Reasoning About the Behavior of Semantic Web Services with Concurrent Transaction Logic

33rd International Conference on Very Large Data Bases (VLDB)

September 23-27 2007, Vienna, Austria

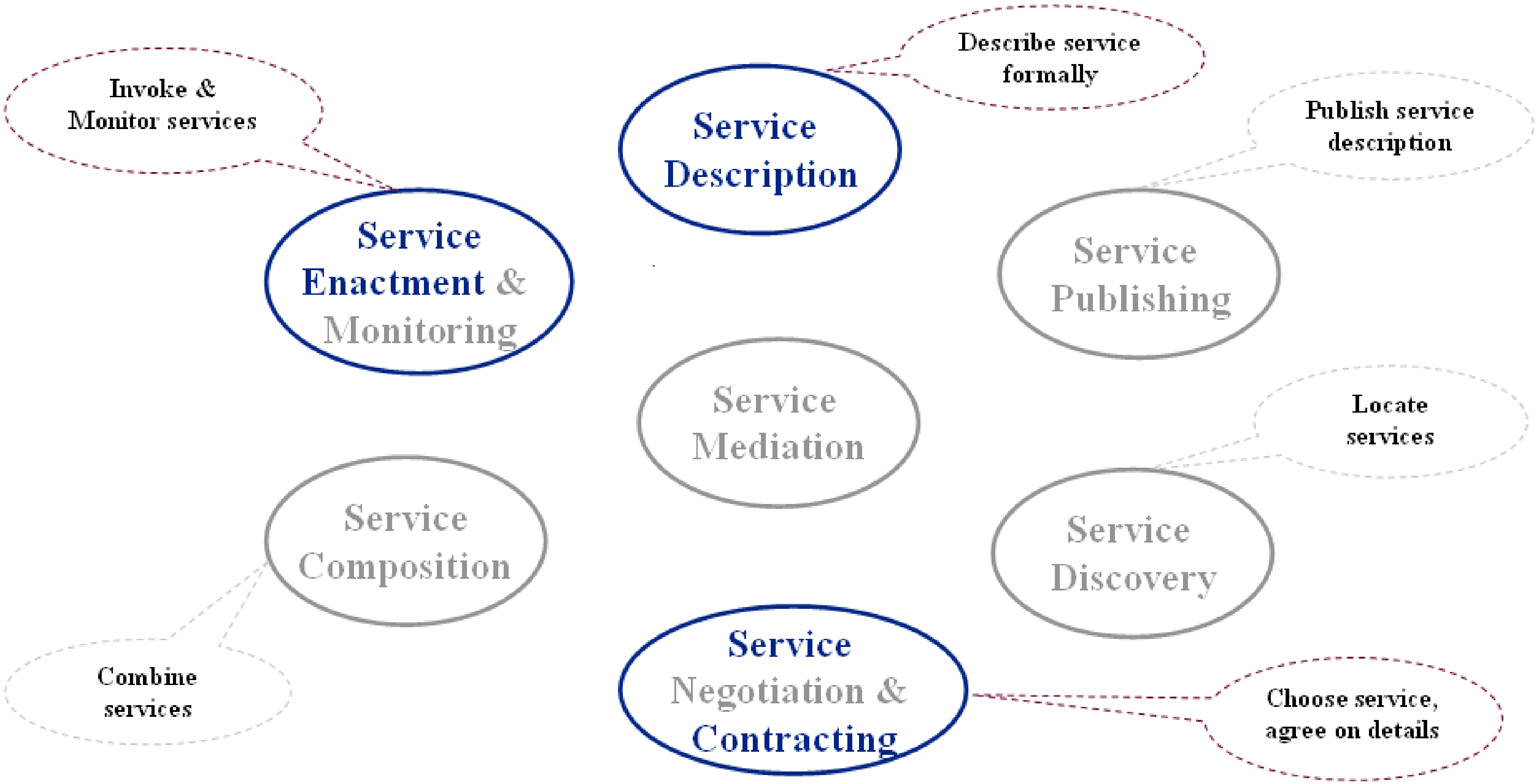
Dumitru Roman[1] and Michael Kifer[2]

[1] *University of Innsbruck / DERI Innsbruck, Austria*

[2] *State University of New York at Stony Brook, New York, U.S.A.*

dumitru.roman@deri.at, kifer@cs.sunysb.edu

Semantic Web Services



SWS Approaches: OWL-S, SWSF, WSMO, SAWSDL, etc.

