Formal Commercial Contracts

Work in Progress

Christian Stefansen
Department of Computer Science, University of Copenhagen

(joint work with Jesper Andersen, Ebbe Elsborg, Fritz Henglein, and Jakob Grue Simonsen)

Presentation at Microsoft Research, Redmond, June 14, 2004

Agenda

- Motivation
- Definitions and Focus
- The REA Model
- Our Contract Model
- Syntax and Semantics
- Contract Analysis

The NEXT Project

Next Generation Software Technology for Enterprise Systems (since 2001)

Partners:

- Microsoft Business Solutions, formerly Navision A/S
- IT University of Copenhagen: Founded 1999, 40 full-time researchers, active broadly in IT (mathematical-technical, design and business) research; see http://www.it.edu
- Department of Computer Science, University of Copenhagen (DIKU): Founded 1970, 30 full-time researchers, active in CS (algorithmics, distributed systems, information systems/HCI, computer vision, programming languages etc.) research; see http://www.diku.dk

Project homepage: http://www.it.edu/next/

What is a Contract?

An agreement to do or not do certain things.

Used by (almost) any company in daily operation. Any exchange is governed by a written or oral agreement or the underlying context and legislation.

Background

- Capturing contractual obligations is important for planning and reporting.
- Monitoring/execution, analysis, and integration not or badly supported in present ERP (Enterprise Resource Planning) systems.

Background

Problems associated with informal contract handling in practice:

- Disagreement on what events have happened
- Disagreement on semantics of contract
- Potential disagreement on residual contract (due to nondeterminacy of consequences)
- Malexecution of contracts
- Entering bad contracts due to impossible or inferior analysis
- Reporting the state of company affairs (current commitments, capacity requirements, risk analysis, valuation, due diligence) is not feasible due to contracts in natural language stored in paper.

Goals

A domain-specific contract specification language addressing the current problems above by giving:

- Flexible user-specified contracts
- Automatic contract execution (less work-intensive, larger degree of non-repudiation)
- Deadline management (integration with workflow systems and ERP systems)
- (Real-time) reporting on active contracts
- Contract analysis (valuation, capacity requirements, risk analysis)
- Contract design support (identifying race conditions, non-determinism)

Focus

The primary focus is on *“happy path” actualization of valid going-concern contracts between two or more disjoint parties*

Happy path A “good faith” intended execution path of a contract. A course of action that complies with the purpose, intent, and “meeting of minds” of the contract.

Example Contract # 1

Abbreviated agreement of sale:

Agreement to Sell Goods
Section 1. Seller shall sell and deliver to buyer (description of goods) no later than (date).
Section 2. In consideration hereof, buyer shall pay (amount in dollars) in cash on delivery.

This describes a basic micro-economic phenomenon, namely, an *exchange* between *economic agents* of scarce *economic resources* that have utility.

REA Definitions (McCarthy, 1982) [1]

Economic Resource Economic resource are defined by Ijiri [1975, pp. 51-2] to be objects that (1) are scarce and have utility and (2) are under the control of an enterprise. In practice, the definition of this entity set can be considered equivalent to that given the term “asset” by the FASB [1979, pp. 51-7] with one exception: economic resources in the schema do not automatically include claims such as accounts-receivable.

Economic Agent Economic agents are “identifiable parties with discretionary power to use or dispose of economic resources”.

Economic Event Economic events are defined by Yu [1976, p. 256] as “a class of phenomena which reflects changes in scarce means [economic resources] resulting from production, exchange, consumption, and distribution.”

The REA has an *independent perspective* as opposed to the *trading partner perspective*.

[1] Proposed as an alternative to double-entry bookkeeping (it is easily shown that there are several maps from the REA model to the traditional ledger).

Contract Example #2: Legal Services

Abbreviated legal services contract:

Agreement to Provide Legal Services

Section 1. The attorney shall provide, on a non-exclusive basis, legal services up to (n) hours per month, and furthermore provide services in excess of (n) hours upon agreement.

Section 2. In consideration hereof, the company shall pay a monthly fee of (amount in dollars) before the 8th day of the following month and (rate) per hour for any services in excess of (n) hours 40 days after the receival of an invoice.

Section 3. This contract is valid 1/1-12/31, 2004.

Patterns in Contracts

Contracts are composed of following “patterns”:

- Specific commitments for the transmission of money, goods or services by whom to whom.
- Sequential execution of subcontracts (services first, payment later)
- Choices between subcontracts, in particular options (excess hours)
- Concurrent execution of subcontracts (each month)
- Repetition of subcontracts (repetion of monthly service)
- Time constraints (in particular deadlines) on individual commitments and whole (sub)contracts.

Real-Life Contracts

We analyzed the domain of commercial contracts by considering 15 standard contracts:

Agreement to Sell Goods	Sale with Installment Payment
General Contract	Agreement to Sell
Balloon Note	Contractor Agreement
Legal Services Agreement	The Danish Trade Law (Købeloven)
Website Development Contract	Lease Contract
Loan and Security Agreement	License Agreement
Operating Agreement	Supply Agreement
Manufacturing Agreement	

Observations from the 15 Contracts

($\star$ = Primary focus)

- Contracts have these main components:
  - Definitions
  - Structural description (processes) $\star$
  - Conditions on specific parts
  - Conditions on all parts
  - Specification of remedies
- Contract path is decided by four choice types:
  - Agents actively choose $\star$
  - Time chooses (say, an option expires)
  - Observables (cf. Peyton Jones/Eber) choose
  - Log of events chooses

Contract specification is very similar to protocol specification for networked systems.

