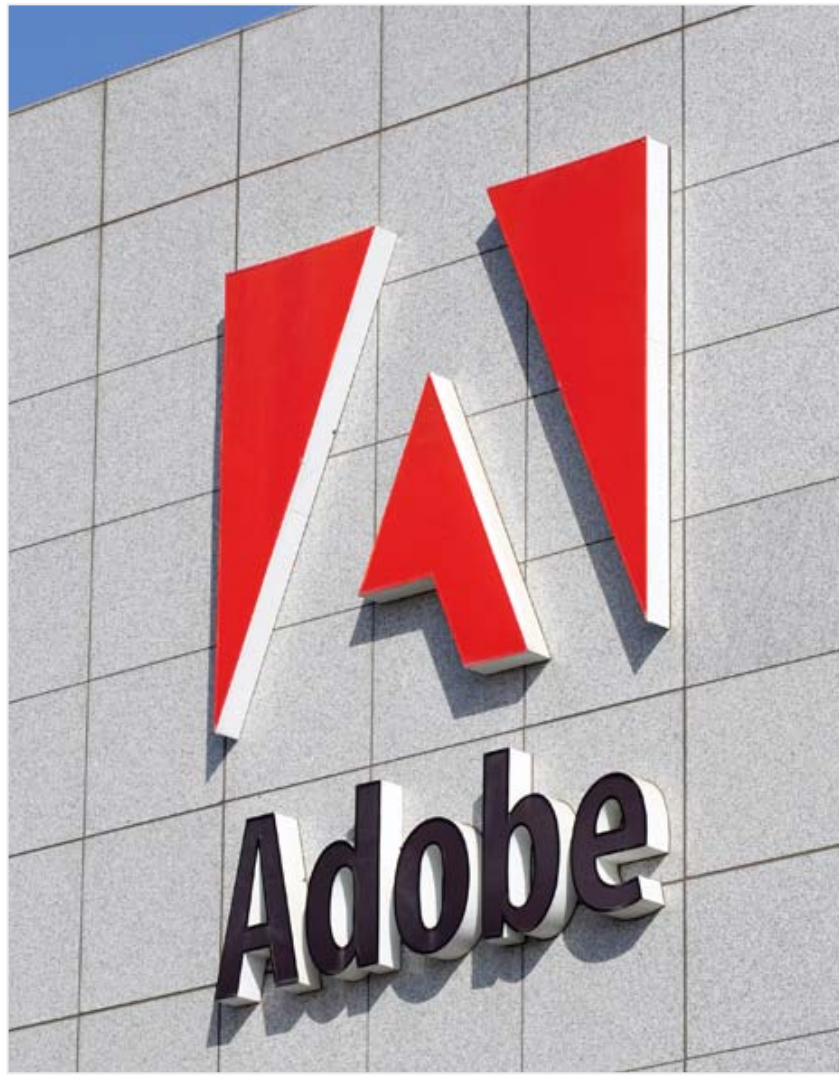
Adventures in Formal Methods at W3C: *The π of Choreography*

Charlton Barreto

W3C Technical Plenary

Cannes-Mandelieu, France

2006-March-01



W3C WORLD WIDE WEB consortium

Adobe

Motivation

- WS-CDL Critical Success Factors
  - CSF-007
    - To be successful, a CDL description MUST be verifiable at runtime
  - CSF-008
    - To be successful, a CDL description MUST enable static verification of correctness properties

Why Formalism?

- To provide mechanisms for ensuring desirable properties of real systems (e.g. type checking, bisimulation, model checking)
  - CSF-008 requires static type checking for behavioral types
- To give formal unambiguous semantics to WS-CDL so that we have a precise idea of behavior, offering guidelines for implementation
  - CSF-007 requires formal semantics to ensure correct monitoring

Correctness

- What formalism can we use for this?
  - We need:
    - Mobility,
    - Concurrency,
    - Location,
    - Identity (of a conversation)
- Process Algebra provides the basis all of this with a few additions

Formalisms

Model	Completeness	Compositionality	Parallelism	Resources
Turing Machines	☑	☒	☒	☑
Lambda	☑	☑	☒	☒
Petri Nets	☑	☒	☑	☑
CCS	☑	☑	☑	☒
π	☑	☑	☑	☑

π-calculus

- A program's state and it's "program pointer" are one and the same
- Completely does away with the representation of state
- It is easy to check for certain properties:
  - Deadlock
  - Compatibility
- Linear Typing to support safety/liveness properties in the presence of non-determinism & non-termination