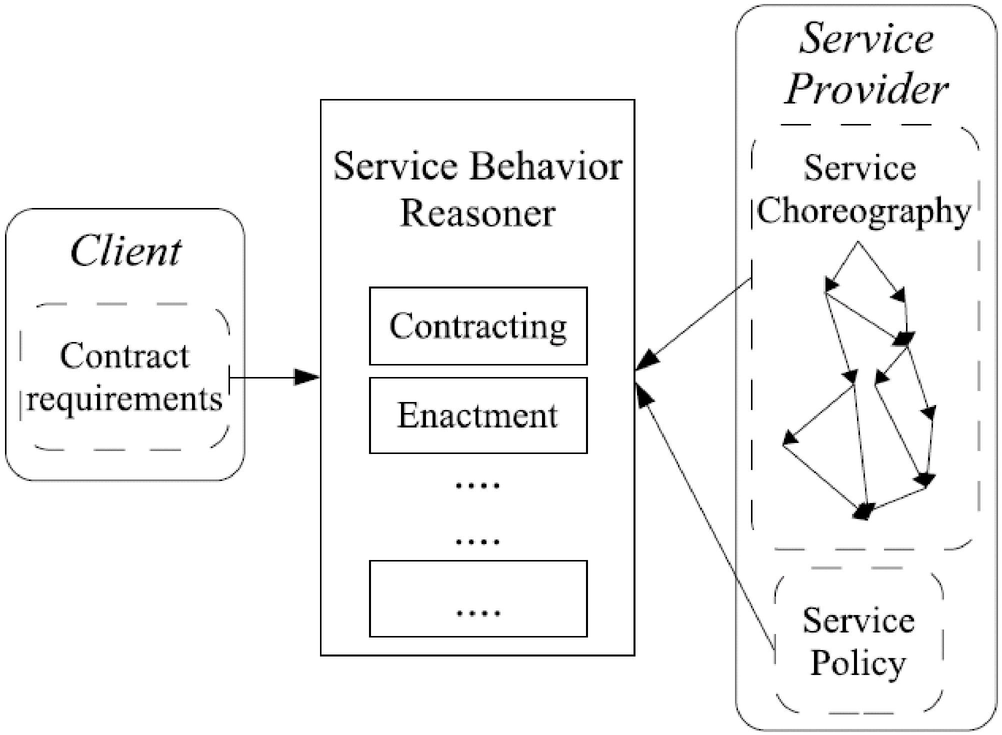
Outline

- Motivation
  - Service behavior: modeling, reasoning, and enactment
- Introduction to Concurrent Transaction Logic (CTR)
  (we use it to do stuff)
- Service modeling with CTR
  - Control Flow
  - Events and Constraints
  - Data Flow and Conditional Control Flow
- Reasoning about choreography and contracts
  - Phase 1: Transformation
  - Phase 2: Extended Proof Theory
- Related Work
- Conclusions

