



Compositional contract specification for REA

Fritz Henglein
Department of Computer Science, University of Copenhagen (DIKU)

FLACOS 2007
October 9-10, 2007



Overview

- Background:
 - DIKU
 - TOPPS
- Resources/Events/Agents
- Process-oriented ERP architecture
- Contracts

DIKU: Department of Computer Science, University of Copenhagen



- Established 1970 by Peter Naur
- Research groups:
 - ALGO: algorithmics & operations research
 - *TOPPS: programming language theory & software technology*
 - IMAGE: theory and application of image analysis and synthesis (computer vision & game animation)
 - DISTLAB: distributed systems (incl. grid computing, database systems, sensor networks)
 - HCI: system development & human-computer interaction



TOPPS: Theory and practice of programming languages

- Faculty:
 - Neil D. Jones (language-based complexity theory, partial evaluation; prof.emer.)
 - Fritz Henglein (type systems, software technology)
 - Robert Glück (supercompilation)
 - Jyrki Katajainen (algorithms, software development)
 - Torben Mogensen (domain-specific languages)
 - Julia Lawall (language-based techniques in OSs)
 - Klaus Grue (logic; on leave)
 - Andrzej Filinski (programming language theory)
 - Jakob Grue Simonsen (computability theory)

Dynamic content
(process descriptions)

- Recurring theme:
Semantics-based program manipulation

Make sure, it's done
correctly and efficiently

Programs as data; analyze and
transform them



General goal

- *To be able to specify, automate, analyze and monitor business processes in an adaptive software architecture.*



Resources-Events-Agents (REA)

- Accounting model proposed by William E. McCarthy (1983)
- Accounting basis: Theory of the Firm
 - Economic resources: Scarce resources (materials, employee time)
 - Economic agents: Individuals, departments, companies
 - Economic events:
 - Transfer of resources between agents
 - Production of resources from other resources
 - (Transportation of resources from one location to another)
 - (...)
- Modeling basis: Entity-Relationship Modeling
 - Focus on static (data model) aspects
 - Difficult to capture dynamic (process) aspects

