Dynamic Process/Service Composition/Combination

Ugo Montanari
Dipartimento di Informatica
Università di Pisa

March 18, 2009

Work in collaboration with
Marzia Buscemi, IMT Lucca
Roadmap

- Global computing
- Pi-calculus
- Constraint semirings
- Cc-pi: Syntax
- Cc-pi: Reduction semantics
- Cc-pi: Examples
- Conclusion and future work
Roadmap

- Global computing
- Pi-calculus
- Constraint semirings
- Cc-pi: Syntax
- Cc-pi: Reduction semantics
- Cc-pi: Examples
- Conclusion and future work
Global Computing

- new models of computation
- new programming and analysis methods
  - distribution, concurrency
  - open endness
  - process mobility, service discovery
  - structuring into sessions, transactions
  - typing, code analysis, verification also at run time
- difficult to distinguish between design, execution and reconfiguration phases
- still distinction between procedural information and declarative information
Composition vs. Combination

- local computer system
  - sequential/parallel program composition
- wide area net
  - discovering and combining processes
- choreography, orchestration, coordination methods
- two-sided or multi-party sessions
- negotiations with non-functional service level agreements
- long transactions with failures and compensations
- architectural design languages for business-to-business, telecom or health applications.
Service-Oriented Computing

- distributed information systems + distributed concurrent programming
- accessing relevant information
  - about the network
  - about data and ontology of the application
- expressive contracts and service level agreements
- guarantees about security
- deadlock avoidance
- conformance of orchestration and choreography
- existence of compensations in the presence of failures
European Project SENSORIA, I

- linguistic primitives for modelling and programming
- qualitative and quantitative analysis methods
- sound engineering and deployment techniques

Some relevant studies (see abstract on the web for links)

**CaSPiS**
- two-sided sessions and pipelining, recursion
- handling (unexpected) termination of the partner's side of a session.
- session types guarantee communicating entities will not block
- session type inference is decidable
- implemented general tool
- MUSE multiparty sessions
European Project SENSORIA, II

- The process calculus Cc-Pi
  - name-passing calculi, concurrent constraint programming
  - requirements on service level agreements are constraints
  - soft notions of constraints

- Architectural Design Rewriting
  - software architectures development & reconfiguration with term-rewriting
  - proof that a design was constructed according to the style
  - naturally supports style-preserving reconfigurations
  - MAUDE implementation

- Lambda-req for security
  - selecting and invoking services
  - behavior of services over-approximated by a type and effect system
  - the approximation is model-checked
Roadmap

- Global computing
- Pi-calculus
- Constraint semirings
- Cc-pi: Syntax
- Cc-pi: Reduction semantics
- Cc-pi: Examples
- Conclusion and future work
About Names

Names can be:
- channels
- identifiers
- values (data)
- objects
- pointers
- references
- locations
- encryption keys
- ...

Names can:
- be created and destroyed
- sent them around to share information
- acquired to communicate with previously unknown processes
- used for evaluation or communication
- be tested to take decisions based on their values
- used as private means of communication, e.g. to share secret
- ...

We assume a countably infinite set of names $\mathcal{N}$ is defined.

\[
\begin{align*}
(\text{Processes}) \quad P & ::= \quad S & \text{sum} \\
& \quad | \quad P_1 | P_2 & \text{parallel composition} \\
& \quad | \quad (\nu x)P & \text{name restriction} \\
& \quad | \quad !P & \text{replication} \\

(Sums) \quad S & ::= \quad 0 & \text{inactive process (nil)} \\
& \quad | \quad \pi . P & \text{prefix} \\
& \quad | \quad S_1 + S_2 & \text{choice} \\

(\text{Prefixes}) \quad \pi & ::= \quad \overline{x}\langle y \rangle & \text{sends } y \text{ on } x \\
& \quad | \quad x(z) & \text{substitutes for } z \text{ the name received on } x \\
& \quad | \quad \tau & \text{internal action} \\
& \quad | \quad [x = y] \pi & \text{matching: tests equality of } x \text{ and } y
\end{align*}
\]
Structural Congruence

\[
\begin{align*}
P | \mathbf{0} & \equiv P & P_1 | P_2 & \equiv P_2 | P_1 & P_1 | (P_2 | P_3) & \equiv (P_1 | P_2) | P_3 \\
S + \mathbf{0} & \equiv S & S_1 + S_2 & \equiv S_2 + S_1 & S_1 + (S_2 + S_3) & \equiv (S_1 + S_2) + S_3 \\
!P & \equiv P | !P & [a = a]_\pi P & \equiv \pi.P \\
(\nu a)\mathbf{0} & \equiv \mathbf{0} & (\nu a)(\nu b)P & \equiv (\nu b)(\nu a)P \\
\end{align*}
\]

\[
\begin{align*}
P & \equiv P & P & \equiv Q & P & \equiv Q & Q & \equiv R & (\text{equivalence}) \\
& & & Q & \equiv P & & & & \\
& & & P & \equiv R & & & & \\
\end{align*}
\]

\[
\begin{align*}
P =_\alpha P' & \quad \frac{P \equiv P'}{C[P] \equiv C[P']} \quad (\text{congruence})
\end{align*}
\]
Reduction Rules

