, interleaved with communications with the other party.
- Communications take the form of sending and receiving values over a channel.



Session Types - 3

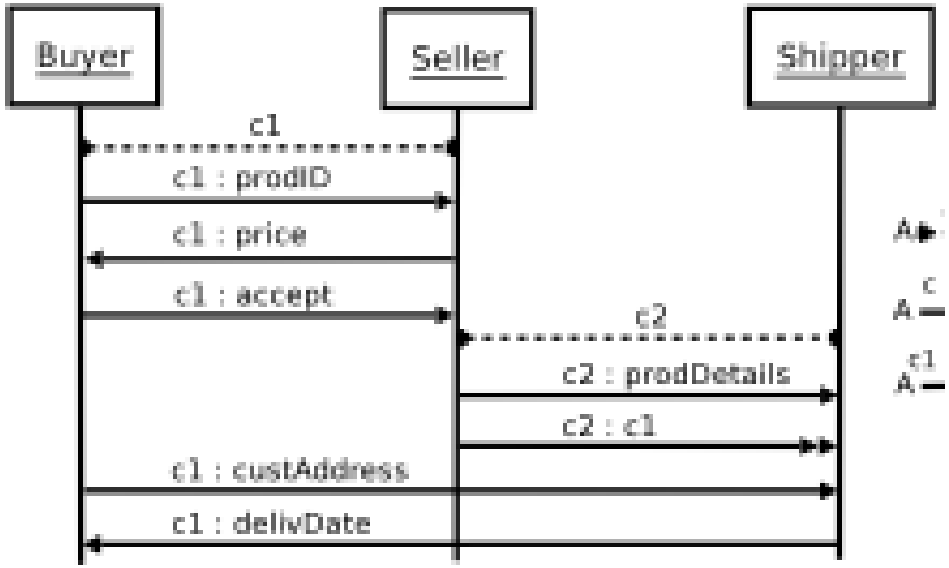
- sequence of interactions between two parties.
- … connection has been established.
- each party may execute its own local computation, interleaved with communications with the other party.
- Communications take the form of sending and receiving values over a channel.
- perfect match between sending/receiving



- Buyer's view:
  ! `ProdId` ?`Price` ! `Decision`
- Seller's view:
  ? `ProdId` !`Price` ? `Decision`

Session Types - 4

- sequence of interactions between two parties.
- … connection has been established
- each party may execute its own local computation, interleaved with communications with the other party.
- Communications take the form of sending and receiving values over a channel.
- perfect match between sending/receiving
- session composition and higher order sessions



Session Types for OO

- *Mariangiola Dezani-Ciancaglini, Dimitris Mostrous, Sophia Drossopoulou, Nobuko Yoshida,*
  Sessions Types for object oriented programmhttp://www.doc.ic.ac.uk/~yoshida/multiparty/multiparty.pdfing ,
  ECOOP 2005

- *Sophia Drossopoulou, Mariangiola Dezani-Ciancaglini, Mario Coppo,*
  Amalgamating the Session Types and the Object Oriented Programming Paradigms **Multiparadigm** Programming with OO Languages 2007 (an ECOOP workshop)

- .....

- .....

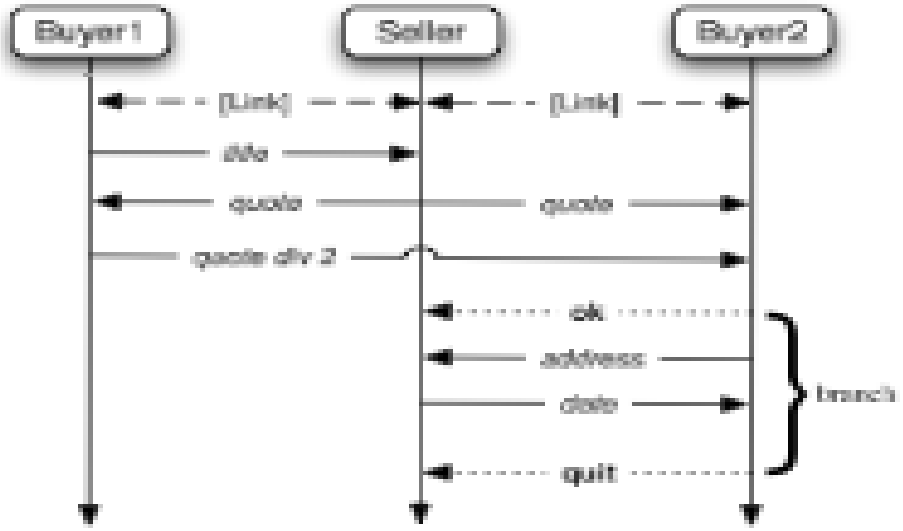
Multiparty Session Types - 1

Kohei Honda, Nobuko Yoshida and Marco Carbone
Multiplarty Asynchronous Session Types
POPL'2008

