NoBPM: Supporting Interaction-Oriented Automation via Normative Specifications of Processes

Munindar P. Singh

North Carolina State University
singh@ncsu.edu

September 2015
Business Process Domains

- Manufacturing
- Healthcare
- Analytics
- Foreign exchange
- Scientific collaboration
Human Elements: Global Hybrid Profile Mooring Launch

© Tom Kleindinst, WHOI
Range of Operating Environments: Glider Being Launched
A Sociotechnical View: Social and Technical Tiers

Requirements

Stakeholders

identify

specify

Norms
  Assumptions
  Mechanisms
  Metrics

realized in

Functional and Control Components

use, maintain

Data

Social Tier

Technical Tier
Representations

View

End point

Interaction

Multiparty

Uniparty

Open

Nature

Steps

Operations

Can be declarative

Constraints on steps

Business relationships

Meaning based

Can be procedural

Constraints on relationships

Can be declarative

Constraints on steps
BPM Today

Operational

- Regimented
- Control focused
- Documented
- Standardized
- Hides rationales

Meaning

- Ad hoc
- Afterthought
- Undocumented
- Regiments

NoBPM

Operational

- Adaptive
- Maintainable
- Scalable
- Supports flexibility
- Regulated
- Communication focused

Social

- Primary
- Norm based
- Generate from meanings

© Singh (NCSU) September 2015
Challenges in and Vision for Business Processes

- Elicit
- Model
- Validate
- Reason
- Enact
- Mine
# Challenges in and Vision for Business Processes

<table>
<thead>
<tr>
<th>BPM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elicit</td>
</tr>
<tr>
<td>Model</td>
</tr>
<tr>
<td>Validate</td>
</tr>
<tr>
<td>Reason</td>
</tr>
<tr>
<td>Enact</td>
</tr>
<tr>
<td>Mine</td>
</tr>
</tbody>
</table>

© Singh (NCSU)  
NoBPM  
September 2015
### Challenges in and Vision for Business Processes

<table>
<thead>
<tr>
<th>Steps</th>
<th>BPM</th>
<th>NoBPM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elicit</td>
<td>Techniques, not requirements</td>
<td>Business relationships</td>
</tr>
<tr>
<td>Model</td>
<td>Steps</td>
<td>Business relationships</td>
</tr>
<tr>
<td>Validate</td>
<td>Extremely difficult (wrt requirements)</td>
<td>Facilitated by abstraction</td>
</tr>
<tr>
<td>Reason</td>
<td>Impossible wrt requirements</td>
<td>Facilitation by abstraction</td>
</tr>
<tr>
<td>Enact</td>
<td>Surprisingly difficult if decentralized</td>
<td>Naturally decentralized</td>
</tr>
<tr>
<td>Mine</td>
<td>Activity patterns</td>
<td>Business relationship patterns</td>
</tr>
</tbody>
</table>

© Singh (NCSU)  
NoBPM  
September 2015 8
Thesis and Overview

Looking back to look ahead . . .

▶ Formal models of normative relationships
  ▶ AI&Law 1999; AAAI 2008; AAMAS 2011; TIST 2013
▶ Application to business processes
  ▶ TSE 2005; TOSEM 2009; Computer 2009; ICSOC 2012; RE 2014; ICSOC 2015ab
▶ Representing and reasoning about commitments (and goals)
  ▶ AAMAS 2008; AAMAS 2009; AAAI 2015; AAMAS 2015
▶ Declarative operational representation and enactment
  ▶ AAMAS 2011; ICWS 2011; AAMAS 2012
▶ Mining human interactions (contracts, chats, emails)
▶ Human-subject studies of modelers
  ▶ TSC 2012; AAMAS 2012; IC 2014; JAAMAS 2014
▶ Human-subject studies of performers
  ▶ Being analyzed; being launched
Governance