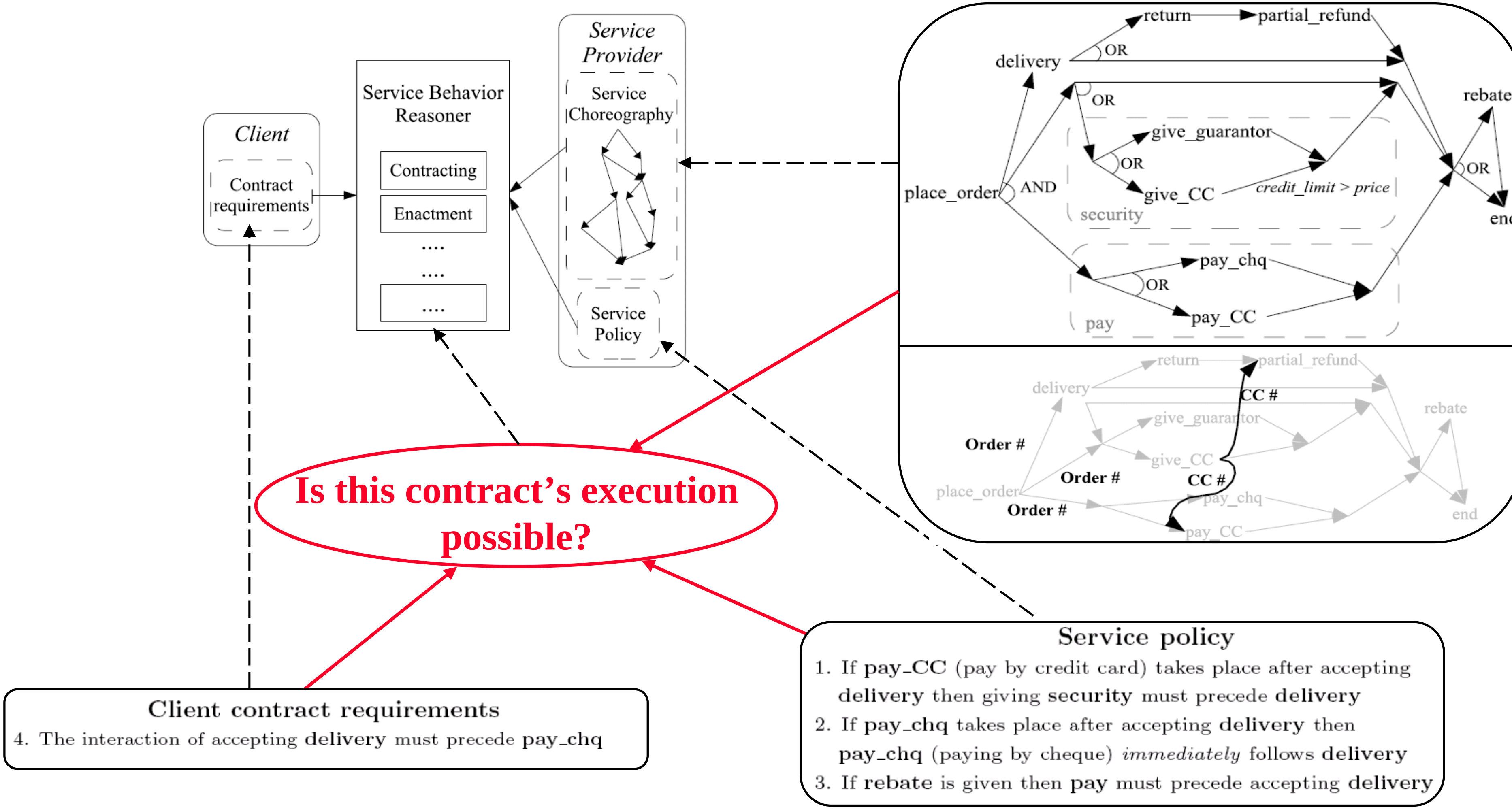
Outline

- **Motivation**
  - **Service behavior: modeling, reasoning, and enactment**
- Introduction to Concurrent Transaction Logic (CTR)
- Service modeling with CTR
- Reasoning about choreography and contracts
- Related Work
- Conclusions

Modeling & Reasoning About Service Behavior



Example: (Conditional) Control and Data Flow Graphs & Constraints



Outline

- Motivation
  - Service behavior: modeling, reasoning, and enactment
- **Introduction to Concurrent Transaction Logic (CTR)**
- Service modeling with CTR
- Reasoning about choreography and contracts
- Related Work
- Conclusions

Introduction to CTR

- An extension of the classical predicate logic to program and reason about state changes
  - Reduces to classical logic when no state transitions
  - *Atomic formulas* of CTR are identical to those of the classical logic:
    - $p(t_1, t_2, \ldots, t_n)$ – where p is a predicate symbol, the t_i's are function terms
    - More complex formulas are built using connectives and quantifiers
- Informal semantics
  - A set of database *states*
    - E.g. $s_1, s_2, \ldots, s_n$
  - A collection of *paths* (sequences of states)
    - E.g. $< s_1>, < s_1, s_2>, < s_1, s_2, \ldots, s_n >$
  - Truth value of CTR formulas is determined over paths, *not* at states
    - E.g. if a formula a is true over a path $< s_1, s_2, \ldots, s_n >$, it means that a can "*execute*" starting at state s_1, change to state $s_2, s_3 \ldots$, etc. Will terminate at state s_n

CTR Syntax

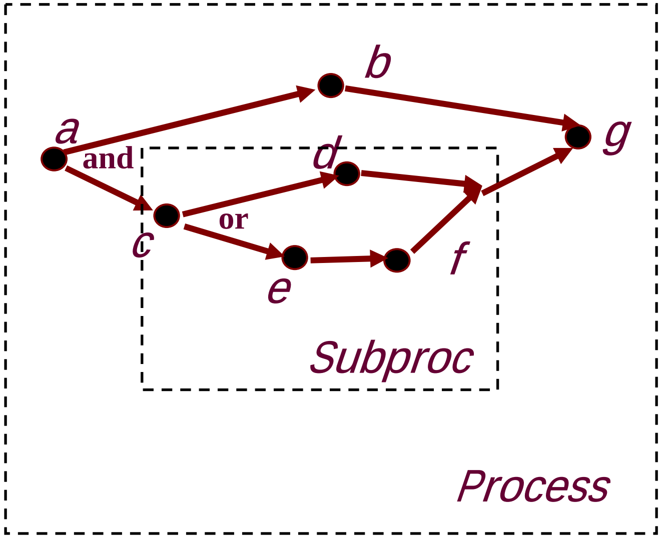
- Countable sets of symbols
  - predicate symbols
  - function symbols
  - variables
- Logical connectives
  - $a \otimes b$ – execute a then execute b
  - $a \mid b$ – a and b must both execute concurrently in an interleaved fashion.
  - $a \wedge b$ – a and b must both execute along the *same* path
  - $a \vee b$ – execute a or execute b non-deterministically
  - $\neg a$ – execute in any way, provided that this will *not* be a valid execution of a
  - $\odot a$ – execute a in isolation execution i.e., without interleaving with other concurrently running activities
- Example: $a \otimes (b \mid (c \otimes (d \vee (e \otimes f)))) \otimes g$

Concurrent-Horn Subset of CTR

- Concurrent-Horn *goals*:
  - Any atomic formula is a concurrent-Horn goal
  - $a \otimes b$, $a \mid b$, and $a \vee b$ are concurrent-Horn goals, if so are *a* and *b*
  - $\odot a$ is a concurrent-Horn goals, if so is *a*
- Concurrent-Horn *rules*
  - CTR formulas of the form *head* <- *body* (i.e. *head* $\vee \neg$ *body*), where *head* is an atomic formula and *body* is a concurrent-Horn goal
    - *head* can be viewed as a subroutine name: one way to execute *head* is to execute its definition, *body*
- Example:

  Process $\leftarrow$ $a \otimes (b \mid Subproc) \otimes g$

  Subproc $\leftarrow$ $(c \otimes (d \vee (e \otimes f)))$



- An SLD-like proof procedure proves concurrent Horn formulas and *executes* them at the same time

