Cc-Pi: A Constraint-Based Language for Contracts with Service Level Agreements

Ugo Montanari
Dipartimento di Informatica
Università di Pisa

November 27, 2008

Work in collaboration with
Marzia Buscemi, IMT Lucca
Roadmap

- Service Level Agreements (SLA)
- Constraint semirings
- Syntax
- Reduction semantics
- Examples
- Open semantics
- Mapping Pi-F calculus on Cc-Pi
- Conclusion and future work
Roadmap

- Service Level Agreements (SLA)
- Constraint semirings
- Syntax
- Reduction semantics
- Examples
- Open semantics
- Mapping Pi-F calculus on Cc-Pi
- Conclusion and future work
Service Level Agreements

SLA contracts
- SLA is a key issue in the SOC paradigm.
- SLAs allow a client and a service provider to define a contract with emphasis on QoS (cost, performance, etc.).

XML-based solutions
- Several technologies based on XML for defining SLAs are currently available (WS-Agreement, WSLA, SLAng).
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 PiF by generalising explicit fusions to named constraints
- integrates cc-programming primitives (ask, tell)
- introduces new primitives for constraint handling (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.
Roadmap

- Service Level Agreements (SLA)
- Constraint semirings
- Syntax
- Reduction semantics
- Examples
- Open semantics
- Mapping Pi-F calculus on Cc-Pi
- 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 \(0, 1 \in A\)
- \(+\) 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: \(\langle\{\text{False}, \text{True}\}, \lor, \land, \text{False}, \text{True}\rangle\)
- 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 $(\nu x.)$ 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 $c = (\mathcal{N} \to D) \to \{\text{True}, \text{False}\}$ (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 $1\eta = \text{True}$
  - $(\nu x. c)$ and $\rho c$ are as expected
Roadmap

- Service Level Agreements (SLA)
- Constraint semirings
- Syntax
- Reduction semantics
- Examples
- Open semantics
- Mapping Pi-F calculus on Cc-Pi
- Conclusion and future work
Cc-pi is parametric wrt named c-semirings (assume c ranges over constraints of an arbitrary named c-semiring)

\( x, y, z, \ldots \) range over \( \mathcal{N} \); \( K \) ranges over a set of process identifiers.

**Prefixes**

\[
\pi \ ::= \ \tau \mid \bar{x}\langle\bar{y}\rangle \mid x\langle\bar{y}\rangle \mid \text{tell} \ c \mid \text{ask} \ c \mid \text{retract} \ c \mid \text{check} \ c
\]

**Unconstrained Proc.**

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

**Constrained Proc.**

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

- Service Level Agreements (SLA)
- Constraint semirings
- Syntax
- Reduction semantics
- Examples
- Open semantics
- Mapping Pi-F calculus on Cc-Pi
- Conclusion and future work
CcPi-Calculus (semantics)

The structural axioms allow to put processes into a normal form

\[(x_1) \ldots (x_n) \langle C \mid U \rangle\]

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

**SOS rules**

\[
\begin{align*}
\text{TAU} & : C \tau U \to C U \\
\text{TELL} & : C \text{tell } d U \to C d U \text{ if } C d \text{ consistent} \\
\text{ASK} & : C \text{ask } d U \to C U \text{ if } C \vdash d \text{ (RETRACT) } C \text{retract } d U \to (C - d) U \\
\text{CHECK} & : C \text{check } d U \to C U \text{ if } C d \text{ consistent} \\
\text{COM} & : C \langle \pi_i U_i + \sum \pi_j V_j \rangle \to (C \langle \bar{y} \rangle U + \sum \pi_j V_j) \to (C \langle \bar{y} = \bar{w} \rangle U V \text{ if } \langle \bar{y} \rangle = \langle \bar{w} \rangle, C \langle \bar{y} = \bar{w} \rangle \text{ consistent and } C \vdash x = z \\
\text{SUM} & : C \pi_i U_i \to P \to C \sum \pi_i U_i \to P \\
\text{PAR} & : P \to P' \to P U \to P' U \\
\text{RES } & : P \to P' \to (x) P \to (x) P' \\
\text{STRUCT} & : P \equiv P' \to P' \to Q' \to Q' \equiv Q \to P \to Q
\end{align*}
\]
Roadmap