Cupid: Information-Based Commitments

Alignment

BSPL, the Blindingly Simple Protocol Language

Conclusions
Governance Challenges: Autonomy and Dynamism

Capturing norms is essential to addressing these challenges

- **Support configurational adaptation**, for example
  - Resource sharing: Offer ocean instrument for sharing
  - Affiliation: Add new laboratories
  - Sanction: Allow external sharing of results to fulfill deliverables

- **Support operational adaptation**, for example
  - Resource sharing: Preempt low-priority users in case of oil spill
  - Affiliation: Forbid unilateral publishing of results
  - Sanction: Absolve researcher who reveals results to prevent public endangerment (extenuating circumstances)

- **Research challenges**
  - Abstractions to capture rules of encounter
  - Methods to design and analyze such abstractions
  - Methods to implement such abstractions
Governance

Current Practice
- Managerial
  - Control focused
  - Regimented
- Manual
  - Hides rationales
  - Error-prone
  - Not maintainable
  - Not scalable

Norm-Based
- Collaborative
- Regulated
  - Supports flexibility
- Communication focused
- Computational
  - Scalable
  - Adaptive
  - Maintainable

Obstructs user needs
Not scalable
Not maintainable
Error-prone
Hides rationales
Control focused
Regimented
Manual

Social Architecture for Governance of Secure Collaboration

Principals and Orgs

Role Façade

- imposes

Liability

- requires

Privilege

- grants

Qualification

- requires Role

Principal

- realizes

Internal Policy

- applies

Individual

- includes

Org

- plays

Role

— Singh (NCSU) —
Normative Relationships for Governance

Each norm type is a directed relationship: crucial for accountability
## Norms as Façades

<table>
<thead>
<tr>
<th>Norm</th>
<th>Subject’s Façade</th>
<th>Object’s Façade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commitment</td>
<td>Liability</td>
<td>Privilege</td>
</tr>
<tr>
<td>Authorization</td>
<td>Privilege</td>
<td>Liability</td>
</tr>
<tr>
<td>Power</td>
<td>Privilege</td>
<td>Liability</td>
</tr>
<tr>
<td>Prohibition</td>
<td>Liability</td>
<td>Privilege</td>
</tr>
<tr>
<td>Sanction</td>
<td>Liability</td>
<td>Privilege</td>
</tr>
</tbody>
</table>
Life Cycle for Norms: 1

Using a variant of the UML state diagram notation

Norm

Terminated

- Null
- Satisfied
- Violated

create

terminate

suspend

reactivate

Active

- Conditional
- In Force

antecedent

Pending

© Singh (NCSU)  NoBPM  September 2015
Life Cycle for Norms: 2

Substate of a terminated norm

<table>
<thead>
<tr>
<th>If terminated in</th>
<th>ant</th>
<th>con</th>
<th>Then</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Com</td>
<td>Aut</td>
<td>Pro</td>
</tr>
<tr>
<td>false false</td>
<td>null</td>
<td>null</td>
<td>null</td>
</tr>
<tr>
<td>false true</td>
<td>sat</td>
<td>vio</td>
<td>null</td>
</tr>
<tr>
<td>true false</td>
<td>vio</td>
<td>null</td>
<td>sat</td>
</tr>
<tr>
<td>true true</td>
<td>sat</td>
<td>sat</td>
<td>vio</td>
</tr>
</tbody>
</table>
Governance and Policies: Two Kinds of Interaction

Conversations with autonomous parties; control over resources

Internal Policy

Interaction: Conversation

Principal Qua Self

Principal Qua Other

Communicative Act

content

actor of

Principal

applies

determines

considers

© Singh (NCSU) NoBPM September 2015
Vocabulary for Governance and Policies

Attributes of principals, their relationships, resource states and capabilities

![Diagram showing the relationships between different terms related to governance and policies.](image-url)

- Action: Organization, Resource, Communication
- Property: Participation, Resource, Norm State
- Stative: Participation, Resource, Norm State
- Actions: Eject, Admit, Contribute, Withdraw, Delegate, Assign, Inform, Request, Query
- Norm States: Member, Registrand, Owns, Created, Violated, Satisfied

