A Basic Calculus for Modeling Service Level Agreement

Rocco De Nicola
{denicola,pugliese}@dsi.unifi.it
Dipartimento di Sistemi e Informatica, Università di Firenze,

Rosario Pugliese

Gianluigi Ferrari  Ugo Montanari  Emilio Tuosto
{giangi,ugo,etuosto}@di.unipi.it
Dipartimento di Informatica, Università di Pisa

COORDINATION05: Namur, 20 – 23 April 2005
Overview of the talk

“Ayudadme a comprender lo que os digo y os lo explicaré mejor”
“Help me in understanding what I’m saying and I will explain it better”

(Antonio Machado)
Overview

- A few motivations
- Background: constraint semirings (c-semirings)
- \( KoS \)
  - design choices...
  - ...weighted links
- Syntax and semantics of \( KoS \)
- Examples
- Conclusion
Motivations

The real technology - behind all of our technologies - is language

(N. Fisher)
Service Oriented Computing

- applications are made by gluing services
- “autonomous”
- independent (local choices, independently built)
- mobile/stationary
- “interconnected”
- interactions governed by programmable coordination policies
- services are searched and binded ... offline
Service Oriented Computing

- applications are made by gluing *services*
  - “autonomous”
  - independent (local choices, independently built)
  - mobile/stationary
  - “interconnected”
- interactions governed by programmable coordination policies
- services are searched and binded ... offline

Can search/bind be dynamic and at run-time?
Service Oriented Computing

- applications are made by gluing *services*
- “autonomous”
- independent (local choices, independently built)
- mobile/stationary
- “interconnected”
- interactions governed by programmable coordination policies
- services are searched and binded ... offline

**Can search/bind be dynamic and at run-time?**

Search and bind wrt application level QoS

- not low-level performance (e.g., throughput, response time)
- but application-related, e.g.
  - price services
  - payment mode
  - data available in a given format
Our approach in brief

WAN programming is not just $\text{go}(P), \bar{s}\langle x \rangle$ or $s(y)$

- Lifting QoS issues to application level...
- ...for programming global computers
- with programmable application level QoS
- and develop proof techniques and tools

We are currently distilling Klaim into $\mathcal{KoS}$ which exploits $c$-semiring for
- expressing application level QoS dependent connections and
- for coordinating remote activities...
- ...by means of $c$-semiring values

First steps (extending Klaim)
in [DFM+03]
Our approach in brief

WAN programming is not just \( \text{go}(P), \bar{s}(x) \) or \( s(y) \)

- Lifting QoS issues to application level...
- ...for programming global computers
- with programmable application level QoS
- and develop proof techniques and tools

First steps (extending Klaim) in [DFM+03]

We are currently distilling Klaim into \( \mathcal{KoS} \) which exploits c-semiring for

- expressing application level QoS dependent connections and
- for coordinating remote activities...
- ...by means of c-semiring values

We also are defining a graph-based model [HT05]
Our approach in brief

WAN programming is not just $\text{go}(P), \bar{s}(x)$ or $s(y)$

- Lifting QoS issues to application level...
- ...for programming global computers
- with programmable application level QoS
- and develop proof techniques and tools

We are currently distilling Klaim into $\mathcal{KoS}$ which exploits $\text{c-semiring}$ for
- expressing application level QoS dependent connections and
- for coordinating remote activities...
- ...by means of $\text{c-semiring}$ values

We also are defining a graph-based model [HT05]

First steps (extending Klaim) in [DFM+03]

Ask me at the poster session...look for or...
Background

“During my nine years at the elementary schools I was not able to teach anything to my professors”

(Bertolt Brecht)
C-Semirings \([BMR95, BMR97]\) for abstracting application level QoS

\[ \langle A, +, \star, 0, 1 \rangle, \text{ where} \]

- \(A\) is a set (containing 0 and 1),
- \(+\), \(\star\): \(A \times A \to A\)

<table>
<thead>
<tr>
<th>+</th>
<th>\star</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x + y = y + x)</td>
<td>(x \star y = y \star x)</td>
</tr>
<tr>
<td>((x + y) + z = x + (y + z))</td>
<td>((x \star y) \star z = x \star (y \star z))</td>
</tr>
<tr>
<td>(x + 0 = x)</td>
<td>(x \star 0 = 0)</td>
</tr>
<tr>
<td>(x + 1 = 1)</td>
<td>(x \star 1 = x)</td>
</tr>
<tr>
<td>(x + x = x)</td>
<td>((x + y) \star z = (x \star z) + (y \star z))</td>
</tr>
</tbody>
</table>
**Constraint Semirings**