Domain-specific languages

Goal: Capture contract patterns in domain-specific language

Hypothesis: DSL based on above contract patterns alone good basis for contract specification

Method for testing hypothesis:

- analysis of adequacy in application domain
- semantics-driven design and analysis of language

Atomic Contracts and Combinators

$\mathrm{Success}/\mathrm{Failure}$

The succeeded/failed contract with no commitments.

transmit$(A_1, A_2, R, T|P).c$

The commitment of agent A_1 to transmit resource R to agent A_2 at time T subject to predicate P.

$;$

A sequence of two contracts. Only when the first contract is reduced to $\mathrm{Success}$ can the execution of the next begin.

$\|$

The parallel and independent execution of two contracts.

$+$

The (non-deterministic) choice of exactly one of two contracts.

$f(\vec{a})$

The expansion to a named contract f with concrete arguments $\vec{a}$.

$\mathrm{letrec}\, f_i[\vec{X}_i] = c_i$ **in** c

Contract c with named contracts f_i with formal arguments X_i bound to c_i.

Example Contract #1: Sale of Goods

```
letrec
  sale [seller, buyer, goods, payment, t1, t2] =
          transmit (seller, buyer, goods, T | T < t1)
       || transmit (buyer, seller, payment, T | T < t2)
in
  sale ("McD", "Me", "Burger", 4 Euros, 7/1, 7/1)
```

Example Contract #2: Legal Services

```
letrec
  legal [lawyer, comp, hours, payment, extraprice, end, n] =
     (Success
      + transmit(lawyer, comp, H, T | n <= T and T < min(end,nextmonth(n)))
     (legal (lawyer, comp, hours, payment, extraprice, end, nextmonth(n))
      + Success)
  || transmit(comp,lawyer, payment, T | T <= nextmonth(n) + 10)
  || (transmit(lawyer, comp, invoice, T1 | obs(hoursspent,n) > hours and
                  (invoice.total = obs(hoursspent,n) - hours) * extraprice)
      .transmit(comp, lawyer, invoice.total, T2 | T2 <= T1 + 45)
      + Success)
in
  legal ("Lawyer X", "Me", 20, 10000 Euros, 1200 Euros, 12/31, 1/1)
```

(Part of) The Denotational Semantics

Trace tr a finite sequence of events

Contract C a set of traces

$$\begin{aligned}
\mathcal{C}[\![\text{Success}]\!]^{\gamma;\rho} &= \{\langle\rangle\} \\
\mathcal{C}[\![\text{Failure}]\!]^{\gamma;\rho} &= \emptyset \\
\mathcal{C}[\![\text{transmit}(A_1, A_2, R, T \mid P).\, c]\!]^{\gamma;\rho} &= \{(a_1, a_2, r, t)\, s : (a_1, a_2, r, t) \in \mathcal{E}, s \in \mathit{Tr} \mid \\
&\qquad \mathcal{R}[\![P]\!]^{\rho} = \text{true} \wedge s \in \mathcal{C}[\![c]\!]^{\gamma;\rho, A_1 \mapsto a_1, A_2 \mapsto a_2, R \mapsto r, T \mapsto t} \\
\mathcal{C}[\![c_1 + c_2]\!]^{\gamma;\rho} &= \mathcal{C}[\![c_1]\!]^{\gamma;\rho} \cup \mathcal{C}[\![c_2]\!]^{\gamma;\rho} \\
\mathcal{C}[\![c_1 \parallel c_2]\!]^{\gamma;\rho} &= \{s \mid \exists s_1 \in \mathcal{C}[\![c_1]\!]^{\gamma;\rho}, s_2 \in \mathcal{C}[\![c_2]\!]^{\gamma;\rho}.\ \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} \rightsquigarrow s\} \\
\mathcal{C}[\![c_1; c_2]\!]^{\gamma;\rho} &= \{s_1 s_2 \mid s_1 \in \mathcal{C}[\![c_1]\!]^{\gamma;\rho} \wedge s_2 \in \mathcal{C}[\![c_2]\!]^{\gamma;\rho}\}
\end{aligned}$$

Basic Questions

Residual contract C/s The residual rights and obligations of C after s has been executed.

Basic questions we want answered, given C and s:

- Is C/s nullable, that is completed or possibly completed successfully?
- Is C/s consistent (not necessarily completed, but can still be extended to a performaing trace s'), or is C/s inconsistent (s cannot be extended to a performing trace by any trace s'.)
- Given consistent C/s, is C/se consistent (e is compliant) or inconsistent (e is a breach of contract)?

Approach: Reduction Semantics

Define reduction semantics $C \xrightarrow{e} C'$, such that C' (in our language) represents C/e. Extends to $C \xrightarrow{s} C'$. Then answer above:

- Is C' equivalent with $\mathrm{Success} + C''$ for some C''?
- Is C' equivalent with $\mathrm{Failure}$ or is it not?
- If $C \xrightarrow{e} C'$, is C' equivalent with $\mathrm{Failure}$?