© Singh (NCSU)
Governance

**Cupid: Information-Based Commitments**

Alignment

BSPL, the Blindingly Simple Protocol Language

Conclusions
Computing Commitment Progression
Via explicit operations or because of logical properties

\[ C(\text{Buyer}, \text{Seller}, \text{goods}, \text{pay}): \text{Active and conditional} \]

- If \( \text{goods} \land C(\text{Buyer}, \text{Seller}, \text{goods}, \text{pay}) \) Then
  - Active and detached (or unconditional or base)
  - \( C(\text{Buyer}, \text{Seller}, T, \text{pay}) \)
- If \( C(\text{Buyer}, \text{Seller}, T, \text{pay}) \) Then
  - If \( \text{pay} \) Then Satisfied
  - If never \( \text{pay} \) Then Violated
- If \( C(\text{Buyer}, \text{Seller}, \text{goods}, \text{pay}) \) Then
  - If \( \text{pay} \) Then Satisfied
  - If never \( \text{pay} \) and never \( \text{goods} \) Then Expired

Can be nested:

\[ C(\text{Seller}, \text{Buyer}, \text{pay}, C(\text{Shipper}, \text{Buyer}, T, \text{deliverGoods})) \]
Operationalizing Commitments: Detach then Discharge

$C(\text{debtor, creditor, antecedent, consequent})$

$d: \text{Debtor}$

$c: \text{Creditor}$

create($d, c, p, q$)

$p$

$q$
Operationalizing Commitments: Discharge First; Optional Detach

How about this?

c: Creditor

create(d, c, p, q)

q

opt

p

[true]
Operationalizing Commitments: Detach First; Optional Discharge

How about this?

d: Debtor

create(d, c, p, q)

p

opt

[true]

q

c: Creditor
Operationalizing Commitments: Creation by Creditor

\[ C(\text{debtor}, \text{creditor}, \text{antecedent}, \text{consequent}) \]

\[
\text{create}(d, c, p, q)
\]

\[ d: \text{Debtor} \quad \text{c: Creditor} \]
Operationalizing Commitments: Strengthening by Creditor

\[ C(\text{debtor}, \text{creditor}, \text{antecedent}, \text{consequent}) \]

\[ C(d, c, p, q) \]

no active commitment

\[ C(d, c, \top, q) \]

\[ C(d, c, p, q) \]

\[ \text{create}(d, c, p, q) \]

\[ d: \text{Debtor} \]

\[ c: \text{Creditor} \]
Cupid: Unifying Accountability and Traceability
Computing states of norms over event stores

- Benefits: Basis for
  - Intelligent decision making
  - Key Performance Indicators

- Begin from event schema
  - Keys
    - Distinguished timestamp attribute

- Specify accountability requirements as norms

- Automatically generate SQL schema from event schema

- Automatically generate SQL queries to determine accountability status at specified moment
  - Now
  - A hypothetical moment to help ascribe responsibility
An Information Model and Commitment Specification

Quote(mID, cID, qID, itemID, uPrice, t) with key qID
Order(cID, mID, oID, qID, qty, addr, t) with key oID
Payment(cID, mID, pID, oID, pPrice, t) with key pID
Shipment(mID, cID, sID, oID, addr, t) with key sID
Refund(mID, cID, rID, pID, rAmount, t) with key rID
Coupon(cID, mID, uID, oID, rebate, t) with key uID

commitment DiscountQuote mID to cID
create Quote
detach Order and Payment[, Quote + 10]
    where pPrice >= 0.9 * uPrice * qty
discharge Shipment[, Payment + 5]

A DiscountQuote commitment from a merchant to a customer is

▶ created upon Quote;
▶ detached if Order happens and Payment happens within ten days of Quote and is for at least 90% of quoted amount (else expires)
▶ discharged if Shipment happens within five days of Payment (else violated)
Example: Compensation
Illustrates nesting: A commitment depends upon another commitment's state

commitment Compensation mID to cID
create Quote
detach violated(DiscountQuote)
discharge Refund[\text{violated}(DiscountQuote) + 9] \text{ where } r\text{Amount} = p\text{Price}