The so-called *reduction semantics* focuses on *internal* moves $P \xrightarrow{\cdot} Q$ only.

\[
\begin{align*}
(RTAU) & \quad (\tau.P + S) \xrightarrow{} P \\
(RCOM) & \quad (a(x).P_1 + S_1) | (\overline{a}(b).P_2 + S_2) \xrightarrow{} P_1[\overline{b/x}] | P_2 \\
(RPAR) & \quad P \xrightarrow{} P' \\
& \quad P \parallel Q \xrightarrow{} P' \parallel Q \\
(RRES) & \quad P \xrightarrow{} P' \\
& \quad (\nu a)P \xrightarrow{} (\nu a)P' \\
(RSTRUCT) & \quad P \equiv Q \quad Q \xrightarrow{} Q' \quad Q' \equiv P' \\
& \quad P \xrightarrow{} P'
\end{align*}
\]
Roadmap

- Global computing
- Pi-calculus
- Constraint semirings
- Cc-pi: Syntax
- Cc-pi: Reduction semantics
- Cc-pi: Examples
- Conclusion and future work
**Constraint Semirings**

**Definition**

A c-semiring is a tuple $\langle A, +, \times, 0, 1 \rangle$ s.t.:

- $A$ a set and $0, 1 \in A$
  - $1 \ (0)$ identity, absorbing on $x$, $+ \ (\times, x)$
- $+$ commutative, associative, idempotent ($a + b$ is the worst constraint that is best than $a$ and $b$)
- $\times$ associative, commutative, distributes over $+$ ($a \times b$ combines $a$ and $b$).

**Partial ordering $\leq$ on c-semirings**

$a \leq b$ iff $a + b = b$ (intuitively, $a$ is more constrained than $b$, alias $a \vdash b$).

**Examples**

- Classical CSPs: $\langle \{False, True\}, \vee, \wedge, False, True \rangle$
- Fuzzy CSPs: $\langle [0, 1], max, min, 0, 1 \rangle$
- Weighted CSPs: $\langle [0, \ldots, +\infty], min, +, +\infty, 0 \rangle$
Named Constraint Semirings

- A named c-semiring is a c-semiring equipped with:
  - name fusions \( x = y \) for all names \( x, y \)
  - a notion of support \( \text{supp}(c) \) for each element \( c \)
  - a hiding operator \( (\forall x.\) that makes \( x \) local in \( c \)
  - a set of axioms (ruling how to combine operations)

- A named constraint is just an element of the named c-semiring.

Example: functional constraints

- Let \( D \) be a domain for \( \mathcal{N} \), a functional constraint is a function \( c = (\mathcal{N} \rightarrow D) \rightarrow \{\text{True}, \text{False}\} \) (es. \( x\eta = a, y\eta = b \))

- A named c-semiring for functional constraints is such that:
  - the elements are all functional constraints over \( \mathcal{N} \) and \( D \)
  - \( (c + d)\eta = c\eta \lor d\eta \) and \( (c \times d)\eta = c\eta \land d\eta \)
  - \( 0\eta = \text{False} \) and \( 1\eta = \text{True} \)
  - \( (\forall x.\ c) \) and \( p\ c \) are as expected
Roadmap

- Global computing
- Pi-calculus
- Constraint semirings
- Cc-pi: Syntax
- Cc-pi: Reduction semantics
- Cc-pi: Examples
- Conclusion and future work
Aims I

1. Providing a formal model for defining SLA contracts and for validating contracts at service execution.
3. Studying mechanisms for resource allocation and for combining different SLA requirements.
Main Ingredients

The CC-Pi calculus is simple process calculus that:

- extends $\pi_F$ by generalising explicit fusions to named constraints
- integrates cc-programming primitives ($\text{ask, tell}$)
- introduces new primitives for constraint handling ($\text{retract, check}$)

SLA Contract Scenario

- A server and client willing to reach an agreement are specified as cc-pi processes that add their own requirements and guarantees as constraints to (possibly, local) stores.
- The synchronisation of two processes results in the combination of their respective stores of constraints and may succeed or be stuck.
CcPi-Calculus (syntax)

- Cc-pi is parametric wrt named c-semirings (assume c ranges over constraints of an arbitrary named c-semiring)
- x, y, z, ... range over N; K ranges over a set of process identifiers.

