Quality of Service Through Connector Composition

Farhad Arbab

Center for Mathematics and Computer Science (CWI), Amsterdam, The Netherlands
and
Leiden Institute for Advanced Computer Science (LIACS), Leiden University, The Netherlands
and
School of Computer Science, University of Waterloo, Ontario, Canada

11 October 2006

Work in Progress

- In Collaboration With
  - Tom Chothia
  - Meng Sun
  - Young-Joo Moon
- And
  - Rob van der Mei
  - Chretien Verhoef

Motivation

- Providers use components and services from multiple vendors to compose new offerings.
- How to model, analyze, and ensure end-to-end QoS in large-scale distributed systems?
  - Requirements:
    - Wide range of quality attributes
    - Expressiveness/coverage
    - Architectural fidelity
    - Compositional
    - Consistent treatment of all components, services, and connectors.

Quality of Service

- Definition:
  - QoS of a system is a measure of comparing the expected values with the experienced values of a set of attributes of that system.
    - Expected value?
    - Experienced value?

QoS for Behavioral Models

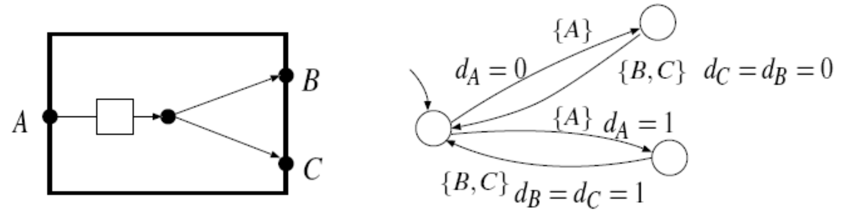
- Behavioral abstraction proposed in Reo offers a suitable model for composition of components/actors and connectors into a system.
  - Computational expressiveness
  - Architectural fidelity
  - Compositional
  - Consistent treatment of components, services, and connector.
- Can this model be extended to accommodate QoS concerns?
  - How?
  - Preserve compositionality?

TDS Semantics

- The TDS semantics of Reo is too fine-grained.
  - Actual time-stamp values often do not matter.
  - Precise relations among specific time-stamp values in different streams:
    - Sometimes intended
    - Sometimes coincidental
  - Atomicity conveyed as equality of real numbers:
    - Too restrictive
    - Unrealistic, especially in distributed systems

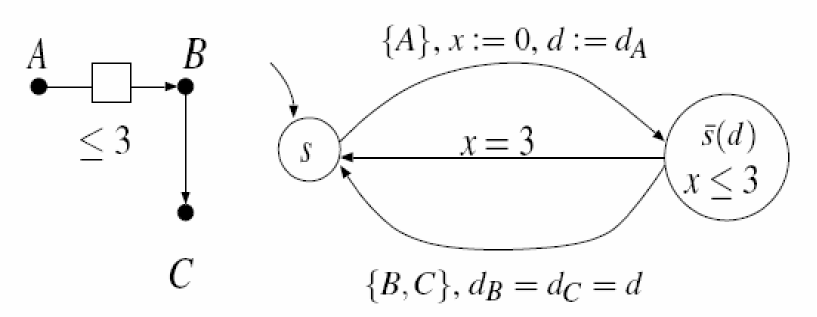
Constraint Automata Semantics

- Abstracts away the time-stamp values.
- Focuses directly on atomicity and order.



Timed Constraint Automata

- Extension of Constraint Automata with time, analogous to Timed Automata.
- States have local clocks that are reset by transitions.



Experience vs. Expectation

- Both TDS and *CA* models use *successful interaction* as the fundamental tick of progress.
- We need new vocabulary to talk about *attempt* vs. *completion* of an interaction.
- Completion can be due to
  - Success (i.e., "the tick" for TDS and *CA*)
  - Time-out

Intermittent Operation Stream

- An Intermittent Operations Stream (IOS) is a twin pair of streams represented as $[\kappa, k]$
- The operation stream $\kappa \in (Data \cup \{\uparrow\})^{\omega}$ $\kappa(0) \in Data$
- The interval stream $k \in \{v \mid (v = @r \vee v = r) \wedge r \in R_{+}\}^{\omega}$
- The clock offset of the interval stream k is $k_{-1} \in R_{+}$
- For convenience, we define $k(-1) = k_{-1}$

		0	1	2	3	4	5	6	...
κ		x_0	x_1	x_2	x_3	$x_4 = \top$	x_5	x_6	...
k	$k_{-1} = 0$	0.5	@0.6	0.4	0.3	0.5	1	@4	...