- generalize binary sessions to multiparty asynchronous sessions

Multiparty Session Types - 2

- generalize binary sessions to multiparty asynchronous sessions

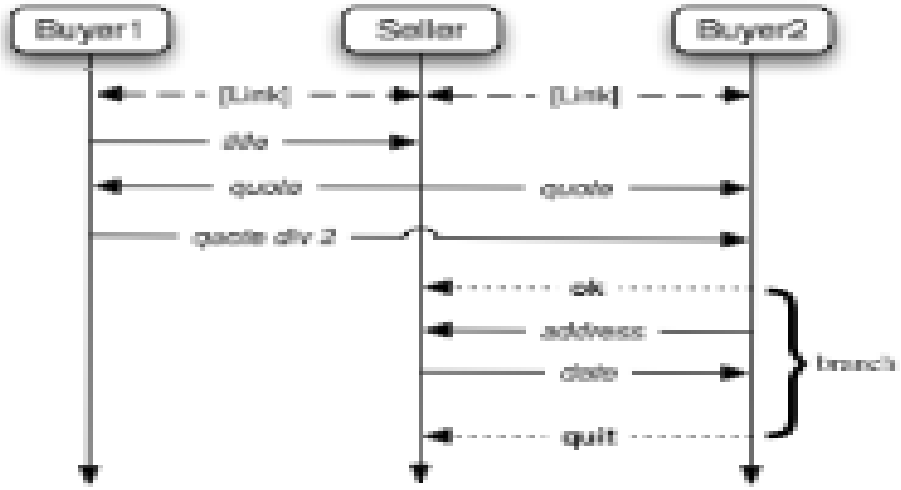


- three participants; they all talk to each other; two buyers collaborate to buy an item.

Multirole Session Types

Multirole session types – why?

- We revisit the two-buyers example from earlier



- Why just two buyes?
- Seller aware of how many/who individual buyers are …
- Could number of buyers vary statically/dynamically?

Multirole session types – why? -2

- The concept of several participants playing the same role is natural.

Multirole session types - when

- **Auction** (several bidders, one auctioneer, several secretaries)
  one conversation, auction of n items
  participants may join/leave dynamically
- **Virtual PC meeting** (PC chair, PC members, reviewers and subreviewers)
  n sub-conversations for n papers,
  participants may play different roles in different conversations
- **Court** (judge, jury, prosecutor, investigator)
  ….
- **Discussion Forum** (moderator, participant)
  …
- **Electronic commerce**
  ….

Multirole session types - what

- Conversations group related communications/actions
- Two or mote roles in one conversation
- One or more participants per role
- Conversation instances created
- Roles communicate with other roles on a channel
- Participants belong to one or more roles
- Participants send/receive messages on the channels corresponding to their role
- Conversations may be nested
- ….
- on top of an oo language

Conversations – e.g.

```
conversation Auction{

    roles  Auctioneer, Bidder, Secretary;

    channel ch1 for Auctioneer to Bidder:
              (  ! Item. (! Price.? Bid)* )*

    channel ch2 for Auctioneer to Secretary:
              ( ! Item. (!keep or !Price.!BidId ) )*

}
```

Design Issues and Further Work

Design Issues

- Of what kind are the participants of a role? Objects or processes?
- May a participant play more than one role?
- How does a participant join a role? Push/pull/authenticate?
- Is communication synchronous/asynchronous?
- Is there a global clock?
- Do all participants in a role send/receive in sync with each other? or according to a "leader"? or according to another role?
- Can a participant join/leave a role while a conversation is running?
- What if a role is empty?
- Which message is read when receiving messages from a role with more than one participant: the "first" message, an arbitrary message, all messages?

Design Issues - 2

- Of what kind are the participants of a role? Objects or **Processes?**
- May a **Process** play more than one role? **YES**
- How does a participant join a role? **Push/**Pull/authenticate?
- Is communication synchronous/**asynchronous**?
- Is there a global clock? **NO**
- Do all participants in a role send/receive in sync with each other? **NO** according to a "leader"? **NO**, or according to another role? **NO**
- Can participant join/leave role while conversation is running? **NOT YET**
- What if a role is empty? **NOT YET**
- Which message is read when receiving messages from a role with more than one participant: the "first" message, an arbitrary message, all messages? **IT DOES NOT MATTER THAT MUCH!**