**Prefixes**

\[ \pi ::= \tau \mid \overline{x}(\bar{y}) \mid x(\bar{y}) \mid \text{tell} \ c \mid \text{ask} \ c \mid \text{retract} \ c \mid \text{check} \ c \]

**Unconstrained Proc.**

\[ U ::= 0 \mid U \mid U \mid \sum_i \pi_i . U_i \mid (x) U \mid K(\bar{y}) \]

**Constrained Proc.**

\[ P ::= U \mid c \mid P \mid P \mid (x) P \]
Roadmap

- Global computing
- Pi-calculus
- Constraint semirings
- Cc-pi: Syntax
- Cc-pi: Reduction semantics
- Cc-pi: Examples
- Conclusion and future work
The structural axioms allow to put processes into a normal form

\[(x_1) \ldots (x_n)(C | U)\]

with \(C\) a parallel composition of constraints and \(U\) an unconstrained process.

**SOS rules**

(TAU) \(C | \tau.U \rightarrow C | U\)

(TELL) \(C | \text{tell } d.U \rightarrow C | d | U\) if \(C | d\) consistent

(ASK) \(C | \text{ask } d.U \rightarrow C | U\) if \(C \vdash d\) (RETRACT) \(C | \text{retract } d.U \rightarrow (C - d) | U\)

(CHECK) \(C | \text{check } d.U \rightarrow C | U\) if \(C | d\) consistent

(COM) \(C | (\overline{x} \langle \overline{y} \rangle.U + \sum \pi_i.U_i) \langle z \overline{w} \rangle.V + \sum \pi'_j.V_j) \rightarrow (C | \overline{y} = \overline{w}) | U | V\)

if \(\overline{y} = \overline{w}, C | \overline{y} = \overline{w}\) consistent and \(C \vdash x = z\)

(SUM) \(\frac{C | \pi_i.U_i \rightarrow P}{C | \sum \pi_i.U_i \rightarrow P}\)

(PAR) \(P \rightarrow P'\) \(\frac{P | U \rightarrow P' | U}{P \rightarrow P'}\)

(RES) \(P \rightarrow P'\) \(\frac{(x)P \rightarrow (x)P'}{(x)P \rightarrow (x)P'}\)

(STRUCT) \(P \equiv P'\) \(P' \rightarrow Q'\) \(Q' \equiv Q\) \(\frac{P \rightarrow Q}{P \rightarrow Q}\)
Roadmap

- Global computing
- Pi-calculus
- Constraint semirings
- Cc-pi: Syntax
- Cc-pi: Reduction semantics
- Cc-pi: Examples
- Conclusion and future work
Example 1

Consider a service offering computing resources (e.g. units of CPUs)
The provider $P$ and a client $C$ want to conclude a SLA contract.
$P_N$ (N available resources) and $C_n$ (at least n resources) are as below

$$P_N = (x_0)(\text{tell } (x_0 = N).Q(x_0))$$
$$Q(x) = (v)(x')(\text{tell } (x' = x - v).\text{tell } (v \leq \text{max}).c\langle v \rangle.Q(x')).$$
$$C_n = (y)(\text{tell } (y \geq n).\overline{c}\langle y \rangle.\tau.\text{retract } (y \geq n).\text{tell } (y = 0)).$$
CcPi-Calculus (example II)

Example 2
A slightly more complex scenario with one provider \( P_N \) and three clients \( C_{n_1}, C_{n_2}, \) and \( C_{n_3} \).

\[
\begin{align*}
x_0 &\equiv N \\
x_1 &= x_0 - v_1 \\
x_2 &= x_1 - v_2 \\
x_3 &= x_2 - v_3 \\
v_1 &\leq \max \quad v_2 &\leq \max \\
v_3 &\leq \max
\end{align*}
\]

Provider \( P_N \):
- \( x_0 \equiv N \)
- \( x_1 = x_0 - v_1 \)
- \( x_2 = x_1 - v_2 \)
- \( x_3 = x_2 - v_3 \)

Client \( C_{n_1} \):
- \( v_1 = y_1 \)
- \( y_1 \geq n_1 \)

Client \( C_{n_2} \):
- \( v_2 = y_2 \)
- \( y_2 \geq n_2 \)