Plugging of an IOS onto a TDS

Definition 1 *Let* $\langle \alpha, a \rangle$ *represents the TDS at a boundary node of a circuit. We denote by* $[\kappa, k] \vdash \langle \alpha, a \rangle$ *the* plugging *of the IOS* $[\kappa, k]$ *onto* $\langle \alpha, a \rangle$*, where:*

1. $\langle \alpha, a \rangle = \mathcal{S}(\kappa, t)$*, for some completion time stream* $t \in \mathbb{R}_+^\omega$

2. *For operation stream* ψ *and* $p \in \mathbb{R}_+^\omega$*, the function* $\mathcal{S}(\psi, p)$ *is defined as:*

$$\mathcal{S}(\psi, p) \equiv \begin{cases} p(0) < p(1) ? \langle \psi(0).\beta, p(0).b \rangle : \langle \beta, b \rangle \wedge \langle \beta, b \rangle = \mathcal{S}(\psi'', p'') & \text{if } \psi(1) = \top \\ \langle \psi(0).\beta, p(0).b \rangle \wedge \langle \beta, b \rangle = \mathcal{S}(\psi', p') & \text{otherwise} \end{cases}$$

3. *For* $i \geq 0$ *the request time stream* $r \in \mathbb{R}_+^\omega$ *is definied as*

$$r(i) \equiv \begin{cases} v + k_{-1} & \text{if } k(i) = @v \\ k(i) + t(i-1) & \text{otherwise} \end{cases}$$

where $t(-1) = k_{-1}$

4. *Streams* r *and* t *satisfy the infinite set of constraints* $\{\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2, ...\}$*, where for* $i \geq 0$*:*

$$\mathcal{C}_i \equiv \begin{cases} t(i) = v + k_{-1} & \text{if } \kappa(i) = \top \wedge k(i) = @v \\ t(i) = r(i-1) + k(i) & \text{if } \kappa(i) = \top \wedge k(i) \neq @v \\ r(i) \leq t(i) < r(i+1) & \text{otherwise} \end{cases}$$

Example - 1/5

- Consider plugging $[\kappa, k] \vdash \langle \alpha, a \rangle$

		0	1	2	3	4	5	6	...
κ		x_0	x_1	x_2	x_3	$x_4 = \top$	x_5	x_6	...
k	$k_{-1} = 0$	0.5	@0.6	0.4	0.3	0.5	1	@4	...

- Derive requests time stream r (item 3):

		0	1	2	3	4	5	6	...
κ		x_0	x_1	x_2	x_3	$x_4 = \top$	x_5	x_6	...
k	$k_{-1} = 0$	0.5	@0.6	0.4	0.3	0.5	1	@4	...
r		$0.5 + k_{-1}$	$0.6 + k_{-1}$	$0.4 + t(1)$	$0.3 + t(2)$	$0.5 + t(3)$	$1 + t(4)$	$4 + k_{-1}$	...

- Constraints C yield completion time stream t (item 4):

$$
\begin{array}{llll}
C_0 & 0.5 + k_{-1} \leq & t(0) & < 0.6 + k_{-1} \\
C_1 & 0.6 + k_{-1} \leq & t(1) & < 0.4 + t(1) \\
C_2 & 0.4 + t(1) \leq & t(2) & < 0.3 + t(2) \\
C_3 & 0.3 + t(2) \leq & t(3) & < 0.5 + t(3) \\
C_4 & & t(4) & = 0.3 + t(2) + 0.5 \\
C_5 & 1 + t(4) \leq & t(5) & < 4 + k_{-1} \\
C_6 & 4 + k_{-1} \leq & t(6) & < \ldots \\
\vdots & & \vdots &
\end{array}
$$

Example - 2/5

- We obtain $\langle \alpha, a \rangle = \mathcal{S}(\kappa, t)$ from the following table by dropping every column whose data value is "_":

$$\begin{array}{cccccccccc} \alpha & x_0 & x_1 & x_2 & \overline{a}(3) < \overline{a}(4) ? x_3 : \_ & x_5 & x_6 & x_7 & \dots \\ a & \overline{a}(0) & \overline{a}(1) & \overline{a}(2) & \overline{a}(3) & \overline{a}(5) & \overline{a}(6) & \overline{a}(7) & \dots \end{array}$$