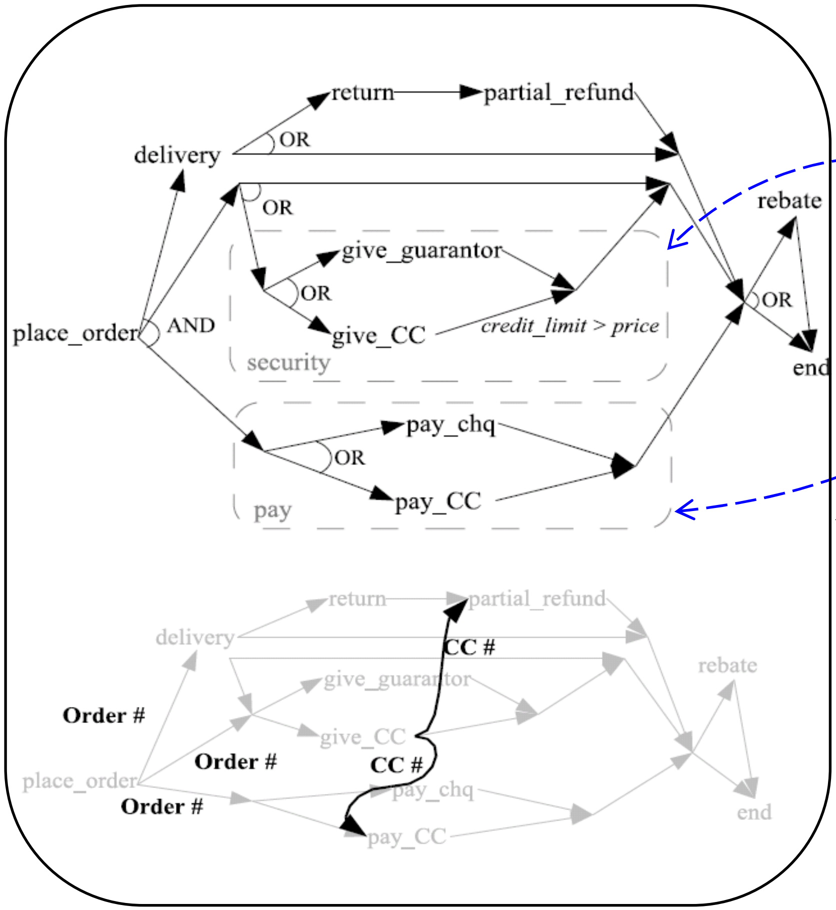
CTR – Elementary State Transitions

- Propositions that represent “built-in” state transitions
  - Usually we use the following elementary state transitions: insert.p and delete.p
    - insert.p: add fact *p* to the current state
    - delete.p: delete fact p from the current state
  - We also use elementary transitions to represent events that happen during workflows: *place_order*, *delivery*, etc.

Outline

- Motivation
  - Service behavior: modeling, reasoning, and enactment
- Introduction to Concurrent Transaction Logic (CTR)
- **Service modeling with CTR**
  - **Control Flow**
  - Events and Constraints
  - Data Flow and Conditional Control Flow
- Reasoning about choreography and contracts
- Related Work
- Conclusions

Modeling Service Choreography with CTR (Control Flow Graphs & Data Flow)



$\text{path} \equiv \Psi \vee \neg\Psi$

$place\_order(Order\#, Price) \leftarrow$
$\quad ((delivery(Order\#) \otimes (refund(Order\#) \vee \mathbf{path}))$
$\quad\ \mid (security(Order\#, Price) \vee \mathbf{path})$
$\quad\ \mid\ pay(Order\#, Price)$
$\quad) \otimes (rebate(Order\#) \vee \mathbf{path}) \otimes end$

$security(Order\#, Price) \leftarrow$
$\quad give\_guarantor(Order\#) \vee$
$\quad (give\_CC(Order\#, CC\#) \otimes$
$\quad\ credit\_limit(CC\#, Limit) \otimes Limit > Price)$

$pay(Order\#, Price) \leftarrow$
$\quad pay\_chq(Order\#, Price) \vee pay\_CC(Order\#, Price)$

$refund(Order\#) \leftarrow$
$\quad return(Order\#) \otimes partial\_refund(Order\#)$

$partial\_refund(Order\#) \leftarrow$
$\quad (payment(Order\#, cc, CC\#) \otimes$
$\quad\ refund\_amount(Order\#, Amount) \otimes$
$\quad\ issue\_credit\_CC(CC\#, Amount))$
$\quad \vee$
$\quad (payment(Order\#, cheque, Cheque\#) \otimes$
$\quad\ refund\_amount(Order\#, Amount) \otimes$
$\quad\ send\_check(Order\#, Amount))$

$give\_CC(Order\#, CC\#) \leftarrow$
$\quad insert.payment(Order\#, cc, CC\#)$

$pay\_chq(Order\#, Price) \leftarrow$
$\quad get\_cheque(Price, Cheque\#) \otimes$
$\quad insert.payment(Order\#, cheque, Cheque\#)$

$pay\_CC(Order\#, Price) \leftarrow$
$\quad payment(Order\#, cc, CC\#) \otimes charge(CC\#, Price)$

Outline

- Motivation
  - Service behavior: modeling, reasoning, and enactment
- Introduction to Concurrent Transaction Logic (CTR)
- Service modeling with CTR
  - Control Flow
  - **Events and Constraints**
  - Data Flow and Conditional Control Flow
- Reasoning about choreography and contracts
- Related Work
- Conclusions