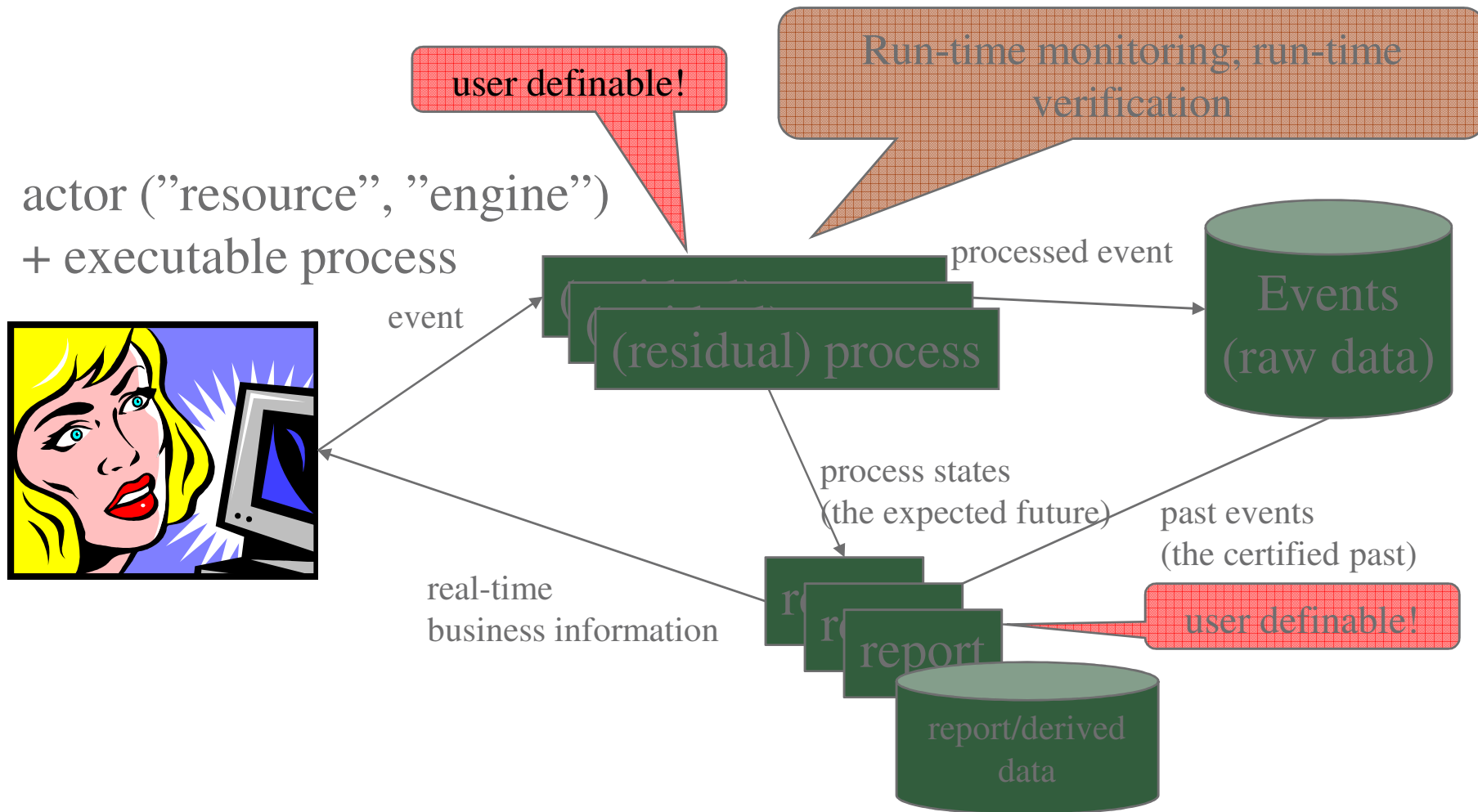


Event-driven architecture

- *Event-driven architecture (EDA)*: Software architecture where system state transitions are driven by processing (data representing) real-world *events*.
- *Process-oriented EDA*: EDA where event processing is matched against an explicit *process specification* that models *expected behavior*.
 - Matching a complete (sub)process specification can be thought of as a *compound (complex) event*.
- *Commercial contract*: process specification of exchange of goods, services and money.
- *Contract-based EDA*: Formal contract specifications as process specifications in EDA



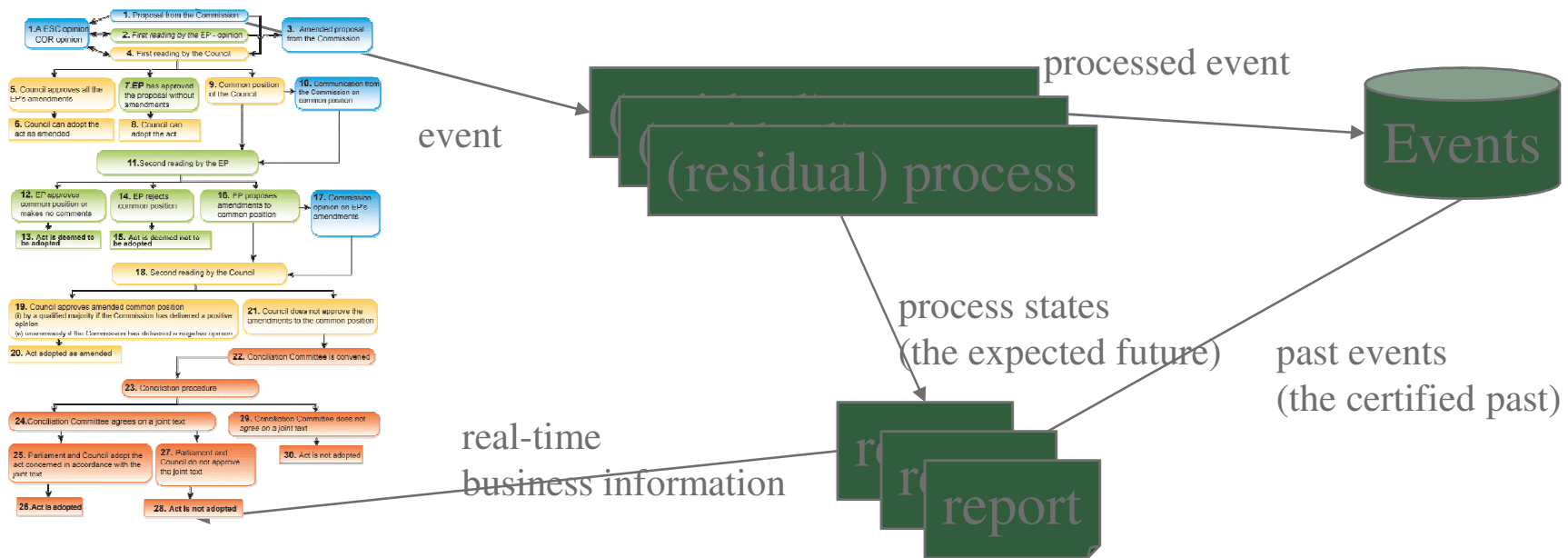
Contract-based event-driven architecture