C-Semirings \([\text{BMR95, BMR97}]\) for abstracting application level QoS

- \(\langle A, +, \star, 0, 1 \rangle\), where
- \(A\) is a set (containing 0 and 1),
- \(+, \star : A \times A \rightarrow A\)

<table>
<thead>
<tr>
<th>( + )</th>
<th>( \star )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x + y = y + x )</td>
<td>( x \star y = y \star x )</td>
</tr>
<tr>
<td>( (x + y) + z = x + (y + z) )</td>
<td>( (x \star y) \star z = x \star (y \star z) )</td>
</tr>
<tr>
<td>( x + 0 = x )</td>
<td>( x \star 0 = 0 )</td>
</tr>
<tr>
<td>( x + 1 = 1 )</td>
<td>( x \star 1 = x )</td>
</tr>
<tr>
<td>( x + x = x )</td>
<td>( (x + y) \star z = (x \star z) + (y \star z) )</td>
</tr>
</tbody>
</table>
**Constraint Semirings**

C-Semirings [BMR95, BMR97] for abstracting application level QoS

- \( \langle A, +, \ast, 0, 1 \rangle \), where
  - \( A \) is a set (containing 0 and 1),
  - \(+, \ast : A \times A \rightarrow A\)

<table>
<thead>
<tr>
<th>+</th>
<th>\ast</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x + y = y + x )</td>
<td>( x \ast y = y \ast x )</td>
</tr>
<tr>
<td>( (x + y) + z = x + (y + z) )</td>
<td>( (x \ast y) \ast z = x \ast (y \ast z) )</td>
</tr>
<tr>
<td>( x + 0 = x )</td>
<td>( x \ast 0 = 0 )</td>
</tr>
<tr>
<td>( x + 1 = 1 )</td>
<td>( x \ast 1 = x )</td>
</tr>
<tr>
<td>( x + x = x )</td>
<td>( (x + y) \ast z = (x \ast z) + (y \ast z) )</td>
</tr>
</tbody>
</table>

- Implicit partial order: \( a \leq b \iff a + b = b \) “\( b \) is better than \( a \)”
A bunch of c-semirings

C-semirings structures can be defined for many frameworks:

- $\langle \{\text{true, false}\}, \lor, \land, \text{false}, \text{true} \rangle$ (boolean): Availability
- $\langle \text{Real}+, \text{min}, +, +\infty, 0 \rangle$ (optimization): Price, propagation delay
- $\langle \text{Real}+, \text{max}, \text{min}, 0, +\infty \rangle$ (max/min): Bandwidth
- $\langle [0, 1], \text{max}, \cdot, 0, 1 \rangle$ (probabilistic): Performance and rates
- $\langle [0, 1], \text{max}, \text{min}, 0, 1 \rangle$ (fuzzy): Performance and rates
- $\langle 2^N, \cup, \cap, \emptyset, N \rangle$ (set-based, where $N$ is a set): Capabilities

**Proposition 1** The cartesian product of c-semirings is a c-semiring.
Oath of Hippocrates
**KoS characteristics**

*KoS* aims at being a minimal calculus for SOC

- *KoS* builds on *Klaim* (e.g., processes are localised)
- ’cause it naturally supports a peer-to-peer programming model
- *KoS* primitives handle QoS values as first class entities
- *KoS* semantics ensures that the QoS values are respected during execution

- only local communications (unlike *Klaim*)
- link construction primitives
- links are “mono-use”
- only one remote action
- which relies on link topology
- semantic transitions report the “cost” of the execution
KoS ... graphically
Migration in $KoS$

$KoS$ ... graphically
Migration in KoS

KoS ... graphically
Migration in KoS

... graphically

KoS
KoS ... graphically
Migration in \( KoS \)

\( KoS \) ... graphically
A motivating example

Consider a scenario where \( n \) servers provide services to \( m \) clients and focus on balancing the load of the servers.