Advantage: Any analysis of contracts automatically extends to analysis of partially executed contracts.

Event Model

Figure 1: Contract Processing Module

Assume events and commitments match one-to-one. We use a discrete time model. Granularity can be chosen arbitrarily. Assume events arrive in ascending time order.

Example Contract #1 Revisited: Sale of Goods

Contract reduction using events:

```
          transmit (seller, buyer, goods, T | T < 7/1)
       || transmit (buyer, seller, 100, T | T < 8/1)

(seller, buyer, goods, 1/1) ->

          Success
       || transmit (buyer, seller, 100, T | T < 8/1)

(buyer, seller, 100, 7/19) ->

          Success
       || Success
```

Example Contract #3 (Compact Notation)

Contract reduction using events:

```
    ((t1 + Success);t2) || t2.t3

-e2->

    Success || t2.t3

-e3->

    Fail
```

Reduction is non-deterministic!

Non-determinism

Remedies in practice:

- Dynamically generated identifiers such as invoice numbers (intuitively corresponding to "tagging" of commitments) to determinize matching.
- Manual "matching" performed by bookkeepers, ensuing discrepancies and disagreements between partners resolved by "backtracking".

Remedy in semantics:

- Explicit tagging of commitments (see Andersen/Elsborg, 2003) or explicit representation of "routing" of event to target commitment
- Backtracking semantics: Deterministic semantics $C \xrightarrow{e} C'$ such that $C/e = C'$. (In nondeterministic semantics: if $C \xrightarrow{e} C'$ then $C' \subseteq C/e$

Reduction with Explicit Control

```
    ((t1 + Success);t2) || t2.t3

-e2,r->

    ((t1 + Success);t2) || t3

-e3,r->

    ((t1 + Success);t2) || Success
```

Reduction with Backtracking

```
    ((t1 + Success);t2) || t2.t3

-e2->

    (Success || t2.t3) + (((t1 + Success);t2) || t3)

-e3->

    Fail + (((t1 + Success);t2) || Success)
```

Some Rules from the Reduction Semantics (Backtracking)

$$\mathrm{D}, \rho \vdash_D \mathrm{Success} \xrightarrow{e} \mathrm{Failure} \qquad \mathrm{D}, \rho \vdash_D \mathrm{Failure} \xrightarrow{e} \mathrm{Failure}$$

$$\frac{\rho \models P[e/\vec{X}]}{\mathrm{D}, \rho \vdash_D \mathrm{transmit}(\vec{X} \mid P).\, c \xrightarrow{e} c[e/\vec{X}]}$$

$$\frac{\mathrm{D} \vdash c \text{ nullable} \quad \mathrm{D}, \rho \vdash_D c \xrightarrow{e} d \quad \mathrm{D}, \rho \vdash_D c' \xrightarrow{e} d'}{\mathrm{D}, \rho \vdash_D c; c' \xrightarrow{e} d; c' + d'}$$

$$\frac{\mathrm{D} \not\vdash c \text{ nullable} \quad \mathrm{D}, \rho \vdash_D c \xrightarrow{e} d \quad \mathrm{D}, \rho \vdash_D c' \xrightarrow{e} d'}{\mathrm{D}, \rho \vdash_D c; c' \xrightarrow{e} d; c'}$$

Multiple Unfoldings

```
f[t] = transmit(a,b,r,t) || f(t)
```

En event (a, b, r, t) can match any of the infinitely many parallel transmits. The denotation of the contract is $\emptyset$.

Contracts should be *guarded* to prevent infinite unfolding.

Intuitively, a contract is guarded iff recursive unfolding is guarded by commitments.

$$\mathrm{D} \vdash \text{Success guarded} \qquad \mathrm{D} \vdash \text{Failure guarded} \qquad \mathrm{D} \vdash \text{transmit}(\vec{X} \mid P).\, c \text{ guarded}$$

$$\frac{\mathrm{D} \nvdash c \text{ nullable} \quad \mathrm{D} \vdash c \text{ guarded}}{\mathrm{D} \vdash c; c' \text{ guarded}} \qquad \frac{\mathrm{D} \vdash c \text{ nullable} \quad \mathrm{D} \vdash c \text{ guarded} \quad \mathrm{D} \vdash c' \text{ guarded}}{\mathrm{D} \vdash c; c' \text{ guarded}}$$

Contract Analysis: Task List and Next Point of Interest

Given a portfolio of running contracts, an agent needs to know:

- What commitments and choices are ready to be executed by the agent?
- When is the next time a contract changes (say, a deadline expires)?

Predicate Languages

Consider a simply predicate language with expressions on the form `[a;b]`, where $a, b \in \mathbb{R} \cup \{-\infty, \infty\}$. An expression `[a;b]` is valid iff $a \leq b$.

Given a time t the predicate $[a; b]$ is true if and only if $t \in [a; b]$.