Constraint Algebra

$\nabla a \equiv path \otimes a \otimes path$

1. Primitive constraints
   - Event *e* must happen ∇e
   - Event *e* must not happen $\neg\nabla e$
2. Immediate serial constraints
   - Events $e_1, e_2, \ldots, e_n$ must happen next to each other with no other events in-between $\nabla\odot(e_1 \otimes e_2 \otimes e_3 \otimes \ldots \otimes \nabla e_n)$
3. Serial constraints
   - Events $e_1, e_2, \ldots, e_n$ must execute (or not execute) in that order with possible interleaving $\nabla e_1 \otimes \neg\nabla e_2 \otimes \nabla e_3 \otimes \neg\nabla e_4 \otimes \ldots \otimes \nabla e_n$
4. Complex constraints
   - If C_1, C_2 are constraints then so are $C_1 \wedge C_2$, and $C_1 \vee C_2$

Constraints Expressivity Examples

- Events *e* and *f* must both occur (in any order)
  - $\nabla e \wedge \nabla f$
- It is not possible for *e* and *f* to happen together
  - $\neg\nabla e \vee \neg\nabla f$
- If event *e* occurs, then *f* must also occur (before or after *e*)
  - $\neg\nabla e \vee \nabla f$; $\nabla e \rightarrow \nabla f$
- If event *e* occurs, then *f* must occur later
  - $\neg\nabla e \vee (\nabla e \otimes \nabla f)$; $\nabla e \rightarrow (\nabla e \otimes \nabla f)$
- If event *f* has occurred, then event *e* must have occurred some time prior to that
  - $\neg\nabla f \vee (\nabla e \otimes \nabla f)$
- If both *e* and *f* occur, then *e* must come before *f*
  - $\neg\nabla e \vee \neg\nabla f \vee (\nabla e \otimes \nabla f)$; $(\nabla e \wedge \nabla f) \rightarrow (\nabla e \otimes \nabla f)$
- If event *e* occurs, then *f* must occur right after *e* with no event in-between
  - $\neg\nabla e \vee \nabla \odot (e \otimes f)$
- If *k* and *d* both occur then *d* must happen right after *k* with no other event in-between
  - $\neg\nabla k \vee \neg\nabla d \vee \nabla \odot (k \otimes d)$ (or $(\nabla k \wedge \nabla d) \rightarrow \nabla \odot (k \otimes d)$)

Service Constraints: Example

Service policy

1. If **pay_CC** (pay by credit card) takes place after accepting **delivery** then giving **security** must precede **delivery**
2. If **pay_chq** takes place after accepting **delivery** then **pay_chq** (paying by cheque) *immediately* follows **delivery**
3. If **rebate** is given then **pay** must precede accepting **delivery**

Client contract requirements

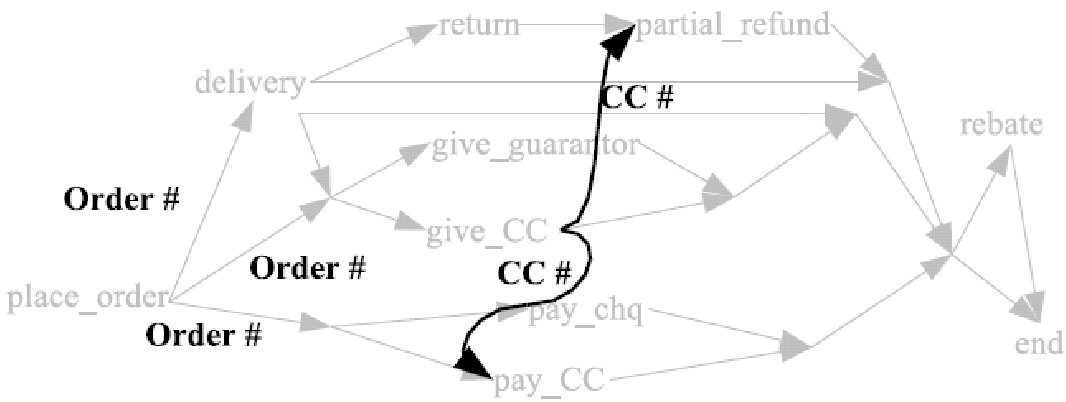
4. The interaction of accepting **delivery** must precede **pay_chq**

1. $\exists\ Order\#\ \exists\ Price$
   $((\triangledown delivery(Order\#) \otimes \triangledown pay\_CC(Order\#, Price)) \rightarrow$
   $(\triangledown security(Order\#, Price) \otimes \triangledown delivery(Order\#)))$
2. $\exists\ Order\#\ \exists\ Price$
   $((\triangledown delivery(Order\#) \otimes \triangledown pay\_chq(Order\#, Price)) \rightarrow$
   $\triangledown \odot (delivery(Order\#) \otimes pay\_chq(Order\#, Price)))$
3. $\exists\ Order\#\ \exists\ Price$
   $(\triangledown rebate(Order\#) \rightarrow$
   $(\triangledown pay(Order\#, Price) \otimes \triangledown delivery(Order\#)))$
4. $\exists\ Order\#\ \exists\ Price$
   $(\triangledown delivery(Order\#) \otimes \triangledown pay\_chq(Order\#, Price))$