- clients \( (c_i) \) and servers \( (s_j) \) are located on different nodes
- \( c_i \) issues requests to \( s_j \) by spawning a process \( R \)

A generic client is described by the following term:

\[
c_i :: \langle s_1, \kappa_1 \rangle \mid \ldots \mid \langle s_n, \kappa_n \rangle \mid !C_\delta
\]

- \( \langle s_j, \kappa_j \rangle \) represents the load \( \kappa_j \) of the server \( s_j \) perceived by \( c_i \)
- \( C_\delta \) and \( R \) specify the behaviour of \( c_i \):

\[
C_\delta \triangleq (\?u, \?v).\varepsilon_v[R]@u.\text{con}_{v*\delta}(u).\langle u, v * \delta \rangle
\]

\[
R \triangleq (\?x).\langle x + 1 \rangle \ldots \text{actual request} \ldots (\?y).\langle y - 1 \rangle
\]

**Remark 1** Remote spawning consumes the traversed links, hence \( c_i \) attempts to re-establish a connection with the server!
A motivating example

$s_j$ is described as:

\[ s_j :: \langle h \rangle \mid \langle c_1, \kappa'_1 \rangle \mid \ldots \mid \langle c_m, \kappa'_m \rangle \mid \text{!(}S\ c_1\ s_j\text{)} \mid \ldots \mid \text{!(}S\ c_m\ s_j\text{)} \]

- \( \langle c_i, \kappa'_i \rangle \) records the QoS value \( \kappa'_i \) assigned to the link towards \( c_i \)
- \( \langle h \rangle \) is the current load of the \( s_j \)
- \( S\ c_i\ s_j \) is a \textit{load manager} for \( c_i \)

\[ S\ c\ s \triangleq (\text{?}l)\langle l \rangle.\text{If}\ s\ l < \text{max}\ \\
\text{then}\ (c, \text{?}v).\text{acc}_{f(v,l)}\langle c \rangle.\langle c, f(v,l) \rangle.\]

\( S \) repeatedly acquires \( \langle h \rangle \) and depending on the load decides whether to accept requests for new connections coming from \( c \).
Let $\mathcal{C}$ be the c-semiring of QoS values (ranged over by $\kappa$)
The semantics of $KoS$ is defined by the relation

\[ N \xrightarrow{\alpha \kappa} M \]

which states that $N$ performs $\alpha$ with a cost $\kappa$ and becomes $M$.

Local transitions (communications, node or link creations) have unitary QoS value, while the only non-trivial QoS values appear on the transitions that spawn processes or show the presence of links.
(PREF) \[ s :: \gamma.P \xrightarrow{\gamma \boxtimes s}^1 s :: P, \gamma \not\in \{node_{\kappa} \langle t \rangle, con_{\kappa} \langle s \rangle, acc_{\kappa} \langle s \rangle\} \]

(CON) \[
\begin{align*}
N & \xrightarrow{s con_{\kappa} \langle t \rangle}^1 N' & M & \xrightarrow{t acc_{\kappa'} \langle s \rangle}^1 M' \\
N \parallel M & \xrightarrow{\tau}^1 N' \parallel M' \parallel s \xrightarrow{\kappa} t \quad \kappa \leq \kappa'
\end{align*}
\]

(COMM) \[
\begin{align*}
N & \xrightarrow{s (T)}^1 N' & M & \xrightarrow{s t}^1 M' & \bowtie (T, t) = \sigma \\
N \parallel M & \xrightarrow{\tau}^1 N' \sigma \parallel M'
\end{align*}
\]
(LINK) \[ s \xrightarrow{\kappa} t \xrightarrow{s \text{ link } t}{\kappa} 0 \]

(NODE) \[ s :: node_\kappa(t).P \xrightarrow{\text{node}(t)} 1 \xrightarrow{s :: P \parallel s \xrightarrow{\kappa} t \parallel t :: 0, \ s \neq t} \]

(PAR) \[ \begin{align*}
N \xrightarrow{\alpha}{\kappa} N' \\
N \parallel M \xrightarrow{\alpha}{\kappa} N' \parallel M
\end{align*} \]
\[
\begin{align*}
\text{if} \quad & \{ \text{bn}(\alpha) \cap \text{fn}(M) = \emptyset \land \\
& \text{(addr}(N') \setminus \text{addr}(N)) \cap \text{addr}(M) = \emptyset \}
\end{align*}
\]

Rule (NODE) allows a process allocated at \( s \) to use a name \( t \) as the address of a new node and to create a new link from \( s \) to \( t \) exposing the QoS value \( \kappa \). The side condition of (PAR) prevents that new nodes (and links) are created by using addresses of existing nodes.
Local spawning is always enabled while \( \varepsilon_\kappa[P]@t \) from \( s \) is not always possible: the net must contain a path of links from \( s \) to \( t \) suitable wrt \( \kappa \).