- where

$$\begin{array}{rcl} 0.5 + k_{-1} \leq & \overline{a}(0) & < 0.6 + k_{-1} \\ 0.6 + k_{-1} \leq & \overline{a}(1) & < 0.4 + \overline{a}(1) \\ 0.4 + \overline{a}(1) \leq & \overline{a}(2) & < 0.3 + \overline{a}(2) \\ 0.3 + \overline{a}(2) \leq & \overline{a}(3) & < 0.5 + \overline{a}(3) \\ & \overline{a}(4) & = 0.3 + \overline{a}(2) + 0.5 \\ 1 + \overline{a}(4) \leq & \overline{a}(5) & < 4 + k_{-1} \\ 4 + k_{-1} \leq & \overline{a}(6) & < \dots \\ & \vdots & \end{array}$$

Example - 3/5

- Consider plugging $[\lambda, l] \vdash \langle \beta, b \rangle$

		0	1	2	3	4	5	6	...
λ		y_0	y_1	$y_2 = \top$	y_3	y_4	y_5	y_6	...
l	$l_{-1} = 0.8$	0.1	0.1	4	@5.3	0.7	1	@8	...

- Derive requests time stream r (item 3):

		0	1	2	3	4	5	6	...
λ		y_0	y_1	$y_2 = \top$	y_3	y_4	y_5	y_6	...
l	$l_{-1} = 0.8$	0.1	0.1	4	@5.3	0.7	1	@8	...
r		$0.1 + l_{-1}$	$0.1 + t(0)$	$4 + t(1)$	$5.3 + l_{-1}$	$0.7 + t(3)$	$1 + t(4)$	$8 + k_{-1}$	...

- Constraints C yield interaction time stream t (item 4):

$$
\begin{array}{llll}
C_0 & 0.1 + l_{-1} \leq & t(0) & < 0.1 + t(0) \\
C_1 & 0.1 + t(0) \leq & t(1) & < 4 + t(1) \\
C_2 & & t(2) & = 0.1 + t(0) + 4 \\
C_3 & 5.3 + l_{-1} \leq & t(3) & < 0.7 + t(3) \\
C_4 & 0.7 + t(3) \leq & t(4) & < 1 + t(4) \\
C_5 & 1 + t(4) \leq & t(5) & < 8 + l_{-1} \\
C_6 & 8 + l_{-1} \leq & t(6) & < \ldots \\
\vdots & & \vdots &
\end{array}
$$

Example - 4/5

- We obtain $\langle \beta, b \rangle = \mathcal{S}(\lambda, t)$ from the following table by dropping every column whose data value is "_":

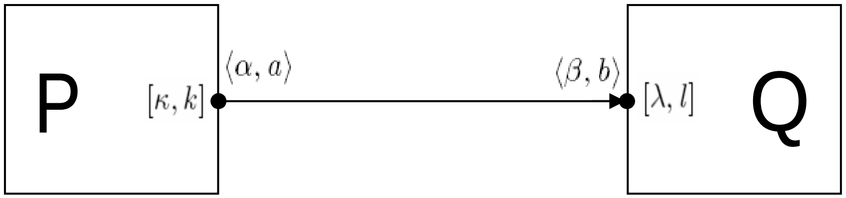
β	y_0	$\overline{b}(1) < \overline{b}(2)?y_1$:_	y_3	y_4	y_5	y_6	y_7	...
b	$\overline{b}(0)$	$\overline{b}(1)$	$\overline{b}(3)$	$\overline{b}(4)$	$\overline{b}(5)$	$\overline{b}(6)$	$\overline{b}(7)$	...

- where

$$
\begin{aligned}
0.1 + l_{-1} \leq\ & \overline{b}(0) && < 0.1 + \overline{b}(0) \\
0.1 + \overline{b}(0) \leq\ & \overline{b}(1) && < 4 + \overline{b}(1) \\
& \overline{b}(2) && = 0.1 + \overline{b}(0) + 4 \\
5.3 + l_{-1} \leq\ & \overline{b}(3) && < 0.7 + \overline{b}(3) \\
0.7 + \overline{b}(3) \leq\ & \overline{b}(4) && < 1 + \overline{b}(4) \\
1 + \overline{b}(4) \leq\ & \overline{b}(5) && < 8 + l_{-1} \\
8 + l_{-1} \leq\ & \overline{b}(6) && < \ldots \\
& \vdots &&
\end{aligned}
$$