Outline

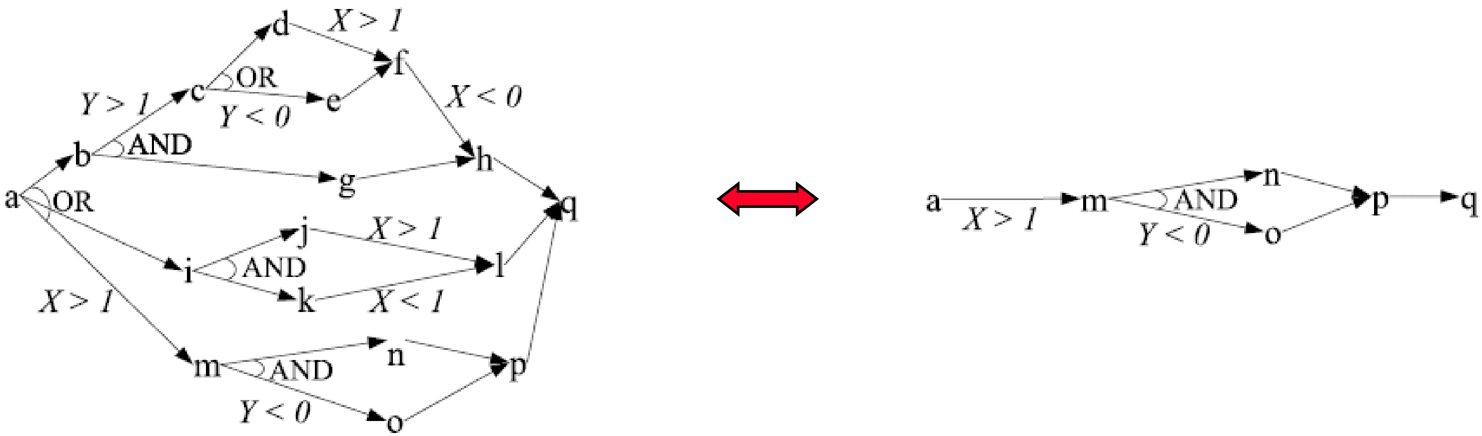
- Motivation
  - Service behavior: modeling, reasoning, and enactment
- Introduction to Concurrent Transaction Logic (CTR)
- Service modeling with CTR
  - Control Flow
  - Events and Constraints
  - **Data Flow and Conditional Control Flow**
- Reasoning about choreography and contracts
  - Phase 1: Transformation
  - Phase 2: Extended Proof Theory
- Related Work
- Conclusions

Constraints Implied by Data Flow



1. $\exists\, Order\#\ \exists\, CC\#\ \exists\, Price$
   $((\nabla give\_CC(Order\#, CC\#) \wedge \nabla pay\_CC(Order\#, Price)) \;\rightarrow$
   $(\nabla give\_CC(Order\#, CC\#) \otimes \nabla pay\_CC(Order\#, Price)))$
2. $\exists\, Order\#\ \exists\, Price$
   $((\nabla give\_CC(Order\#, Price) \wedge \nabla partial\_refund(Order\#)) \rightarrow$
   $(\nabla give\_CC(Order\#, Price) \otimes \nabla partial\_refund(Order\#)))$

Reduction of Conditional Control Flows



Can propagate constraints and reduce control flows by eliminating (or flagging) impossible parts.

Outline

- Motivation
  - Service behavior: modeling, reasoning, and enactment
- Introduction to Concurrent Transaction Logic (CTR)
- Service modeling with CTR
  - Control Flow
  - Events and Constraints
  - Data Flow and Conditional Control Flow
- **Reasoning about choreography and contracts**
- Related Work
- Conclusions

Reasoning About Service Behavior

- *Contracting*: determine if contracting for the service is possible
  - Find out if there is an execution of the CTR formula $G \wedge C$ given the set of service choreography definitions R, i.e.
    - Check that there is a path $s_1, s_2, \ldots, s_k$ such that ($\models$ is CTR entailment)

$$R, s_1, \ldots, s_k \models G \wedge C$$

- *Enactment*
  - Find a constructive *proof* that

$$R, s_1, \ldots, s_k \models G \wedge C \text{ for some path } s_1, \ldots, s_k$$

    - Each such proof is a way to execute the choreography so that all the constraints are satisfied

Solution – Overview

- Phase 1
  - Aim: get rid of primitive constraints and distribute disjunctions
  - Translate the formula $G \wedge C$ into an equivalent formula

    $$\bigvee_i (G_i \ \bigwedge_j serialConstr_{i,j})$$

    where each $serialConstr_{i,j}$ is either an immediate serial constraint or a (plain) serial constraint, and G_i is a concurrent-Horn goal
    - Each step in this transformation can be viewed as an inference rule in a proof theory
- Phase 2
  - Extend the proof theory of Horn CTR to formulas of the form

    $$G \ \bigwedge_j serialConstr_j$$

    which result from the Phase 1. Then use proof theory on these formulas
    - If a proof is found, then enactment of the service is possible

Outline

- Motivation
  - Service behavior: modeling, reasoning, and enactment
- Introduction to Concurrent Transaction Logic (CTR)
- Service modeling with CTR
  - Control Flow
  - Events and Constraints
  - Data Flow and Conditional Control Flow
- Reasoning about choreography and contracts
  - **Phase 1: Transformation**
  - Phase 2: Extended Proof Theory
- Related Work
- Conclusions

Phase 1 – Normal Form Transformation

- Applying Complex Constraints

$$\begin{array}{lcl} T \wedge (C_1 \vee C_2) & \vdash & (T \wedge C_1) \vee (T \wedge C_2) \\ T \wedge (C_1 \wedge C_2) & \vdash & (T \wedge C_1) \vee (T \wedge C_2) \end{array}$$