Task List

```
fun tasks a t Success      = []
  | tasks a t Fail         = []
  | tasks a t
    transmit(a1,a2,r,t|[a;b]).c = if a = a1 andalso t in [a;b]
                                  then [trans(a1,a2,r,t|[a;b])]
                                  else []
  | tasks a t (c1 + c2)    = choose(tasks a t c1, tasks a t c2)
  | tasks a t (c1 ; c2)    = if nullable c1
                             then if tasks a t c1 <> []
                                  then choose(tasks a t c1, tasks a t c2)
                                  else tasks a t c2
                             else tasks a t c1
  | tasks a t (c1 || c2)   = (tasks a t c1) @ (tasks a t c2)
  | tasks a t (f a)        = tasks a t (expand f a)
```

Next Point of Interest (NPOI)

```
fun npoi t Success     = INF
  | npoi t Fail        = INF
  | npoi t
    transmit(a1,a2,r,t|[a;b]).c = if t <= a then a else
                                  if t <= b then b else INF
  | npoi t (c1 + c2)   = min(npoi t c1, npoi t c2)
  | npoi t (c1 ; c2)   = min(npoi t c1, if nullable c1
                                        then npoi t c2 else INF)
  | npoi t (c1 || c2)  = min(npoi t c1, npoi t c2)
  | npoi t (f a)       = npoi t (expand f a)
```

Analysis Example: Legal Services

Status of (guarded) legal services contract C on 2/4:

```
   (transmit (lawyer, lawyer, nohours, T | [3/1;3/1])
    + transmit(lawyer, comp, 20, T | [2/1;3/1]));
   (legal (lawyer, comp, hours, payment, extraprice, end, nextmonth(n))
    + Success)
|| transmit(comp, lawyer, payment, T | [3/1;3/11])
|| (transmit(lawyer, comp, invoice, T1 | [3/1;inf])
    .transmit(comp, lawyer, invoice.total, T2 | [T1;T1+45]) + Success)
|| transmit(comp, lawyer, payment, T | [2/1;2/11])
|| transmit(comp, lawyer, invoice.total, T2 | [2/3;3/18])

tasks 2/4 comp C =
    [transmit(comp, lawyer, payment, T | [2/1;2/11]),
     transmit(comp, lawyer, invoice.total, T2 | [2/3;3/18])]
npoi 2/4 C = 2/11
```

More Contract Analyses

- Account payable/accounts receivable
- Termination: (a) earliest termination (b) latest termination
- Check for no race conditions
- Deadlock
- Valuation (with respect to some agent A)
- Inventory line, supply requirement, resource flow profile (optimistic/pessimistic) - forecasting, supply-chain management
- General model checking for business rules: (a) static (b) dynamic/runtime (Timed LTL checking)

Discussion

- Good fit to domain despite language paucity
- Separation of predicate language and contract language seems good
- Event model should perhaps include other event types (e.g. timer events)
- Further experience with predicate layer is desirable
- Exploitation of algebraic properties using denotational model should be explored further

Related Work

William McCarthy: Contract state machine

Peyton Jones/Eber: Compositional contracts (only trading partner perspective)

Formal calculi: π-Calculus, CSP (Communicating Sequential Processes)

Many standardization initiatives: BPEL4WS, CrossFlow, UEML, etc.

Extensions

- Consider business rules (policies), perhaps using timed LTL or a contract “overlaying” operator
- Assume events and commitments do not match one-to-one
- User interface using a simple, tabular form

Contracts Not Considered

Some contract types and why they were left out:

- Employment agreements, severance, retirement plans *require non-static agents* regarded dually as *separate parties* and employees.
- Non-disclosure agreements, codes of ethics impose conditions on *all existing contracts simultaneously* and specify things *not to do* in an often *subjective manner*.
- Arbitration, remedies, and trade laws require the encoding of a large *body of statutes* and a fairly involved *exception handling mechanism*.
- Chapter 11s, separation, partnerships, bankruptcy, mergers, acquisitions *modify the structure of the enterprise*.
- Legal haggles (lawsuit, counter-suit, settlements) are *subjective*

Contract Process

Five phases (ISO OpenEDI, Part 1, 2003; Part 4, 2004; UBAC):

1. Planning
2. Identification
3. Negotiation
4. **Actualization**
5. Post-actualization

Syntax for Contract Specifications

$$\Gamma; \Delta \vdash \mathrm{Success} : \mathrm{Contract} \qquad \Gamma; \Delta \vdash \mathrm{Failure} : \mathrm{Contract}$$

$$\frac{\Gamma(f) = \vec{\tau} \rightarrow \mathrm{Contract} \quad \Delta \vdash \vec{a} : \vec{\tau}}{\Gamma; \Delta \vdash f(\vec{a}) : \mathrm{Contract}} \qquad \frac{\begin{array}{c}\Delta' = \Delta, A_1 : \mathrm{Agent}, A_2 : \mathrm{Agent}, R : \mathrm{Resource}, T : \mathrm{Time} \\ \Gamma; \Delta' \vdash c : \mathrm{Contract} \\ \Delta' \vdash P : \mathrm{Boolean}\end{array}}{\Gamma; \Delta \vdash \mathrm{transmit}(A_1, A_2, R, T \mid P).\, c : \mathrm{Contract}}$$