Example - 5/5

- If a producer P and a consumer Q are connected by a Sync channel, we have:



$$
\begin{array}{rlllrll}
0.5 + k_{-1} \leq & a(0) & & \wedge & 0.1 + l_{-1} \leq & b(0) & \\
0.6 + k_{-1} \leq & a(1) & & \wedge & 0.1 + b(0) \leq & b(1) & \leq 0.1 + b(0) + 4 \\
0.4 + a(1) \leq & a(2) & & \wedge & 5.3 + l_{-1} \leq & b(2) & \\
0.3 + a(2) \leq & a(3) & \leq 0.3 + a(2) + 0.5 & \wedge & 0.7 + b(3) \leq & b(3) & \\
1 + a(3) \leq & a(4) & & \wedge & 1 + b(4) \leq & b(4) & \\
4 + k_{-1} \leq & a(5) & & \wedge & 8 + l_{-1} \leq & b(5) & \\
& \vdots & & & & \vdots &
\end{array}
$$

- In theory, Sync channel means a=b. But in practice, we may want to impose:

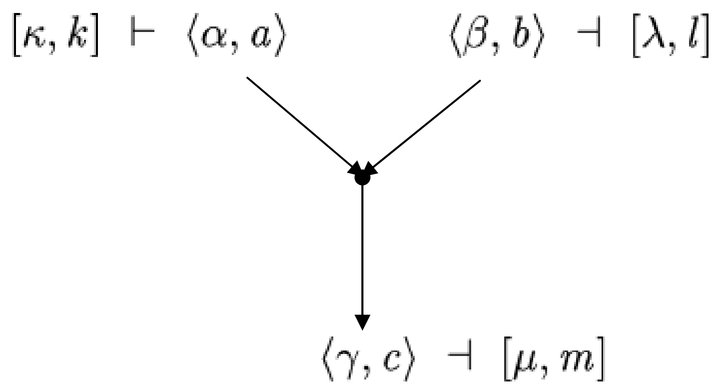
$$|a - b| \leq s$$

$$a(i) \leq b(i) < a(i+1) \leq b(i+1)$$

$$a(i+1) - a(i) \geq t$$

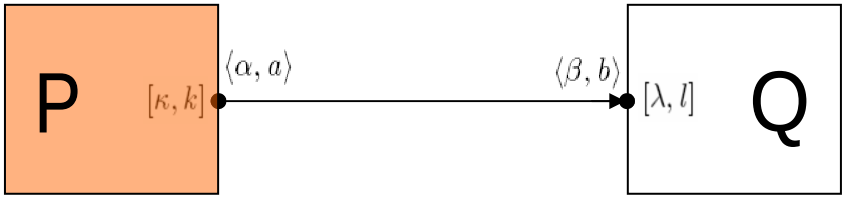
Connector Imposed Delays

- The exclusion of a(i)=b(j) by the merger affects the completion times of the operations as much as the IOSs do.



Performance

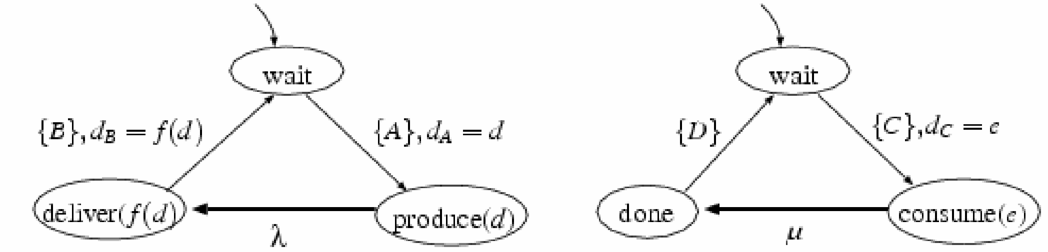
- Fixing specific values for k and l, we can quantify P's experience in a specific run of the system:
  - Average delay between request and completion.
  - Frequency of timeouts versus successful completion.