A Compensation commitment is created upon Quote and says that if DiscountQuote is violated, the merchant will refund the payment within nine days of the violation.
Properties

- All Cupid queries are safe
  - Given any possible model $M$ with finite extensions for base events, the extension of $Q$ relative to $M$, $\llbracket Q \rrbracket$, is finite
- *Well-identified* specifications capture a notion of adequate correlation among the events that in the specification.
- Instances of a *finitely expirable* specification are guaranteed to expire if not detached within a finite amount of time.
- Instances of a *finitely violable* specification are guaranteed to expire if not discharged within a finite amount of time.
Governance

Cupid: Information-Based Commitments

Alignment

BSPL, the Blindingly Simple Protocol Language

Conclusions
Commitment Progression via Messaging

Two views of the same enactment

- $C_A$ is $C(Alice, Bob, paid($12), del(mas))$
- $C_{UA}$ is $C(Alice, Bob, \top, del(mas))$
Asynchrony May Cause Misalignment
Does Bob infer a commitment that Alice doesn't?

- $c_A$ is $C(\text{paid}($12), del(\text{mas}))$
- $c_{UA}$ is $C(\top, \text{del}(\text{mas}))$

Diagram:

- Alice: $c_A$, Create($c_A$), Release($c_A$), $\neg c_A$
  - Aligned

- Bob: $c_A$, Create($c_A$), $\neg c_A$

- Alice: $c_A$, Create($c_A$), Release($c_A$), $\neg c_A$
  - Misaligned

- Bob: $c_A$, Create($c_A$), $\neg c_A$

- Alice: $c_A$, Create($c_A$), Detach($c_A$), $\neg c_{UA}$, Cancel($c_A$), $\neg c_{UA}$
  - Misaligned

- Bob: $c_A$, Create($c_A$), $c_{UA}$
Commitment-Level Interoperability
Alignment with respect to a creditor’s expectations

- Operational interoperability: Liveness of composed system
- Commitment alignment
  - Vector of observations \( O = [O_1 \ldots O_n] \); one per agent
  - \( O \) is causally valid: if \( w \) receives a message from \( z \) then \( z \) must have sent that message to \( z \)
  - \( O \) is aligned wrt \( C(x, y, r, u) \) iff

\[
O_y \vdash C(x, y, r, u) \Rightarrow O_x \vdash C(x, y, r, u)
\]

- Noto Bene: this is asymmetric
- Interoperability: Either \( O \) is aligned wrt \( c \) or no matter what happens, \( O \) will extend to an \( O' \) that is aligned wrt \( c \)
Alignment When the Relevant Messages have Landed

- $c_A$ is $C($paid($12$), del(mas))
- $c_{UA}$ is $C($⊤, del(mas))
Challenge: Guarantee Alignment
By appropriately constraining an agent’s local computations

▶ Formalize updates by messages indicating creation, cancelation, release, detach, and discharge
▶ Formalize information propagation constraints
   ▶ Notify creditors of discharges
   ▶ Notify debtors of detaches
▶ Taking into account commitment reasoning

\[
\text{Detach. } C(r \land s, u) \land r \rightarrow C(s, u) \\
\text{L-Disjoin. } C(r, u) \land C(s, u) \rightarrow C(r \lor s, u) \\
\text{R-Conjoin. } C(r, u) \land C(r, v) \rightarrow C(r, u \land v)
\]

▶ Prove: Computations guarantee complete \( O \) for any incomplete \( O \)
Challenges of nonFIFO Message Delivery

A weak commitment cannot be canceled when a strong commitment holds

- \( C_A \) is \( C(\text{paid}($12), \text{del}(\text{mas})) \)
- \( C_B \) is \( C(\text{paid}($12), \text{del}(\text{soc})) \)
- \( C_{AB} \) is \( C(\text{paid}($12), \text{del}(\text{mas}) \land \text{del}(\text{soc})) \)
- \( C_A \land C_B \vdash C_{AB} \)
Transaction-Related Challenges