$$\frac{\Gamma; \Delta \vdash c_1 : \mathrm{Contract} \quad \Gamma; \Delta \vdash c_2 : \mathrm{Contract}}{\Gamma; \Delta \vdash c_1 + c_2 : \mathrm{Contract}} \qquad \frac{\Gamma; \Delta \vdash c_1 : \mathrm{Contract} \quad \Gamma; \Delta \vdash c_2 : \mathrm{Contract}}{\Gamma; \Delta \vdash c_1 \;\&\; c_2 : \mathrm{Contract}}$$

$$\frac{\Gamma; \Delta \vdash c_1 : \mathrm{Contract} \quad \Gamma; \Delta \vdash c_2 : \mathrm{Contract}}{\Gamma; \Delta \vdash c_1 \parallel c_2 : \mathrm{Contract}} \qquad \frac{\Gamma; \Delta \vdash c_1 : \mathrm{Contract} \quad \Gamma; \Delta \vdash c_2 : \mathrm{Contract}}{\Gamma; \Delta \vdash c_1 ; c_2 : \mathrm{Contract}}$$

$$\frac{\begin{array}{c}\Gamma = \{f_i \mapsto \tau_{i1} \times \ldots \times \tau_{in_i} \to \text{Contract}\}_{i=1}^{m} \\ \Gamma; \Delta, X_{i1} : \tau_{i1}, \ldots, X_{in_i} : \tau_{in_i} \vdash c_i : \text{Contract}\end{array}}{\Delta \vdash \{f_i[\vec{X}_i] = c_i\}_{i=1}^{m}\} : \Gamma}$$

$$\frac{\Delta \vdash \{f_i[\vec{X}_i] = c_i\}_{i=1}^{m} : \Gamma \quad \Gamma; \Delta \vdash c : \text{Contract}}{\Delta \vdash \text{letrec}\,\{f_i[\vec{X}_i] = c_i\}_{i=1}^{m} \text{ in } c : \text{Contract}}$$

Denotational Semantics

$$\begin{aligned}
Dom[\![\text{Boolean}]\!] &= \{\text{true}, \text{false}\} \\
Dom[\![\text{Agent}]\!] &= \mathcal{A} \\
Dom[\![\text{Resource}]\!] &= \mathcal{R} \\
Dom[\![\text{Time}]\!] &= \mathcal{T} \\
\mathcal{E} &= \mathcal{A} \times \mathcal{A} \times \mathcal{R} \times \mathcal{T} \\
Tr &= \mathcal{E}^* \\
Dom[\![\text{Contract}]\!] &= 2^{Tr} \\
Dom[\![\tau_1 \times \ldots \times \tau_n \rightarrow \text{Contract}]\!] &= Dom[\![\tau_1]\!] \times \ldots \times Dom[\![\tau_n]\!] \rightarrow 2^{Tr}
\end{aligned}$$

$$\begin{array}{rcl}
\mathcal{C}[\![\mathrm{Success}]\!]^{\gamma;\rho} & = & \{\langle\rangle\} \\
\mathcal{C}[\![\mathrm{Failure}]\!]^{\gamma;\rho} & = & \emptyset \\
\mathcal{C}[\![f(\vec{a})]\!]^{\gamma;\rho} & = & \gamma(f)(\mathcal{R}[\![\vec{a}]\!]^{\rho}) \\
\mathcal{C}[\![\mathrm{transmit}(A_1, A_2, R, T \mid P).\, c]\!]^{\gamma;\rho} & = & \{(a_1, a_2, r, t)\, s : (a_1, a_2, r, t) \in \mathcal{E}, s \in \mathit{Tr} \mid \\
 & & \mathcal{R}[\![P]\!]^{\rho} = \mathrm{true} \wedge s \in \mathcal{C}[\![c]\!]^{\gamma;\rho, A_1 \mapsto a_1, A_2 \mapsto a_2, R \mapsto r, T \mapsto t} \\
\mathcal{C}[\![c_1 + c_2]\!]^{\gamma;\rho} & = & \mathcal{C}[\![c_1]\!]^{\gamma;\rho} \cup \mathcal{C}[\![c_2]\!]^{\gamma;\rho} \\
\mathcal{C}[\![c_1 \;\;\&\;\; c_2]\!]^{\gamma;\rho} & = & \mathcal{C}[\![c_1]\!]^{\gamma;\rho} \cap \mathcal{C}[\![c_2]\!]^{\gamma;\rho} \\
\mathcal{C}[\![c_1 \parallel c_2]\!]^{\gamma;\rho} & = & \{s \mid \exists s_1 \in \mathcal{C}[\![c_1]\!]^{\gamma;\rho}, s_2 \in \mathcal{C}[\![c_2]\!]^{\gamma;\rho}. \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} \rightsquigarrow s\} \\
\mathcal{C}[\![c_1; c_2]\!]^{\gamma;\rho} & = & \{s_1 s_2 \mid s_1 \in \mathcal{C}[\![c_1]\!]^{\gamma;\rho} \wedge s_2 \in \mathcal{C}[\![c_2]\!]^{\gamma;\rho}\} \\
\mathcal{D}[\![\{f_i[\vec{X}_i] = c_i\}_{i=1}^{m}]\!]^{\rho} & = & \mathit{least}\,\gamma : \gamma = \{f_i \mapsto \lambda\vec{x}_i.\mathcal{C}[\![c_i]\!]^{\gamma;\rho,\vec{X}_i \mapsto \vec{x}_i}\}_{i=1}^{m} \\
\mathcal{C}[\![\mathrm{letrec}\,\{f_i[\vec{X}_i] = c_i\}_{i=1}^{m} \text{ in } c]\!]^{\rho} & = & \mathcal{C}[\![c]\!]^{\mathcal{D}[\![\{f_i[\vec{X}_i]=c_i\}_{i=1}^{m}]\!]^{\rho};\rho}
\end{array}$$