Client \( C_{n_3} \):
- \( v_3 = y_3 \)
- \( y_3 \geq n_3 \)
A CallBySms Service Scenario, I

3rd Party Application

Network Operator Domain

- Service subscription
- Service execution
A CallBySms Service Scenario, II

1. The Third Party application subscribes the services that are used by the CallBySms service and signs a SLA contract with the Network Operator;
2. The CallBySMS service is activated and the Third Party application receives a service number, e.g. 11111;
3. Mary sends an SMS “REGISTER sunshine” to the service number 11111;
4. The service associates “sunshine” to the opaque-id of Mary;
5. John sends an SMS “CALL sunshine” to the service number 11111;
6. The service retrieves the opaque-id associated to “sunshine” and set-up a call;
7. John’s phone rings; John answers and gets the ringing tone;
8. Mary’s phone rings; Mary answers;
9. John and Mary are connected.
CallBySms Specification in cc-pi

Policies
\[ c_{\text{time}} = (7am \leq i \leq 9am) \times (5pm \leq f \leq 9pm) \]
\[ c_{\text{req}} = nc \leq \text{max\_call} \]
\[ d_{\text{time}} = (6am \leq i' \leq 8am) \times (4pm \leq f' \leq 6pm) \]
\[ d_{\text{req}} = (ncp' \leq \text{call\_per\_pers}) \times (nr' \leq nc'/\text{call\_per\_pers}) \]

3rdPA-ParX Negotiation
\[ \text{ParX\_Neg} = (i, f, nc, beg, end) (\text{tell } c_{\text{time}} \times c_{\text{req}}, x\langle i, f, nc, beg, end \rangle.0) \]
\[ \text{3rdPA\_Neg} = (i', f', ncp', nc', nr', beg', end') (\text{tell } d_{\text{time}} \times d_{\text{req}}, x\langle i', f', nc', beg', end' \rangle.0) \]

Clock
\[ \text{Clock} = (t_0) (\text{tell } t = t_0, \text{Cl}(t, t_0)) \]
\[ \text{Cl}(t, t') = \text{retract } t = t'. \text{tell } t = t' + 1. \text{Cl}(t, t' + 1) \]

Service Execution
\[ \text{ParX\_Ex} = \text{check } (t = i). \text{beg}\langle i \rangle. \text{ParX\_Acpt\_Reqst}. \text{check } (t = f). \text{end}\langle f \rangle.0 \]
\[ \text{3rdPA\_Ex} = \text{beg}\langle f \rangle.3\text{rdPA\_Acpt\_Reqst}. \text{end}\langle t' \rangle.0 \]

Handling registration requests
\[ \text{Regist\_User} = (\text{mary}) (\overline{(\text{mary, sunshine})}. \text{mary}\langle i \rangle). \text{Wait\_Calls} \]
\[ \text{ParX\_Acpt\_Reqst} = (id, nn, ch) (\overline{(id, nn, ch)}). \text{tell } (nr' \leq \text{max\_call/call\_per\_pers}). \text{tell } (nr' = nr' + 1). x\langle nn', ch' \rangle. (\text{3rdPA\_Acpt\_Reqst} | \text{3rdPA\_Acpt\_Call}) \]

Handling call requests
\[ \text{Wait\_Calls} = (\text{call}) (\text{mary} (\text{call}'). \text{call}\langle i \rangle). \text{Wait\_Calls} \]
\[ \text{Caller} = (\text{john}) \text{sunshine} (\text{john}). \text{tell } (nc = nc + 1). x\langle \text{call}, ch' \rangle. (\text{tell } (ncp' \leq \text{call\_per\_pers}). \text{tell } (ncp' = ncp' + 1). ch'\langle j \rangle. \text{3rdPA\_Acpt\_Call}) \]

System
\[ S = (t, x, z)(\text{3rdPA\_Neg} | \text{ParX\_Neg} | \text{3rdPA\_Ex} | \text{ParX\_Ex} | \text{Caller} | \text{Regist\_User} | \text{Clock}) \]
Roadmap

- Global computing
- Pi-calculus
- Constraint semirings
- Cc-pi: Syntax
- Cc-pi: Reduction semantics
- Cc-pi: Examples
- Conclusion and future work
Conclusion and Future Work

- Cc-Pi part of EU FET GC2 project Sensoria
- Reduction semantics at ESOP 2007 and symbolic semantics at ESOP 2008
- Names as keys for secure retract
- Efficient evaluation of constraints via locality restrictions and dynamic programming
- Extension to include behavioral types?
- Extension to handle assume - guarantee?
- Extension to handle ontologies?