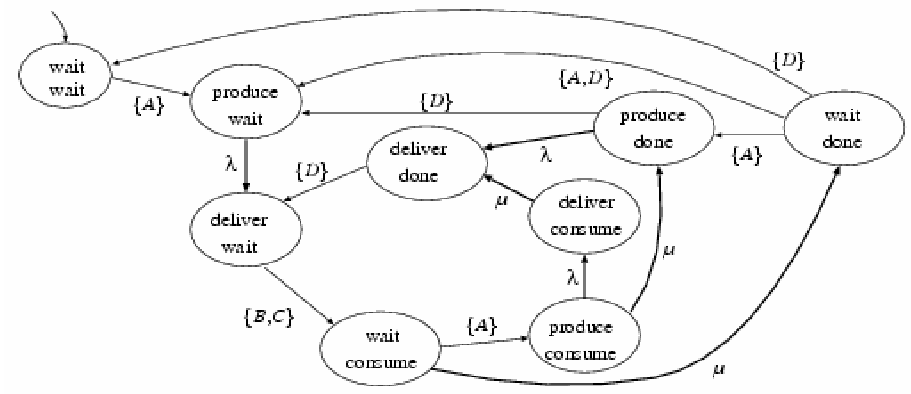
- To properly characterize system performance, we need stochastic variables instead of exact values.

Stochastic Constraint Automata

- Constraint Automata with stochastic (Markov chain) transitions as well as interactive transitions.

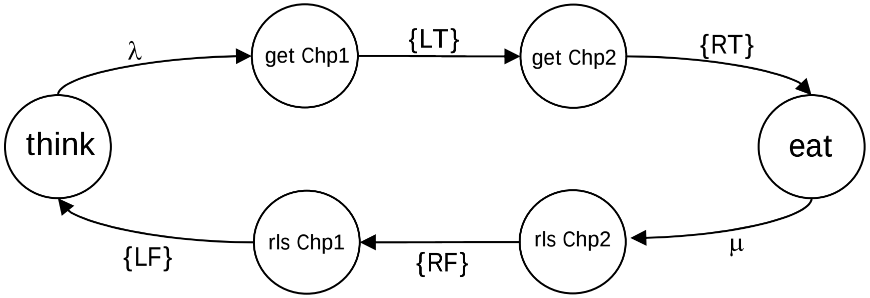


- CA product is extended to allow composition of SCA



SCA Model of a Dining Philosopher

- Stochastic Constraint Automaton for a dining philosopher.



Q-Algebra

- We define a general purpose framework for QoS measures: Q-Algebra.
- "Constraint Semirings" have been used to model QoS values in the past.
- Q-Algebras extend Constraint Semirings with a composition operator.
- We can add these costs to automata models or process calculi to make ***compositional*** models of ***concurrent*** systems.

Constraint Semirings

- Constraint semirings model QoS values with a domain and two operations:
  - \+ picks between values.
  - * combines them.

- both operations are associative and have identities, * is communicative, + distributes over *, etc.

- Examples:
  - Memory assigned:

    Domain: **Z** U { ∞ }, Choose: min, Combine: +
  - Access control
    - Domain: $2^{principals}$, Choose: union, Combine: intersect

Composition Operators

- With concurrent processes there are two ways to combine values: sequentially * and concurrently |
- We define a QoS Algebra: (D,+,*,|,0,1) such that (D,+,*,0,1) and (D,+,|,0,1) are constraint semirings.
- % of CPU needed:
  - Domain: {1,..,100} Choose: max,
  - Combine concurrent: + Combine sequential: max
- Memory assignment:
  - Domain: Z U {∞} Choose: max
  - Combine concurrent: + Combine sequential: +

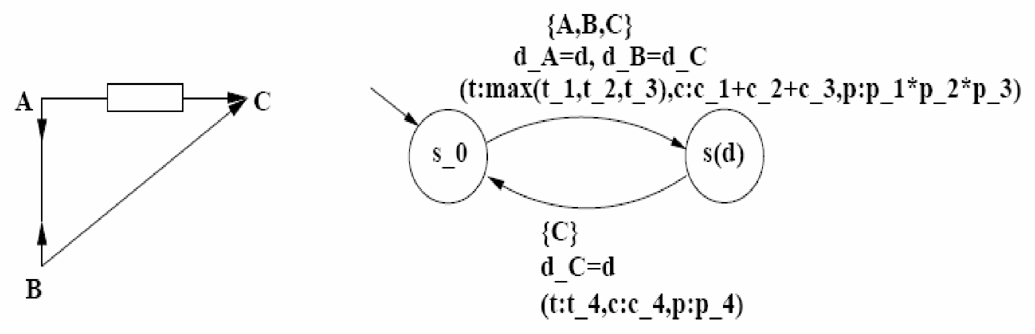
Overview of Q-Algebra

- Q-Algebra is a framework that can be used to model many kinds of QoS value.
- We distinguish between the concurrent and sequential combination of QoS values.
- Q-Algebra costs can be added to a range of formalisms.
- We have defined an automata model with these costs that can be model checked for cost based properties.