Residuation

We wish to monitor contracts as events come in and determine if events are performing or non-performing. If an event is performing, we must be able to find the remainder of the contract (henceforth the *residual* contract).

$$\mathcal{C}[\![c/e]\!]^{\gamma;\rho} = \{s' : s' \in \mathit{Tr} \mid es' \in \mathcal{C}[\![c]\!]^{\gamma;\rho}\}$$

Let us write $\mathrm{D}, \rho \models c = c'$ if $\mathcal{C}[\![c]\!]^{\gamma;\rho} = \mathcal{C}[\![c']\!]^{\gamma;\rho}$ where $\gamma = \mathcal{D}[\![\mathrm{D}]\!]^{\rho}$.

Residuation Equalities

$$\mathrm{D} \models \mathrm{Success}/e = \mathrm{Failure} \qquad \mathrm{D} \models \mathrm{Failure}/e = \mathrm{Failure}$$

$$\mathrm{D} \models f(\vec{a})/e = c[\vec{a}/\vec{X}]/e \text{ if } (f[\vec{X}] = c) \in \mathrm{D}$$

$$\mathrm{D} \models \mathrm{transmit}(A_1, A_2, R, T \mid P).\, c/(a_1, a_2, r, t) = \begin{cases} c[a_1/A_1, a_2/A_2, r/R, t/T] & \text{if } \mathcal{R}[\![P]\!]^{\emptyset} = \mathrm{true} \\ \mathrm{Failure} & \text{otherwise} \end{cases}$$

$$\mathrm{D} \models (c_1 + c_2)/e = c_1/e + c_2/e$$

$$\mathrm{D} \models (c_1 \ \& \ c_2)/e = c_1/e \ \& \ c_2/e$$

$$\mathrm{D} \models (c_1 \parallel c_2)/e = c_1/e \parallel c_2 + c_1 \parallel c_2/e$$

$$\mathrm{D} \models (c_1; c_2)/e = \begin{cases} c_1/e; c_2 + c_2/e & \text{if } c_1 \text{ contains } \langle\rangle \\ c_1/e; c_2 + & \text{otherwise} \end{cases}$$

Guarded Contracts

To prevent uncontrolled unfolding and infinite residuation, we introduce guarding.

Intuitively, a contract is guarded iff unfolding is guarded by commitments. (Analogously, if `FIRST()` set computation terminates.)

`f[a] = transmit(a1,a2,r,t|true) || f(a+1)` is not guarded

`f[a] = transmit(a1,a2,r,t|true); f(a+1)` is guarded

(Exercise for the audience: what is wrong with both contracts?)

Guarded Contracts

$$\mathrm{D} \vdash \text{Success guarded} \qquad \mathrm{D} \vdash \text{Failure guarded}$$

$$\mathrm{D} \vdash \text{transmit}(\vec{X} \mid P).\, c \text{ guarded}$$

$$\frac{\mathrm{D} \vdash c \text{ guarded} \quad (f[\vec{X}] = c) \in \mathrm{D}}{\mathrm{D} \vdash f(\vec{a}) \text{ guarded}} \qquad \frac{\mathrm{D} \vdash c \text{ guarded} \quad \mathrm{D} \vdash c' \text{ guarded}}{\mathrm{D} \vdash c + c' \text{ guarded}}$$

$$\frac{\mathrm{D} \vdash c \text{ guarded} \quad \mathrm{D} \vdash c' \text{ guarded}}{\mathrm{D} \vdash c \;\;\&\;\; c' \text{ guarded}} \qquad \frac{\mathrm{D} \vdash c \text{ guarded} \quad \mathrm{D} \vdash c' \text{ guarded}}{\mathrm{D} \vdash c \parallel c' \text{ guarded}}$$

$$\frac{\mathrm{D} \nvdash c \text{ nullable} \quad \mathrm{D} \vdash c \text{ guarded}}{\mathrm{D} \vdash c; c' \text{ guarded}} \qquad \frac{\mathrm{D} \vdash c \text{ nullable} \quad \mathrm{D} \vdash c \text{ guarded} \quad \mathrm{D} \vdash c' \text{ guarded}}{\mathrm{D} \vdash c; c' \text{ guarded}}$$

$$\frac{\mathrm{D} \vdash c \text{ guarded}}{\vdash \text{letrec}\, \mathrm{D} \;\mathsf{in}\; c \text{ guarded}}$$