- Applying Primitive Constraints

$$\begin{array}{lcll} (\alpha \wedge \nabla\alpha) & \vdash & \alpha & \\ (\beta \wedge \nabla\alpha) & \vdash & \neg\mathtt{path} & \mathit{if}\ \alpha \neq \beta \\ (\alpha \wedge \neg\nabla\alpha) & \vdash & \neg\mathtt{path} & \\ (\beta \wedge \neg\nabla\alpha) & \vdash & \beta & \mathit{if}\ \alpha \neq \beta \end{array}$$

$$\begin{array}{lcl} (T \otimes K) \wedge \nabla\alpha & \vdash & \begin{cases} (T \wedge \alpha) \otimes K & \mathit{if}\ \alpha\ \mathit{occurs\ in}\ T \\ T \otimes (K \wedge \alpha) & \mathit{if}\ \alpha\ \mathit{occurs\ in}\ K \end{cases} \\ T \otimes K \wedge \neg\nabla\alpha & \vdash & (T \wedge \neg\nabla\alpha) \otimes (K \wedge \neg\nabla\alpha) \\ (T \mid K) \wedge \alpha & \vdash & \begin{cases} (T \wedge \alpha) \mid K & \mathit{if}\ \alpha\ \mathit{occurs\ in}\ T \\ T \mid (K \wedge \alpha) & \mathit{if}\ \alpha\ \mathit{occurs\ in}\ K \end{cases} \end{array}$$

$$\begin{array}{lcl} (T \mid K) \wedge \neg\nabla\alpha & \vdash & (T \wedge \neg\nabla\alpha) \mid (K \wedge \neg\nabla\alpha) \\ \odot T \wedge \sigma & \vdash & \odot(T \wedge \sigma,\ T) \\ (T \vee K) \wedge \sigma & \vdash & (T \wedge \sigma) \vee\ (K \wedge \sigma) \end{array}$$

- The result of the transformation can be one of:
  - $\neg$*path*, i.e. inconsistency
    - Enactment is not possible
  - A formula of the form $\bigvee_i (G_i \ \bigwedge_j serialConstr_{i,j})$
    - Scheduling might be possible; apply Phase 2 for each $G_i \ \bigwedge_j serialConstr_{i,j}$ separately

Outline

- Motivation
  - Service behavior: modeling, reasoning, and enactment
- Introduction to Concurrent Transaction Logic (CTR)
- Service modeling with CTR
  - Control Flow
  - Events and Constraints
  - Data Flow and Conditional Control Flow
- Reasoning about choreography and contracts
  - Phase 1: Transformation
  - **Phase 2: Extended Proof Theory**
- Related Work
- Conclusions

Phase 2 – Extended Proof Theory

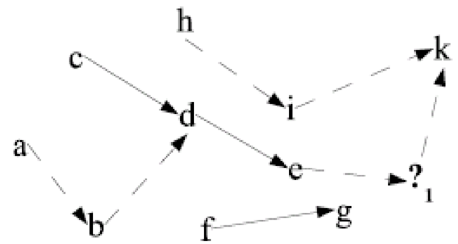
- A proof theory for formulas of the form

$$G \bigwedge_j serialConstr_j$$

- Two steps
  1. Check constraints for internal consistency and eliminate redundancy
     - If the constraints are consistent, then go to next step, which is based on inference rules
  2. Inference rules

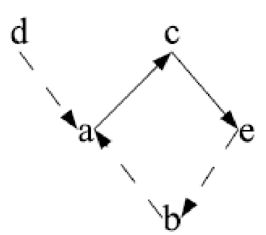
Phase 2, Step 1 – Constraint Graphs

- Constraint graph

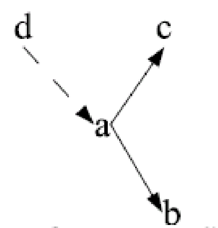


$$\{\nabla\odot(c\otimes d\otimes e),\ \nabla\odot(f\otimes g),\ \nabla a\otimes\nabla b\otimes\nabla d,\ \nabla h\otimes\nabla i\otimes\nabla k,\ \nabla e\otimes\nabla ?_1\otimes\nabla k\}$$

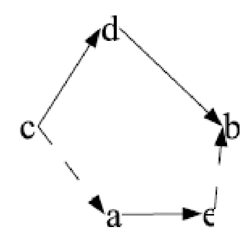
- Inconsistency patterns (capture all inconsistencies)



$$\{\nabla\odot(a\otimes c),\nabla\odot(c\otimes e),\ \nabla e\otimes\nabla b\otimes\nabla a,\ \nabla d\otimes\nabla a\}$$