Contract-based event-driven architecture

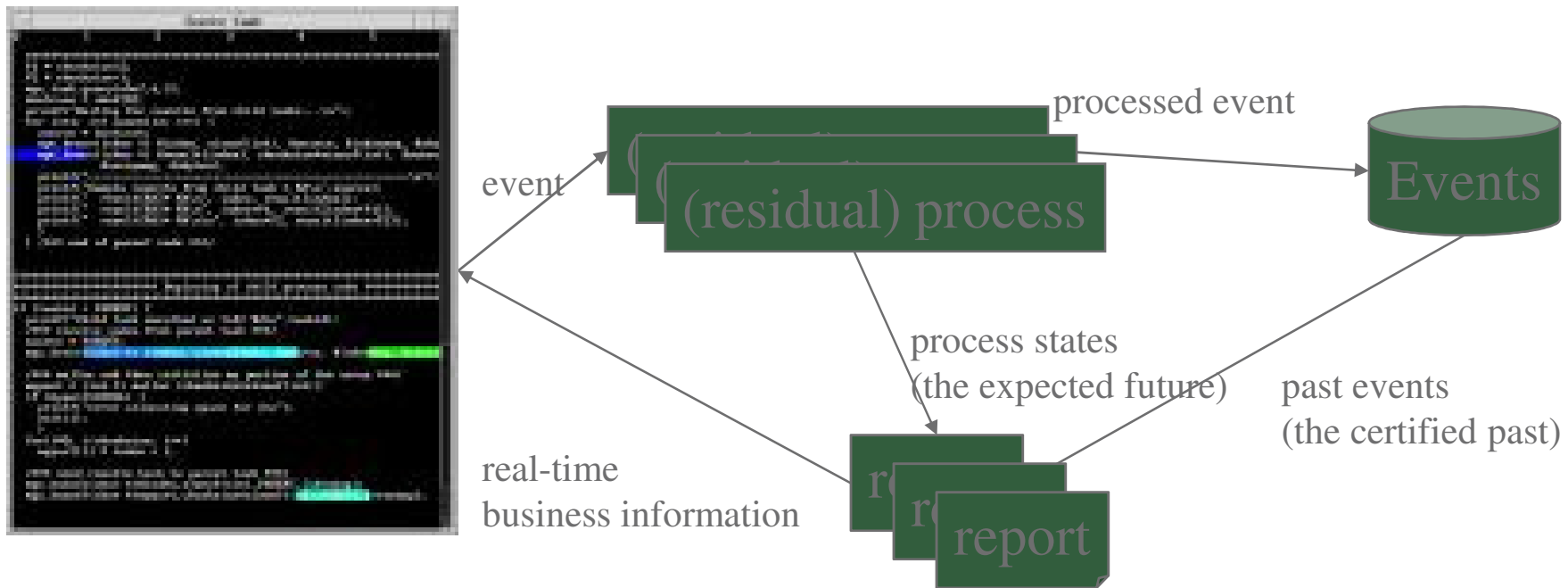
workflow engine
+ workflow specification