Nullable Contracts

$$\mathrm{D} \vdash \mathrm{Success\ nullable} \qquad \frac{\mathrm{D} \vdash c \text{ nullable} \quad (f[\vec{X}] = c) \in \mathrm{D}}{\mathrm{D} \vdash f(\vec{a}) \text{ nullable}}$$

$$\frac{\mathrm{D} \vdash c \text{ nullable}}{\mathrm{D} \vdash c + c' \text{ nullable}} \qquad \frac{\mathrm{D} \vdash c' \text{ nullable}}{\mathrm{D} \vdash c + c' \text{ nullable}}$$

$$\frac{\mathrm{D} \vdash c \text{ nullable} \quad \mathrm{D} \vdash c' \text{ nullable}}{\mathrm{D} \vdash c \ \& \ c' \text{ nullable}} \qquad \frac{\mathrm{D} \vdash c \text{ nullable} \quad \mathrm{D} \vdash c' \text{ nullable}}{\mathrm{D} \vdash c \parallel c' \text{ nullable}}$$

$$\frac{\mathrm{D} \vdash c \text{ nullable} \quad \mathrm{D} \vdash c' \text{ nullable}}{\mathrm{D} \vdash c; c' \text{ nullable}}$$

$$\frac{\mathrm{D} \vdash c \text{ nullable}}{\vdash \mathrm{letrec}\, \mathrm{D} \text{ in } c \text{ nullable}}$$

Deterministic Reduction (Backtracking/Deferred Matching)

$$\mathrm{D}, \rho \vdash_D \text{Success} \xrightarrow{e} \text{Failure} \qquad \mathrm{D}, \rho \vdash_D \text{Failure} \xrightarrow{e} \text{Failure}$$

$$\frac{\rho \models P[e/\vec{X}]}{\mathrm{D}, \rho \vdash_D \text{transmit}(\vec{X} \mid P).\, c \xrightarrow{e} c[e/\vec{X}]} \qquad \frac{\mathrm{D}, \rho \vdash_D c[\vec{a}/\vec{X}] \xrightarrow{e} c' \quad (f[\vec{X}] = c) \in \mathrm{D}}{\mathrm{D}, \rho \vdash_D f(\vec{a}) \xrightarrow{e} c'}$$

$$\frac{\mathrm{D}, \rho \vdash_D c \xrightarrow{e} d \quad \mathrm{D}, \rho \vdash_D c' \xrightarrow{e} d'}{\mathrm{D}, \rho \vdash_D c + c' \xrightarrow{e} d + d'} \qquad \frac{\mathrm{D}, \rho \vdash_D c \xrightarrow{e} d \quad \mathrm{D}, \rho \vdash_D c' \xrightarrow{e} d'}{\mathrm{D}, \rho \vdash_D c \ \& \ c' \xrightarrow{e} d \ \& \ d'}$$

$$\frac{\mathrm{D}, \rho \vdash_D c \xrightarrow{e} d \quad \mathrm{D}, \rho \vdash_D c' \xrightarrow{e} d'}{\mathrm{D}, \rho \vdash_D c \parallel c' \xrightarrow{e} c \parallel d' + d \parallel c'} \qquad \frac{\mathrm{D} \vdash c \text{ nullable} \quad \mathrm{D}, \rho \vdash_D c \xrightarrow{e} d \quad \mathrm{D}, \rho \vdash_D c' \xrightarrow{e} d'}{\mathrm{D}, \rho \vdash_D c; c' \xrightarrow{e} d; c' + d'}$$

$$\frac{\mathrm{D} \not\vdash c \text{ nullable} \quad \mathrm{D}, \rho \vdash_D c \xrightarrow{e} d \quad \mathrm{D}, \rho \vdash_D c' \xrightarrow{e} d'}{\mathrm{D}, \rho \vdash_D c; c' \xrightarrow{e} d; c'} \qquad \frac{\mathrm{D}, \rho \vdash_D c \xrightarrow{e} c'}{\rho \vdash_D \text{letrec}\, \mathrm{D} \text{ in } c \xrightarrow{e} \text{letrec}\, \mathrm{D} \text{ in } c'}$$

Nondeterministic Reduction (Eager Matching)

$$\mathrm{D}, \rho \vdash_N \text{Success} \xrightarrow{e} \text{Failure} \qquad \mathrm{D}, \rho \vdash_N \text{Failure} \xrightarrow{e} \text{Failure}$$

$$\frac{\rho \models P[e/\vec{X}]}{\mathrm{D}, \rho \vdash_N \text{transmit}(\vec{X} \mid P).\, c \xrightarrow{e} c[e/\vec{X}]} \qquad \frac{(f[\vec{X}] = c) \in \mathrm{D}}{\mathrm{D}, \rho \vdash_N f(\vec{a}) \xrightarrow{\tau} c[\vec{a}/\vec{X}]}$$

$$\mathrm{D}, \rho \vdash_N c + c' \xrightarrow{\tau} c \qquad \mathrm{D}, \rho \vdash_N c + c' \xrightarrow{\tau} c'$$