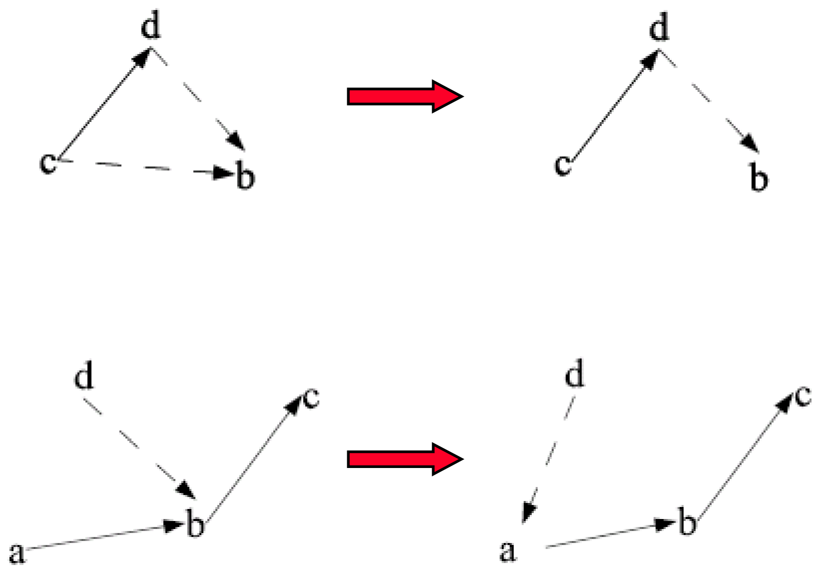


$$\{\nabla\odot(a\otimes b),\nabla\odot(a\otimes c),\ \nabla d\otimes\nabla a\}$$



$$\{\nabla\odot(c\otimes d\otimes b),\ \nabla\odot(a\otimes e),\ \nabla c\otimes\nabla a,\ \nabla e\otimes\nabla b\}$$

Phase 2, Step 1 – Redundancy Elimination & Well Formed Constraint Graphs



Phase 2, Step 2 – Inference Rules

- Applying transaction definitions
  - if a <- b is in ***P*** then

$$\frac{P,\, D \;\text{---}\vdash\; (\exists)\; (\psi' \wedge C')\; \sigma}{P,\, D \;\text{---}\vdash\; (\exists)\; \psi \wedge C}$$

ψ' is ψ with some occurrence of a replaced with b;

C' is C after deleting a and splicing edges adjacent on a

- Querying the database
  - if a is a database predicate in ψ and $D \models a$ then

$$\frac{P,\, D \;\text{---}\vdash\; (\exists)\; (\psi' \wedge C)\; \sigma}{P,\, D \;\text{---}\vdash\; (\exists)\; \psi \wedge C}$$

ψ' is ψ with some occurrence of a deleted

Phase 2, Step 2 – Inference Rules (Cont'd)

- Executing elementary updates
  - If a is an elementary update s.t. D_1 –a–> D_2 then

$$\frac{P, D_2 \text{ ---}\vdash (\exists)\ (\psi' \wedge C')\ \sigma}{P, D_1 \text{ ---}\vdash (\exists)\ \psi \wedge C}$$

ψ' is ψ with some occurrence of a deleted

C' is C after deleting some nodes (details omitted)

- Executing atomic transactions
  - If $\odot\alpha$ occurs in ψ then

$$\frac{P, D \text{ ---}\vdash (\exists)\ (\alpha \otimes \psi') \wedge C}{P, D \text{ ---}\vdash (\exists)\ \psi \wedge C}$$

ψ' is ψ with some occurrence of $\odot\alpha$ deleted

Outline

- Motivation
  - Service behavior: modeling, reasoning, and enactment
- Introduction to Concurrent Transaction Logic (CTR)
- Service modeling with CTR
  - Control Flow
  - Events and Constraints
  - Data Flow and Conditional Control Flow
- Reasoning about choreography and contracts
  - Phase 1: Transformation
  - Phase 2: Extended Proof Theory
- **Related Work**
- Conclusions

Related Work

- Service contracting
  - Existing work focuses on defining frameworks, models, and architectures different aspects and phases of e-contracting (negotiation, enforcement, violation detection, monitoring, legal aspects)
  - We provide a simple yet realistic and useful framework for e-contracting
    - Solve a *concrete* problem in establishing of contracts and enacting Web services
- Workflow/process modeling
  - Many languages for process modeling, e.g. YAWL, DecSerFlow
  - Ours is as expressive as DecSerFlow, and additionally integrates with conditional control flows, data flows, provides reasoning mechanisms
- Process verification
  - Most of the existing approaches use model checking for verification
    - Complexity exponential in the size of the control graph
  - CTR's integrates several process modeling paradigms: conditional control flows, data flows, hierarchical modeling, constraints
    - *Complexity polynomial in the size of the control graph* and exponential in the size of the constraints (due to better structuring of the problem)

Outline

- Motivation
  - Service behavior: modeling, reasoning, and enactment
- Introduction to Concurrent Transaction Logic (CTR)
- Service modeling with CTR
  - Control Flow
  - Events and Constraints
  - Data Flow and Conditional Control Flow
- Reasoning about choreography and contracts
  - Phase 1: Transformation
  - Phase 2: Extended Proof Theory
- Related Work
- **Conclusions**

Conclusions

- Formulated the problems of choreography, contracting, and enactment for semantic Web services using Concurrent Transaction Logic
  - complex set of constraints
  - data flow and conditional process controls
  - extended CTR proof theory
- Presented reasoning techniques for
  - deciding if automatic contracting for a service is possible
  - finding a choreography that obeys the policy of the service and the user requirements of the contract
  - enacting the service
- Can be extended to multi-party contracts
- Possible extensions
  - more expressive interaction patterns, e.g. loops
  - subsets of constraints for which the verification problem has a better complexity

Thank you!