- Service Level Agreements (SLA)
- Constraint semirings
- Syntax
- Reduction semantics
- Examples
- Open semantics
- Mapping Pi-F calculus on Cc-Pi
- 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

\[
P_N = (x_0)(\text{tell } (x_0 = N).Q(x_0))
\]
\[
Q(x) = (v)(x')(\text{tell } (x' = x - v).\text{tell } (v \leq \text{max}).c(v).Q(x')).
\]
\[
C_n = (y)(\text{tell } (y \geq n).\overline{c}(y).\tau.\text{retract } (y \geq n).\text{tell } (y = 0)).
\]

![Diagram of provider $P_N$ and client $C_n$.]
Example 2

A slightly more complex scenario with one provider $P_N$ and three clients $C_{n_1}$, $C_{n_2}$, and $C_{n_3}$.
A CallBySms Service Scenario, I

3rd Party Application

Network Operator Domain

Parlay X ShortMessaging

Parlay X ThirdPartyCall

Mary

CALL sunshine

John

REGISTER sunshine

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

\[
c_{\text{time}} = (7\text{am} \leq t \leq 9\text{am}) \times (5\text{pm} \leq f \leq 9\text{pm})
\]

\[
c_{\text{freq}} = n\leq \text{max.call}
\]

\[
d_{\text{time}} = (6\text{am} \leq t' \leq 8\text{am}) \times (4\text{pm} \leq f' \leq 6\text{pm})
\]

\[
d_{\text{freq}} = (ncp' \leq \text{call.per.pers}) \times (nr' \leq ncp'/\text{call.per.pers})
\]

3rdPA-PARX NEGOTIATION

\[
\text{ParX.Neg} = (i, f, nc, beg, end) \langle \text{tell } c_{\text{time}} \times c_{\text{freq}}, x(i, f, nc, beg, end) \rangle.0
\]

\[
\text{3rdPA.Neg} = (i', f', ncp', nc', nr', beg, end') \langle \text{tell } d_{\text{time}} \times d_{\text{freq}}, \overline{x}(i', f', nc', beg', end') \rangle.0
\]

CLOCK

\[
\text{Clock} = (t_0) \langle \text{tell } t = t_0.\text{Cl}(t, t_0) \rangle
\]

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

SERVICE EXECUTION

\[
\text{ParX.Ex} = \text{check } (t = i).beg().\text{ParX.Acpt.Reqst.check } (t = f).end().0
\]

\[
\text{3rdPA.Ex} = \overline{beg}().3\text{rdPA.Acpt.Reqst.end}().0
\]

HANDLING REGISTRATION REQUESTS

\[
\text{Regist.User} = (\text{mary})(\overline{x}(\text{mary}, \text{sunshine}).\text{mary}).\text{Wait Calls}
\]

\[
\text{ParX.Acpt.Reqst} = (id, nn, ch)(\overline{x}(id, nn).\overline{x}(nn, ch).\overline{id}).(\text{ParX.Acpt.Reqst} | \text{ParX.Acpt.Call})
\]

\[
\text{3rdPA.Acpt.Reqst} = (nn', ncp, ch') (\text{check } (nCP' \leq \text{max.call/\text{call.per.pers}}).\text{tell } (nCP' = nCP' + 1).x(nn', ch')).(3\text{rdPA.Acpt.Reqst} | 3\text{rdPA.Acpt.Call})
\]

HANDLING CALL REQUESTS

\[
\text{Wait.Call} = (\text{call})(\text{mary}(\text{call}).\text{call'}).\text{Wait Calls}
\]

\[
\text{Caller} = (\text{john}).\text{sunshine}(\text{john}).\text{john}.0
\]

\[
\text{ParX.Acpt.Call} = (\text{call})(\text{check } (\text{nc} \leq \text{max.call}).\text{tell } (\text{nc} = \text{nc} + 1).\text{nn}(\text{call}).\overline{ch}().\overline{id}(\text{john}).\text{ParX.Acpt.Call})
\]

\[
\text{3rdPA.Acpt.Call} = (\text{check } (ncp' \leq \text{call.per.pers}).\text{tell } (ncp' = ncp' + 1).ch').3\text{rdPA.Acpt.Call}
\]