Expressions are …

- this | e.m(e) | e.f | e.f := e' | **new** C — usual oo stuff
- **spawn** e | **suspend** e — usual concurrent stuff
- **new** cv — create a conversation
- e.r.**join** | e.r.**leave** — join/leave a role in conversation
- e.**start** — start a conversation
- e.ch.**send**(e') | e.ch.**receive** — send/receive one message
- e.ch.**receiveAll(**x**)**{ e' } — repeatedly receive messages and execute e'

Modelling challenge

Communication is

- Asynchronous
- Multicast (potentially in both ways)
- Different participants in the same role may be in different "stages" of communication

For this we introduced

- a noticeboard per conversation instance and channel, which contains the values sent so far per "communication stage"
- a status per process, conversation and channel consisting of
  - Read stage
  - Read index
  - Write stage

runtime organization

- Object == ClassId x (FldId → Value)
- Value == true | false | null | Addr
- Heap == Addr → (Object or Conversation)
- Conversation ==

  ConvId x

  (RoleId → ProcAddr*) x participants per role

  (ChId x stage → Value*) noticeboard per channel

- Heap == Addr → (Object or Conversation) and

  (ProcAddr x ConvAddr x ChanID

  →

  stage x int x stage)

  read stage, read index, write stage

Receiving

???

---

k.ch.receive, p, h ➔ ???

where h is a heap, p/k are process/convers. addr.

Receiving

h(p,k,ch) = (rdStage, rdIndx, wrStg)

---

k.ch.receive, p, h ➔ v, p, h'

where h is a heap, p/k are process/convers. addr.

Read the progress status of p, k, ch

Receiving

h(p,k,ch) = (rdStage, rdIndx, wrStg)

h'= h[p,k,ch |-> (rdStage+1, rdIndx, wrStage)]

---

k.ch.receive, p, h ➔ v, p, h'

where h is a heap, p/k are process/convers. addr.

Increment the read stage of p, k, ch

Receiving

h(p,k,ch) = (rdStage, rdIndx, wrStg)

h'= h[p,k,ch |-> (rdStage+1, rdIndx, wrStage)]

h(k,chId)=vals vals[rdStage] = … v …

k.ch.receive, p, h ➔ v, p, h'

where h is a heap, p/k are process/convers. addr.

Read arbitrary value from the noticeboard of k, ch of rdStage

Receiving

h(p,k,ch) = (rdStage, rdIndx, wrStg)

h'= h[p,k,ch |-> (rdStage+1, rdIndx, wrStage)]

h(k,chId)=vals vals[rdStage] = … v …

k.ch.receive, p, h ➔ v, p, h'

where h is a heap, p/k are process/convers. addr.

h(p,k,ch) = (rdStage, rdIndx, wrStg)

h'= h[p,k,ch |-> (rdStage, rdIndx+1, wrStage)]

h(k,chId)=vals vals[rdStage,rdIndx] = v

k.ch.receiveAll(x){e}, p, h ➔ e[v/x]; k.ch.receiveAll(x){e}, p, h'

Sending

???

---

k.ch.send(v), p, h ➔ ???

Sending

h(p,k,ch) = (rdStage, rdIndx, wrStg)

h'' = h[p,k,ch |-> (rdStage, rdIndx, wrStage+1)]

---

k.ch.send(v), p, h ➔ null, p, h'

Sending

h(p,k,ch) = (rdStage, rdIndx, wrStg)

h'' = h[p,k,ch |-> (rdStage, rdIndx, wrStage+1)]

h' = h''[k,ch,wrStage|-> h(k,ch,wrStage)::v]

k.ch.send(v), p, h ➔ null, p, h'

Design Issues - 3

- …
- Can participant join/leave role while conversation is running?
  AT FIXED POINTS OF THE CONVERSATION
- What if a role is empty?
  THROW EXCEPTION WHEN TIMEOUT
- Undo session part / failure of subconversation
- Do we want to indicate/restrict the possible number of parameters in a role (eg exactly one auctioneer)
- Garbage collect noticeboards no longer needed

Design Issues - 4

From **channel-centric** session types:

```
conversation Auction{
    …
    channel ch1 for Auctioneer to Bidder:
            (  ! Item. (! Price.? Bid)* )*
    channel ch2 for Auctioneer to Secretary:
            ( ! Item. (!keep or !Price.!BidId ) )*
}
```

to **participant-centric** session types

```
conversation Auction{
   …
 communications Auctioneer:
    ( ! Bidder.Item. ! Secretary.Item.
      (! Bidder.Price.? Bidder.Bid)*.
         Secretary.(!keep or Secretary.!Price.!BidId ) )*
}
```

Further Work

- Types
- Soundness proof
- Examples
- Advanced Features
- Case Studies
- Implementation
- Integration with GUI builder

Thank you!

Thank you!