Aligned but unrealistic outcomes due to the lack of transactions

- $C_A$ is $C(paid($12$), del(mas))$
- $C_C$ is $C(paid($15$), del(soc))$
- $C_{UC}$ is $C(\top, del(soc))$
- $C_{UAC}$ is $C(\top, del(mas) \land del(soc))$
- $C_{UA} \land C_{UC} \vdash C_{UAC}$

- Cancel strongest possible (unconditional) version of any commitment
- Atomicity: Cancel an entire transaction, not parts
Governance

Cupid: Information-Based Commitments

Alignment

BSPL, the Blindingly Simple Protocol Language
  Correctness and Model Checking
  Local State Transfer
  Composition

Conclusions
Business Protocols

No!

- No internal reasoning
  - No private predicates in guards
- No method calls
  - No self calls
- No synchronous messages
  - No business puts itself on indefinite hold waiting for its partner to proceed
- No causally invalid expectations
  - No \textit{nonlocal} choice
    - No nonlocal choice that matters
  - No control of incoming message occurrence or ordering
  - No dependence on occurrence or ordering of remote message emission or reception
- No reliance on ordering across channels
  - No reliance on ordering within a channel unless warranted
Traditional Specifications: Procedural
Low-level, over-specified protocols, easily wrong

- Traditional approaches
  - Emphasize arbitrary ordering and occurrence constraints
  - Then work hard to deal with those constraints

- Our philosophy: The Zen of Distributed Computing
  - Necessary ordering constraints fall out from *causality*
  - Necessary occurrence constraints fall out from *integrity*
  - Unnecessary constraints: simply *ignore* such
BSPL, the Blindingly Simple Protocol Language

Main ideas

- Only *two* syntactic notions
  - Declare a message schema: as an atomic protocol
  - Declare a composite protocol: as a bag of references to protocols

- Parameters are central
  - Provide a basis for expressing meaning in terms of bindings in protocol instances
  - Yield unambiguous specification of compositions through public parameters
  - Capture progression of a role’s knowledge
  - Capture the completeness of a protocol enactment
  - Capture uniqueness of enactments through keys

- Separate structure (parameters) from meaning (bindings)
  - Capture many important constraints purely structurally
Key Parameters in BSPL
Marked as 「key」

- All the key parameters *together* form the key
- Each protocol must define at least one key parameter
- Each message or protocol reference must have at least one key parameter in common with the protocol in whose declaration it occurs
- The key of a protocol provides a basis for the uniqueness of its enactments
Parameter Adornments in BSPL

Capture the essential causal structure of a protocol (for simplicity, assume all parameters are strings)

- ⌜in⌝: Information that must be provided to instantiate a protocol
  - Bindings must exist locally in order to proceed
  - Bindings must be produced through some other protocol

- ⌜out⌝: Information that is generated by the protocol instances
  - Bindings can be fed into other protocols through their ⌜in⌝ parameters, thereby accomplishing composition
  - A standalone protocol must adorn all its public parameters ⌜out⌟

- ⌜nil⌟: Information that is absent from the protocol instance
  - Bindings must not exist
The *Hello* Protocol

```
Hello {
  role Self, Other
  parameter out greeting key

  Self → Other: hi[out greeting key]
}
```

- At most one instance of *Hello* for each greeting
- At most one *hi* message for each greeting
- Enactable standalone: no parameter is `⌜in⌝`
- The key of *hi* is explicit; often left implicit on messages
The Pay Protocol

Pay \{ 
  \text{role Payer, Payee} \\
  \text{parameter in ID key, in amount} \\
\}