SYSTEM

\[
S = (t, x, z)(3\text{rdPA.Neg} | \text{ParX.Neg} | 3\text{rdPA.Ex} | \text{ParX.Ex} | \text{Caller} | \text{Regist.User} | \text{Clock})
\]
Roadmap

- Service Level Agreements (SLA)
- Constraint semirings
- Syntax
- Reduction semantics
- Examples
- Open semantics
- Mapping Pi-F calculus on Cc-Pi
- Conclusion and future work
Labelled Transition System

(PREF) \[ c \langle \pi.U \rangle \xrightarrow{\pi} c \langle U \rangle \quad \pi = \tau, \bar{x}\langle y \rangle, x\langle y \rangle \]

(TELL) \[ c \langle \text{tell } d.U \rangle \xrightarrow{\tau} c \langle d \cup U \rangle \text{ if } c \otimes d \neq 0 \]

(ASK) \[ c \langle \text{ask } d.U \rangle \xrightarrow{\tau} c \langle U \rangle \text{ if } c \leq d \]

(SUM) \[ c \langle \sum \pi_i.U_i \xrightarrow{\alpha} U' \rangle \]

(PAR) \[ P \xrightarrow{\alpha} P' \quad \text{bn}(\alpha) \cap \text{fn}(U) = \emptyset \]

(COMM) \[ c \langle U \xrightarrow{x\langle y \rangle} U' \rangle \quad c \langle V \xrightarrow{\bar{z}\langle w \rangle} V' \rangle \quad c \otimes (y = w) \neq 0 \quad c \leq x = z \]

(RES) \[ P \xrightarrow{\alpha} P' \quad x \notin \text{n}(\alpha) \]

(OPEN) \[ P \xrightarrow{z\langle x \rangle} P' \quad \text{store}(P) \not\subseteq x = z \]

(STRUCT) \[ P \equiv P' \quad P' \xrightarrow{\alpha} Q' \quad Q' \equiv Q \]

\[ P \xrightarrow{\alpha} Q \]
Open Bisimilarity

\[
\begin{align*}
\text{store}(c) &= c \\
\text{store}(P|Q) &= \text{store}(P) \otimes \text{store}(Q) \\
\text{store}(U) &= 1 \\
\text{store}((x)P) &= \forall x. \text{store}(P)
\end{align*}
\]

**Definition 1 (open bisimilarity).** Open bisimilarity \( (~^0) \) is the largest symmetric relation \( S \) between processes such that \( PSQ \) implies:

1. \( \text{store}(P) = \text{store}(Q) \);
2. If \( P \xrightarrow{\alpha} P' \) with \( \text{bn}(\alpha) \cap \text{fn}(Q) = \emptyset \) then \( Q \xrightarrow{\beta} Q' \) and \( P' S Q' \), for some \( Q' \) and \( \beta \) such that \( \text{store}(P) \preceq \alpha = \beta \);
3. \( c | P S c | Q \), for all constraints \( c \neq 0 \).
Symbolic Semantics

\[
a \div b = \max \{ x \in A \mid b \otimes x \preceq a \}
\]
Symbolic Bisimilarity

**Definition 2 (symbolic bisimilarity).** Symbolic (open) bisimilarity ($\sim^s$) is the largest symmetric relation $S$ between processes such that $PSQ$ implies:

1. $\text{store}(P) = \text{store}(Q)$;
2. If $P \xrightarrow[\alpha]{\alpha} P'$ with $\alpha \neq \tau$ and $\text{bn}(\alpha) \cap \text{fn}(Q) = \emptyset$ then $Q \xrightarrow[\beta]{\beta} Q'$ and $P' S Q'$, for some $Q'$ and $\beta$ such that $\text{store}(P) \leq \alpha = \beta$;
3. If $P \xrightarrow[c]{c} P'$ then $Q \xrightarrow[d]{d} Q'$ and $c \mid P' S c \mid Q'$, for some $Q'$ and $d$ such that $c \leq d$.  