Contract-based event-driven architecture

interpreter
+ business process source code



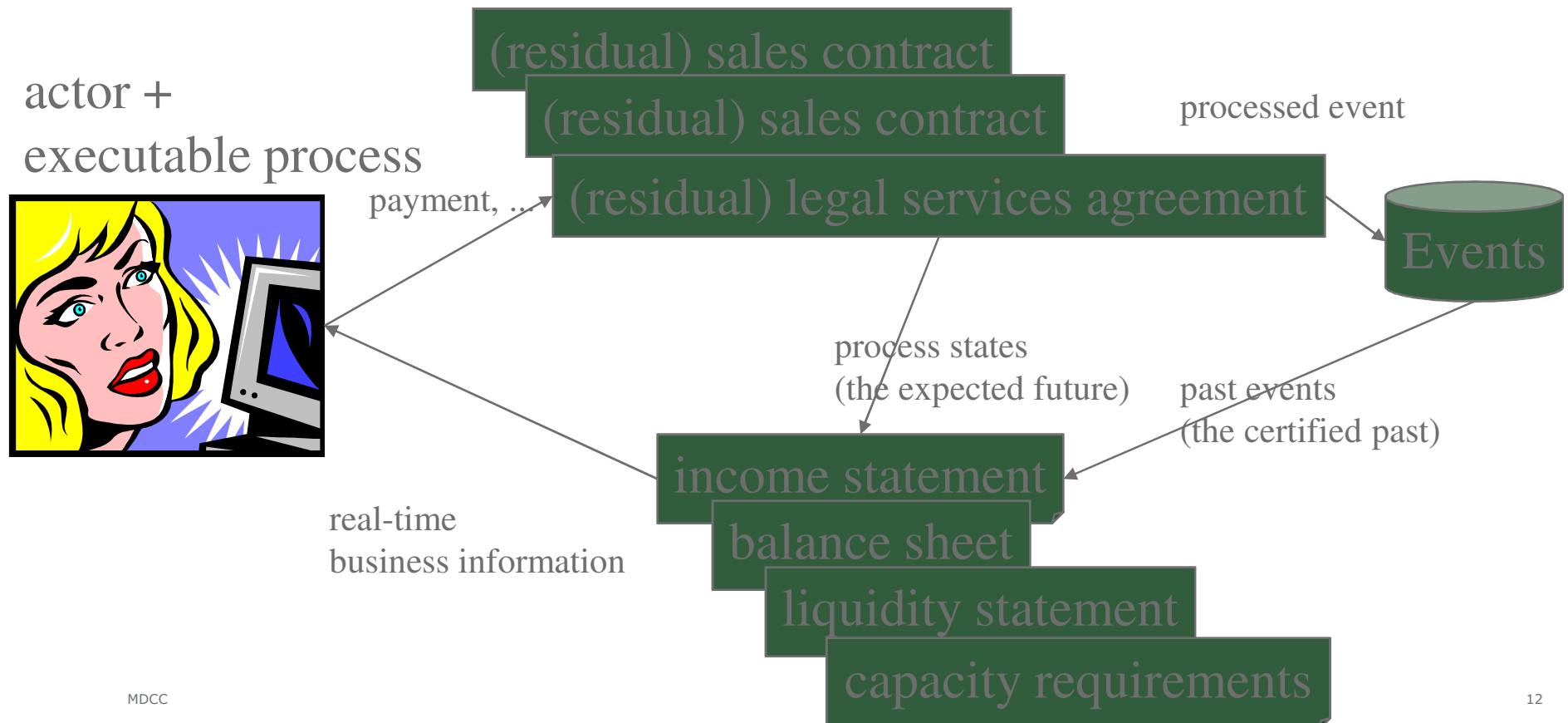


Architecture, ontologically

- **Process type** (protocol, behavior type)
- **Event log** (journal, data repository)
- **Report** (incl. business analytics, KPI, databases (!), auditing, valuation, task lists, tax statements, etc.)
- **Actor + executable process** (internal process):
 - **black-box** (internal process unknown or infeasible):
 - human
 - legacy system
 - application coded in GP PL (C, Java, C#...)
 - **white-box** (specification of internal process known):
 - workflows (“graphical” descriptions)
 - programs in business process languages (“textual” descriptions)



Contract-based event-driven architecture: ERP





Paper-Based Contract-Handling is Costly

- No formal representation results in:
 - Manual handling in auxiliary systems
 - Ad-hoc deadline management
 - Ambiguous semantics
 - Time-consuming valuation
 - Cannot report on future events
 - No coordination with production planning or supply-chain management
 - Missed opportunities (call options etc.)
- Potential benefits of formal representation:
 - Alleviate the problems above
 - Several consistency checks at time of writing contract
 - (Semi-)automated contract analysis (incl. valuation)
(DSL programs are in a sense "intelligent data")



We Analyzed 15 Full-Length Contracts

- Data model
- Structure

Agreement to Sell Goods

General Contract

Balloon Note

Legal Services Agreement

Website Development Contract

Loan and Security Agreement

Operating Agreement

Manufacturing Agreement

Sale with Installment Payment

Agreement to Sell

Contractor Agreement

The Danish Trade Law

Lease Contract

License Agreement

Supply Agreement



Syntax

Basic REA ontology (represents a commitment)
Other basic event specifications possible

Success/Failure

The success or 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 (afterwards do contract c).