(ROUTE) states that \( P \) can traverse a link go an intermediate node \( r \) provided that costs are respected.

(LAND) describes the last hop: in this case, \( P \) is spawned at \( t \), provided that the QoS value of the whole path that has been found is lower than \( \kappa \).
Links in $\mathcal{KoS}$ are public:

$$N \overset{\triangle}{=} s :: \varepsilon_3[P]@t \ || \ s \xrightarrow{1} r \ || \ r :: con_2\langle t \rangle.\varepsilon_2[Q]@t \ || \ t :: acc_2\langle r \rangle,$$

- $s$ and $r$ are trying to spawn a process on $t$ (but no path to $t$ exists).
- $r$ is aware that a link must be first created (and $t$ agrees on that).

Initially, only (CON) can be applied:

$$N' \overset{\triangle}{=} s :: \varepsilon_3[P]@t \ || \ s \xrightarrow{1} r \ || \ r :: \varepsilon_2[Q]@t \ || \ r \xrightarrow{2} t \ || \ t :: \text{nil}.$$  

$r \xrightarrow{2} t$ provides now a path (costing 3) from $s$ to $t$, hence using (PREF), (LINK), (ROUTE) and (LAND) we derive

$$N' \xrightarrow{\tau}{\frac{3}{\text{nil}}} s :: \text{nil} \ || \ r :: \varepsilon_2[Q]@t \ || \ t :: P.$$  

Noteworthy, the migration of $P$ prevents $Q$ to be spawned because the link created by $r$ has been used by $P$. 

A Basic Calculus for Modeling Service Level Agreement, April 14, 2005: 20
Private links can be traversed only by those processes having the appropriate “rights”. Access rights are (particular) names.

\[
N \triangleq s :: \varepsilon_{\{r,s\}}[P]@t \parallel s \{r\} s'
\]

\[
M \triangleq s :: \varepsilon_{\{r,s\}}[P]@t \parallel s \{r,u\} s'
\]

\( P \) can traverse the link in \( N \) but not in \( M \)

Access rights c-semiring:

\[
\mathcal{R} = \langle \wp_{\text{fin}}(\mathcal{S}) \cup \{\mathcal{S}\}, \text{glb}, \cup, \mathcal{S}, \emptyset \rangle
\]

\( X \leq Y \iff Y \subseteq X \)

A private link between the nodes \( s \) and \( t \) can be specified as

\[
(\nu p)(s :: P \parallel s \{p\} t \parallel t :: Q)
\]
Permanent and stable links

\( \kappa \)oS links are vanishing but **permanent links** can be easily encoded:

\[
\begin{align*}
    s &:: \! \text{con}_\kappa \langle t \rangle \parallel t :: \! \text{acc}_\kappa \langle s \rangle \\
\end{align*}
\]

A slight variation are **stable links**, which are links existing until a given condition is satisfied.

\[
\text{Stable}_s \ G \ t \triangleq \! \text{con}_\kappa \langle t \rangle \mid \varepsilon[\text{While } G \text{ do } \text{acc}_\kappa \langle s \rangle \text{ od nil}]@t
\]
Conclusions

“Run, rabbit run
Dig that hole, forget the sun
And when at last the work is done
Don’t sit down it’s time to dig another one

(Breathe, Roger Waters)
Conclusions

We presented $\mathcal{KoS}$

- $\mathcal{KoS}$ aims at conveying the idea that QoS aspects are important for SOC applications
- $\mathcal{KoS}$ formally exploits c-semirings for representing QoS aspects
- c-semirings accounts for uniform handling of multicriteria QoS

Future work

- Further development of $\mathcal{KoS}$ theory e.g., observational semantics for $\mathcal{KoS}$ based on the idea of observing QoS values
- Equipping $\mathcal{KoS}$ with a type systems
  - having dependent types on links and their costs
  - types for access control to deal with QoS attributes
  - types for capturing the notion of contract
- Including $\mathcal{KoS}$ features in existing Klaim implementations
- In the paper we handled the QoS composition in overlay networks We intend to extend $\mathcal{KoS}$ with more general mechanisms for composing overlay networks than simple parallel composition via links
References