Main Result

**Theorem 2.** Symbolic bisimilarity $\sim^s$ and open bisimilarity $\sim^o$ coincide.
Roadmap

- Service Level Agreements (SLA)
- Constraint semirings
- Syntax
- Reduction semantics
- Examples
- Open semantics
- Mapping Pi-F calculus on Cc-Pi
- Conclusion and future work
\[ \pi ::= \tau \mid \bar{x}(\bar{y}) \mid x(\bar{y}) \quad U ::= U \mid U U \mid \pi.U \mid (x)U \mid I(\bar{y}) \quad P ::= U \mid \bar{x} = \bar{y} \mid P P \mid (x)P \]

\[
\begin{align*}
P \mid 0 & \equiv_F P \\
P \mid Q & \equiv_F Q \mid P \\
(P \mid Q) \mid R & \equiv_F (P \mid (Q \mid R)) \\
(x)(y)P & \equiv_F (y)(x)P \\
P \mid (x)Q & \equiv_F (x)(P \mid Q) \quad \text{if } x \notin \text{fn}(P) \\
I(\bar{y}) & \equiv_F [\bar{y}/\bar{x}]U \quad \text{if } I(\bar{x}) \overset{\text{def}}{=} U
\end{align*}
\]

\[E^{\text{eq}}(0) = \text{Id} \quad E^{\text{eq}}(x = y) = \{ (x, y), (y, x) \} \cup \text{Id} \quad E^{\text{eq}}(\pi.U) = \text{Id} \]

\[E^{\text{eq}}(P \mid Q) = E^{\text{eq}}(P) \cup E^{\text{eq}}(Q) \quad E^{\text{eq}}((x)P) = E^{\text{eq}}(P) \setminus x \quad E^{\text{eq}}(I(\bar{y})) = E^{\text{eq}}([\bar{y}/\bar{x}]U) \quad \text{if } I(\bar{x}) \overset{\text{def}}{=} U\]

\[\begin{align*}
\text{(PREF)} & \quad \pi.U & \rightarrow^F & \pi.U \\
\text{(COMM)} & \quad P \xrightarrow{\tau} P' Q \xrightarrow{\tau} Q' & \rightarrow^F & P \xrightarrow{\mu} P' \quad \text{bn}(\mu) \cap \text{fn}(Q) = 0 \\
\text{(PAR)} & \quad P \xrightarrow{\mu_f} P' \quad \text{bn}(\mu) \cap \text{fn}(Q) = 0 & \rightarrow^F & P \xrightarrow{\mu_f} P' \mid Q \\
\text{(OPEN)} & \quad P \xrightarrow{(\bar{w})\bar{y}} P' & \rightarrow^F & x = z \notin E^{\text{eq}}(P), x \in \bar{y} \setminus \bar{w} \quad (\ast) \\
\text{(STRUCT)} & \quad (x)P \xrightarrow{\mu_f} (x)P' & \rightarrow^F & P \equiv P' \xrightarrow{\mu_f} Q' \equiv Q \quad P \xrightarrow{\mu_f} Q
\end{align*}\]

\(\ast\): Plus the analogous rule for output prefixes
Bisimilarity

Definition 3 (inside-outside bisimilarity). Inside-outside bisimilarity ($\sim_{io}$) is the largest symmetric relation $S$ between Pi-F processes such that $PSQ$ implies:

- $\text{Eq}(P) = \text{Eq}(Q)$;
- If $P \xrightarrow{\mu} P'$ with $\text{bn}(\mu) \cap \text{fn}(Q) = 0$ then $Q \xrightarrow{\mu} Q'$ and $P' S Q'$;
- $P \mid x = y \ S \ Q \mid x = y$, for all fusions $x = y$. 


Mapping Pi-F Calculus into Cc-Pi Calculus

\[
\begin{align*}
[\tau.U] &= \tau.[U] \\
[\bar{x}(\bar{y}).U] &= (z)(\bar{x}(z).[U] | z = \bar{y}) \\
[x(\bar{y}).U] &= (z)(x(z).[U] | z = \bar{y}) \\
[0] &= 0 \\
[P | Q] &= [P] | [Q] \\
[(x)P] &= (x)[P] \\
[I(\bar{y})] &= [[\bar{y}/\bar{x}]U] \\
\text{if } I(\bar{x}) &\overset{\text{def}}{=} U
\end{align*}
\]

\[P \sim^o Q \iff [P] \sim^{io} [Q] \]
Roadmap

- Service Level Agreements (SLA)
- Constraint semirings
- Syntax
- Reduction semantics
- Examples
- Open semantics
- Mapping Pi-F calculus on Cc-Pi
- 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
- Connection with work by Bonchi and Montanari about symbolic semantics via a normalization functor