$$\frac{\mathrm{D}, \rho \vdash_N c \xrightarrow{e} d \quad \mathrm{D}, \rho \vdash_N c' \xrightarrow{e} d'}{\mathrm{D}, \rho \vdash_N c \;\&\; c' \xrightarrow{e} d \;\&\; d'} \qquad \frac{\mathrm{D}, \rho \vdash_N c \xrightarrow{\tau} d}{\mathrm{D}, \rho \vdash_N c \;\&\; c' \xrightarrow{\tau} d \;\&\; c'}$$

$$\frac{\mathrm{D}, \rho \vdash_N c' \xrightarrow{\tau} d'}{\mathrm{D}, \rho \vdash_N c \;\&\; c' \xrightarrow{\tau} c \;\&\; d'} \qquad \mathrm{D}, \rho \vdash_N \text{Success} \;\&\; \text{Success} \xrightarrow{\tau} \text{Success}$$

$$\frac{\mathrm{D}, \rho \vdash_N c \xrightarrow{\lambda} d}{\mathrm{D}, \rho \vdash_N c \parallel c' \xrightarrow{\lambda} d \parallel c'} \qquad \frac{\mathrm{D}, \rho \vdash_N c' \xrightarrow{\lambda} d'}{\mathrm{D}, \rho \vdash_N c \parallel c' \xrightarrow{\lambda} c \parallel d'}$$

$$\mathrm{D}, \rho \vdash_N \mathrm{Success} \parallel c \xrightarrow{\tau} c \qquad \mathrm{D}, \rho \vdash_N c \parallel \mathrm{Success} \xrightarrow{\tau} c$$

$$\frac{\mathrm{D}, \rho \vdash_N c \xrightarrow{\lambda} d}{\mathrm{D}, \rho \vdash_N c; c' \xrightarrow{\lambda} d; c'} \qquad \mathrm{D}, \rho \vdash_N \mathrm{Success}; c' \xrightarrow{\tau} c' \qquad \frac{\mathrm{D}, \rho \vdash_N c \xrightarrow{e} c'}{\rho \vdash_N \mathrm{letrec}\,\mathrm{D}\ \textsf{in}\ c \xrightarrow{e} \mathrm{letrec}\,\mathrm{D}\ \textsf{in}\ c'}$$

Deterministic Reduction (Eager Matching with Explicit Control)

- All nondeterministsm can be reduced to a series of choices and routing decisions!
- We express routing decisions as an element of $D = \{f, s, l, r\}^*$.
- A control-annotated event then is an element of $D \times E$.

Deterministic Reduction (Eager Matching with Explicit Choice)

$$\mathrm{D}, \rho \vdash_C \mathrm{Success} \xrightarrow{e} \mathrm{Failure}$$

$$\mathrm{D}, \rho \vdash_C \mathrm{Failure} \xrightarrow{e} \mathrm{Failure}$$

$$\frac{\rho \models P[e/\vec{X}]}{\mathrm{D}, \rho \vdash_C \mathrm{transmit}(\vec{X} \mid P).\, c \xrightarrow{e} c[e/\vec{X}]}$$

$$\frac{(f[\vec{X}] = c) \in \mathrm{D}}{\mathrm{D}, \rho \vdash_C f(\vec{a}) \xrightarrow{\tau} c[\vec{a}/\vec{X}]} \qquad \mathrm{D}, \rho \vdash_C c + c' \xrightarrow{\tau} c \qquad \mathrm{D}, \rho \vdash_C c + c' \xrightarrow{\tau} c'$$

$$\frac{\mathrm{D}, \rho \vdash_C c \xrightarrow{\lambda} d \quad \mathrm{D}, \rho \vdash_C c' \xrightarrow{\lambda} d'}{\mathrm{D}, \rho \vdash_C c \ \& \ c' \xrightarrow{\lambda} d \ \& \ d'}$$

$$\frac{\mathrm{D}, \rho \vdash_C c \xrightarrow{\lambda} d}{\mathrm{D}, \rho \vdash_C c \parallel c' \xrightarrow{e} d \parallel c'} \qquad \frac{\mathrm{D}, \rho \vdash_C c' \xrightarrow{e} d'}{\mathrm{D}, \rho \vdash_C c \parallel c' \xrightarrow{e} c \parallel d'}$$

$$\frac{\mathrm{D}, \rho \vdash_C c \xrightarrow{\lambda} d}{\mathrm{D}, \rho \vdash_C c; c' \xrightarrow{\lambda} d; c'} \qquad \mathrm{D}, \rho \vdash_C \mathrm{Success}; c' \xrightarrow{\tau} c'$$

$$\frac{\mathrm{D}, \rho \vdash_C c \xrightarrow{e} c'}{\rho \vdash_C \mathrm{letrec}\,\mathrm{D}\ \mathsf{in}\ c \xrightarrow{e} \mathrm{letrec}\,\mathrm{D}\ \mathsf{in}\ c'}$$

Properties of Reduction Semantics

Proposition 1. *Controlled reduction is deterministic: For all* c, c', c'', E*, if* $c \xrightarrow{E} c'$ *and* $c \xrightarrow{E} c''$ *then* $c' = c''$*.*

Theorem 2. *For all* c, e *there exists unique* c' *such that* $c \xrightarrow{e} c'$ *(with deferred matching) and* $c/e = \mathcal{C}[\![c']\!]$*.*