$c_1; c_2$

A sequence of two contracts. The first contract must be reduced to Success before the second can begin.

$c_1 \parallel c_2$

Parallel, independent execution of two contracts.

$c_1 + c_2$

(Non-deterministic) choice between two contracts.

$f(\vec{a})$

Expansion to body of contract f with arguments \vec{a} .

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: 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 receipt of an invoice.
- **Section 3.** This contract is valid 1/1-12/31, 2008.



Example: Legal Services Agreement Code

```
letrec
extra (att, com, invoice, pay) =
  ( Success
  + transmit (att, com, invoice, T2).
    transmit (com, att, pay, T3 | T3 <= T2 + 45d))

legal (att, com, fee, invoice, pay, n, m, end) =
  transmit (att, com, H, T | n < T and T <= m).
    ( extra (att, com, invoice, pay)
    || transmit (com att, fee, T | T <= m + 8d)
    || ( legal (att, com, fee, invoice, pay, m, min(m + 30d,end), end)
      + transmit (att, com, end, T | end <= T)))

in
  legal ("Attorney", "Company", 10000, invoice, pay, 0, 30, 360)
```



When Is a Contract Satisfied?

- Contracts denote sets of finite traces. A trace is a finite sequence of events:

$$s ::= \langle \rangle \mid \text{transmit}(a1, a2, r, t) s$$

- So contracts classify a given trace as performing or nonperforming. Formally:

$$\{s \mid \delta' \vdash_D^\delta s : c\}$$



The Satisfaction Relation

$$\frac{\delta \oplus \delta'' \models P \quad \delta'' \vdash_D^\delta s : c \quad (\delta'' = \delta' \oplus \{X \mapsto v\})}{\delta' \vdash_D^\delta \text{transmit}(v) s : \text{transmit}(X|P).c}$$

$$\delta' \vdash_D^\delta \langle \rangle : \text{Success}$$

$$\frac{\delta' \vdash_D^\delta s_1 : c_1 \quad \delta' \vdash_D^\delta s_2 : c_2 \quad (s_1, s_2) \rightsquigarrow s}{\delta' \vdash_D^\delta s : c_1 \parallel c_2}$$

- How about *Failure*?
- Note: Base language describing *events* is orthogonal to contract language. Plug in your own.