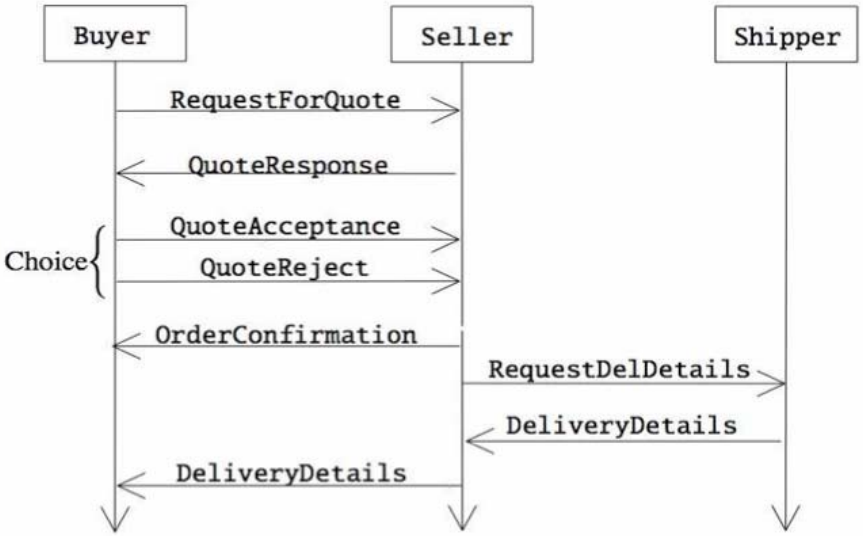
π-calculus

- Formally we have:

| ::= ∑iAi -> Bi: si<opi,ei,yi>.Ii	(choice)
| A -> B : ch(s~).I	(init)
| I | I	(par)
| if e@A then I else I	(if/then/else)
| (vs~)I	(new)
| rec X.I	(recursion)
| x@A := e.I	(assign)
| X	(recVar)

where e ::= v|x|f(e1, …., ek).

Example



if $x_{\texttt{quote}} \leq 100$ @Buyer **then**

{ Buyer → Seller : $B2Sch\langle \texttt{QuoteAccept} \rangle$.

Seller → Buyer : $B2Sch\langle \texttt{OrderConfirmation} \rangle$.

Seller → Shipper : InitS2H($S2Hch$).

Seller → Shipper : $S2Hch\langle \texttt{RequestDeliveryDetails} \rangle$.

Shipper → Seller : $S2Hch\langle \texttt{DeliveryDetails}, v_{\texttt{details}}, x_{\texttt{details}} \rangle$.

Seller → Buyer : $B2Sch\langle \texttt{DeliverDetails}, x_{\texttt{details}}, y_{\texttt{details}} \rangle$.**0** }

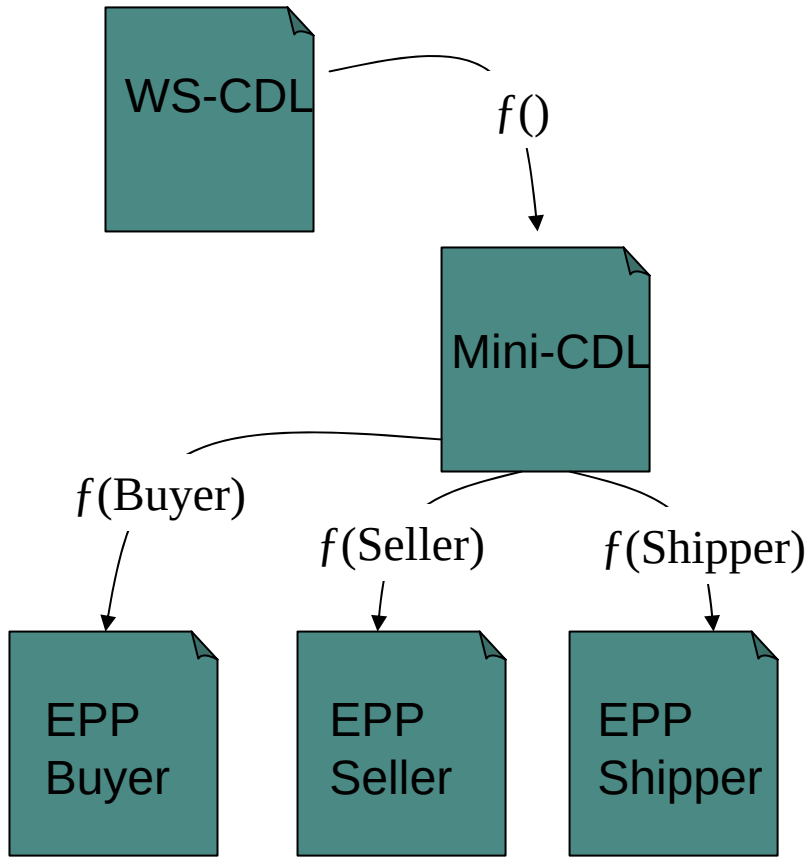
else

{ Buyer → Seller : $B2Sch\langle \texttt{QuoteReject} \rangle$.**0** }

- Notice the condition in the conditional branch, $x \leq 100$, is explicitly *located* at Buyer's.

π-calculus

- We also have an end-point calculus
  - Just as readable as before, but the point is…



References

- [1] Carbone, Honda, Yoshida, "Programming interaction with Types "
- [2] Kavantzas, "Aggregating Web Services: Choreography and WS-CDL "
- [3] Carbone, Honda, Yoshida, et. al. "A Theoretical Basis of Communication-Centred Concurrent Programming"
- [4] Milner, Parrow, Walker "A Calculus of Mobile Processes"

Resources

- W3C Web Services Choreography page: http://www.w3.org/2002/ws/chor
- This talk is linked from http://www.w3.org/Talks