\text{Payer} \leftrightarrow \text{Payee}: \text{payM[ in ID, in amount ]}

- At most one \text{payM} for each ID
- Not enactable standalone: \textbf{why?}
- The key of \text{payM} is implicit (for brevity)
The *Purchase* Protocol

\[
\text{Purchase} \left\{ \begin{array}{c}
\text{role } B, \ S, \ \text{Shipper} \\
\text{parameter out ID key, out item, out price, out outcome} \\
\text{private address, resp} \\
B \rightarrow S: \text{rfq} [\text{out ID, out item}] \\
S \rightarrow B: \text{quote} [\text{in ID, in item, out price}] \\
B \rightarrow S: \text{accept} [\text{in ID, in item, in price, out address, out resp}] \\
B \rightarrow S: \text{reject} [\text{in ID, in item, in price, out outcome, out resp}] \\
S \rightarrow \text{Shipper}: \text{ship} [\text{in ID, in item, in address}] \\
\text{Shipper} \rightarrow B: \text{deliver} [\text{in ID, in item, in address, out outcome}] \\
\end{array} \right. 
\]

- At most one item, price, and outcome binding per ID
- Enactable standalone
- *reject* conflicts with *accept* on response (*a private parameter*)
- *reject or deliver* must occur for completion (*to bind outcome*)
Knowledge and Viability
When is a message viable? What effect does it have on a role’s local knowledge?

- Knowledge increases monotonically at each role
- An \( \text{out} \) parameter \textbf{creates} and transmits knowledge
- An \( \text{in} \) parameter transmits knowledge
- Repetitions through multiple paths are harmless and superfluous
Possible Enactments as Sets of Local Histories

Each participant’s local history: sequence of messages sent and received

- rfq
- quote
- accept
- ship
- deliver

Buyer

Seller

Shipper

- rfq
- quote
- reject

Buyer

Seller

Shipper
Safety and Liveness Violations

Encode a protocol's causal structure in temporal logic and evaluate properties.

### Purchase Unsafe

- **Buyer**
- **Seller**
- **Shipper**

- rfq ➔
- quote ➔
- accept ➔ ship ➔
- reject ➔
- deliver ➔

### Purchase Minus Ship

- **R**
- **C**
- **S**

- request ➔
- accept ➔
- deliver

**Safety Violation**

**Liveness Violation**

*Cannot occur*
Realizing BSPL via LoST (Local State Transfer)

Does not assume FIFO or reliable messaging

- **Internal Reasoning**
- **Business Meaning**
- **Local State**
- **Messages**

**LoST**

**Application-specific**

**Communication**

- Unique messages
- Integrity checks on incoming messages
- Consistency of local choices on outgoing messages
Implementing LoST

Think of the message logs you want

- For each role
  - For each message that it sends or receives
    - Maintain a local relation of the same schema as the message
  - Receive and store any message provided
    - It is not a duplicate
    - Its integrity checks with respect to parameter bindings
    - Garbage collect expired sessions: requires additional annotations
- Send any unique message provided
  - Parameter bindings agree with previous bindings for the same keys for \(\text{in}\) parameters
  - No bindings for \(\text{out}\) and \(\text{nil}\) parameters exist
Comparing LoST and ReST

<table>
<thead>
<tr>
<th></th>
<th>ReST</th>
<th>LoST</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Modality</strong></td>
<td>Two-party; client-server; synchronous</td>
<td>Multiparty interactions; peer-to-peer; asynchronous</td>
</tr>
<tr>
<td><strong>Computation</strong></td>
<td>Server computes definitive resource state</td>
<td>Each party computes its definitive local state and the parties collaboratively and (potentially implicitly) compute the definitive interaction state</td>
</tr>
<tr>
<td><strong>State</strong></td>
<td>Server maintains no client state</td>
<td>Each party maintains its local state and, implicitly, the relevant components of the states of other parties from which there is a chain of messages to this party</td>
</tr>
</tbody>
</table>
### Comparing LoST and ReST