Contract semantics

- Denotational semantics that characterizes satisfaction:

$$C[[c]]^{\mathcal{D}[[D]]^{\delta; \delta \oplus \delta'}} = \{s \mid \delta' \vdash_D^\delta s : c\}.$$

- *Residuation*: Matching an event and representing the residual part as a bona-fide *contract expression*. Several possibilities:
 - *Deferred matching*: Deterministic, delays decision as to which particular commitment is matched; corresponds to trace-based semantics, with no internal choice.
 - *Eager matching*: Nondeterministic, picks particular commitment to match against; corresponds to internal choice of commitment followed by reduction
 - *Eager matching with routing*: Deterministic, instrumented version of eager matching that codes the internal choice as routing information and thus turns it into an external choice.
- Eager matching corresponds most closely to practice!



Contract Analysis

- Compositional and applicable not only to original contracts, but also to residual contracts!
- Wish to answer questions ranging from simple (what is my current todo list?) to complex (what is the current value of the contract based on my stochastic model?)
- Particularly important analysis: *Failure*, i.e. does the contract have no satisfying traces.

$$\frac{\forall \delta', \forall t' \geq t : (\delta \oplus \delta' \oplus T \mapsto t' \models \neg P)}{D, \delta, t \vdash \text{transmit}(XT \mid P). c \text{ failed}}$$



Why domain-specific languages?

- Domain-oriented concepts
 - Usable by BP specialists
- Orthogonal language concepts
 - Simple, yet expressive
 - Easy to analyze due to “simple” semantics and orthogonal design
 - Reusability (due to compositionality)
- No low-level details
 - Rapidly changed
- *Analyzability*
 - Programs as “intelligent data” – cannot only be “executed” but analyzed and transformed
 - Due to *restricted computed expressiveness* (the more expressive a language, the harder to analyze its programs)



Select references

- Andersen, Elsborg, Henglein, Simonsen, Stefansen, “Compositional specification of commercial contracts”, STTT 2006
- Peyton Jones, Eber, Seward, “Composing contracts: An adventure in financial engineering”, 2000
- Kristoffersen, Pedersen, Andersen, Runtime Verification of Timed LTL using Disjunctive Normalized Equation Systems, ITU, 2003



Other related Work

- **Concurrency models and languages**
 - Formal process calculi (CCS, pi, CSP, Bigraphs)
 - Process calculus-based languages (PiDuce)
 - Petri nets
- **BPEL – Business Process Execution Language**
- **YAWL – Yet Another Workflow Language**
Petri net-based workflow tool. Ostensibly covers all 20 control-flow patterns.
- **WS-CDL - Choreography Description Language**
In particular, see Kohei Honda's recent work on behavioral type systems.



Other related work...

- **Online Consultant, Resultmaker A/S:**
 - constraint-based workflow specification
 - Certain actions
 - declarative specifications of dependencies:
 - temporal (control)
 - used in hospital sector



Plans and more information

- **3gERP Project:** 4-year strategic research project with Microsoft Development Center Copenhagen, Copenhagen Business School on rapidly deployable universal ERP-system for SMEs; funded by Danish National Advanced Technology Foundation
- Plans/ongoing work related to contracts:
 - Contract residuation (monitoring) engine (prototype)
 - Contract-based GUI generation for role-based user interfaces
 - Logical concept modeling (specifically VAT)
 - Declarative specification of ex-post reports and real-time computation by finite differencing
 - Compositional contract analysis language
 - Stochastic contract analysis (valuation/pricing)
 - EDA architecture prototype for simple ERP system (primarily finance, though not debit-credit accounting based)
- More info: <http://www.3gERP.org>