<table>
<thead>
<tr>
<th></th>
<th>ReST</th>
<th>LoST</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Transfer</strong></td>
<td>State of a resource, suitably represented</td>
<td>Local state of an interaction via parameter bindings, suitably represented</td>
</tr>
<tr>
<td><strong>Idempotent</strong></td>
<td>For some verbs, especially GET</td>
<td>Always; repetitions are guaranteed harmless</td>
</tr>
<tr>
<td><strong>Caching</strong></td>
<td>Programmer can specify if cacheable</td>
<td>Always cacheable</td>
</tr>
<tr>
<td><strong>Uniform interface</strong></td>
<td>GET, POST, ...</td>
<td>(\text{in}, \text{out}, \text{nil})</td>
</tr>
<tr>
<td><strong>Naming</strong></td>
<td>Of resources via URIs</td>
<td>Of interactions via (composite) keys, whose bindings could be URIs</td>
</tr>
</tbody>
</table>
Comparing LoST and WS-CDL

- **Similarity:** both emphasize interaction
- **Differences:** WS-CDL is
  - Procedural
    - Explicit constructs for ordering
    - Sequential message ordering by default
  - Agent-oriented
    - Includes agents’ internal reasoning within choreography (specify what service an agent executes upon receiving a message)
    - Relies on agents’ internal decision-making to achieve composition (take a value from Chor A and send it in Chor B)
  - No semantic notion of completeness
Composing Protocols

Without imposing private constraints on a party playing a role

Bilateral Foreign Exchange

T

request →

M

response ←

Trilateral Foreign Exchange

T

request →

X

response ←

M

response ←

▶ Is Trilateral = Bilateral ⊗ Bilateral?
The *Trilateral* Protocol

Also called price discovery

### Trilateral

- **role** Taker, Exchange, Maker
- **parameter** out ID, key, out query, out res

#### General Bilateral

- `GeneralBilateral(Taker, Exchange, out ID, out query, in res)`
- `GeneralBilateral(Exchange, Maker, in ID, in query, out res)`

```
T          X          M

request[out ID, out query] ➔

response[in ID, in res] ➔

request[in ID, in query] ➔

response[in ID, out res] ➔
```
Governance

Cupid: Information-Based Commitments

Alignment

BSPL, the Blindingly Simple Protocol Language

Conclusions
Recommendations for Research

Many PhD dissertations remain to be written . . .

- A fresh start on business processes
  - Corner concerns of BPM become central concerns in NoBPM
  - Sociotechnical focus: Supporting regulation, not just regimentation
- Operational models and architectures that support regulation
  - Monitoring
  - Sanctioning
- Agent decision making to support
  - Proactive behavior
  - Incorporating goals as duals of norms
- Declarative languages
  - Need for theoretical results
  - Need for implementations and evaluations
- Don’t forget the humans!
Challenges and Partial Recent Progress

- Storing and retrieving events to determine the state of a norm
  - Mapping commitments to relational algebra [AAAI 2015]
- Maintaining alignment of views despite decentralization
  - Communications to guarantee (eventual) alignment [AAMAS 2015]
  - TBD: maximizing partial or “quick” alignment
- Designing protocols and Org contexts for monitorability
  - Failure of compositionality of monitorability [IJCAI 2015]
  - Automatically close a context to ensure monitorability
- Designing protocols and Org for robustness and resilience
  - Typology of sanctions and sanctioning processes [Draft]
  - TBD: Formalization of normative robustness and resilience
  - TBD: Reasoning about sanctions for design of Orgs
- Design processes conducive to autonomy
  - Abstract formal model of a sociotechnical design process [RE 2014]
  - TBD: Methodologies
Thanks

- Amit Chopra, Lancaster
- Matteo Baldoni and Cristina Baroglio, Torino
- Sponsors over the years
  - Consortium for Ocean Leadership
  - DARPA
  - IBM
  - Intel
  - National Science Foundation
- Read and publish in or ask me about
  - (Current EIC) ACM Transactions on Internet Technology
  - (Former EIC) IEEE Internet Computing

http://www.csc.ncsu.edu/faculty/mpsingh/