Quantified Constraint Automata

Definition 4. *A Q-Constraint Automata is a tuple* $\mathcal{Q} = (S, s_0, \mathcal{N}, R, \longrightarrow)$ *where*

- S *is a set of states, also called configurations,*
- $s_0 \in S$ *is its initial state,*
- $\mathcal{N}$ *is a finite set of nodes,*
- $R = (C, \oplus, \otimes, \odot, \mathbf{0}, \mathbf{1})$ *is a labeled Q-algebra with domain* C *of costs,*
- $\longrightarrow \subseteq \bigcup_{N \subseteq \mathcal{N}} S \times \{N\} \times DC(N) \times C \times S$*, called the transition relation.*



Product of Q-Constraint Automata

- Analogous to product of Constraint Automata, but costs are (parallel-) composed on synchronizing transitions.

Definition 5. *The product of two QCA* $\mathcal{Q}_1 = (S_1, s_{0,1}, \mathcal{N}_1, R_1, \longrightarrow_1)$ *and* $\mathcal{Q}_2 = (S_2, s_{0,2}, \mathcal{N}_2, R_2, \longrightarrow_2)$ *where* R_1 *and* R_2 *are consistent Q-algebra is defined as a Q-Constraint Automata* $\mathcal{Q}_1 \bowtie \mathcal{Q}_2$ *with the components*

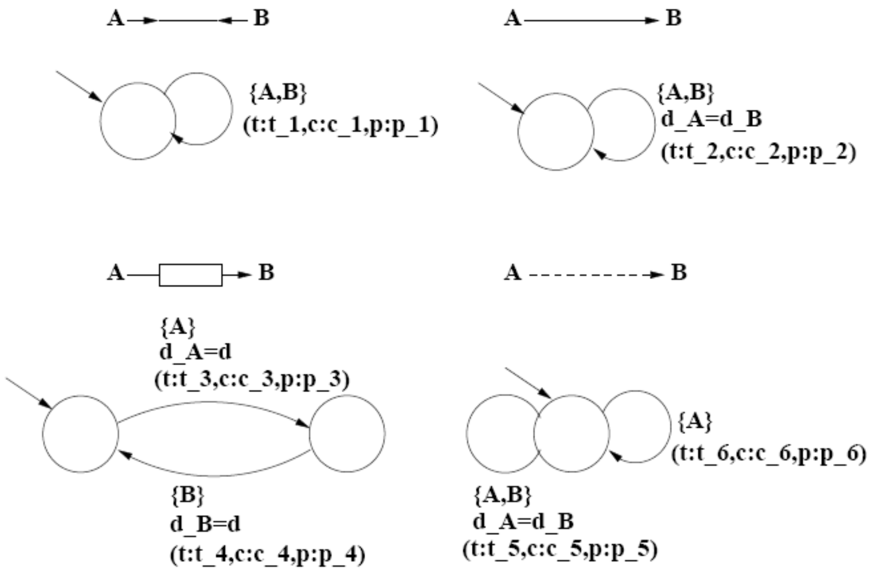
$$(S_1 \times S_2, (s_{0,1}, s_{0,2}), \mathcal{N}_1 \times \mathcal{N}_2, R_1 \bowtie R_2, \longrightarrow)$$

where $\longrightarrow$ *is given by the following rules:*

- *If* $s_1 \xrightarrow{N_1, g_1, c_1}_1 s_1'$, $s_2 \xrightarrow{N_2, g_2, c_2}_2 s_2'$, $N_1 \cap \mathcal{N}_2 = N_2 \cap \mathcal{N}_1$ *and* $g_1 \wedge g_2$ *is satisfiable, then* $\langle s_1, s_2 \rangle \xrightarrow{N_1 \cup N_2, g_1 \wedge g_2, c_1 \odot c_2} \langle s_1', s_2' \rangle$.
- *If* $s_1 \xrightarrow{N, g, c}_1 s_1'$, *where* $N \cap \mathcal{N}_2 = \emptyset$ *then* $\langle s_1, s_2 \rangle \xrightarrow{N, g, c} \langle s_1', s_2 \rangle$.
- *If* $s_2 \xrightarrow{N, g, c}_2 s_2'$, *where* $N \cap \mathcal{N}_1 = \emptyset$ *then* $\langle s_1, s_2 \rangle \xrightarrow{N, g, c} \langle s_1, s_2' \rangle$.

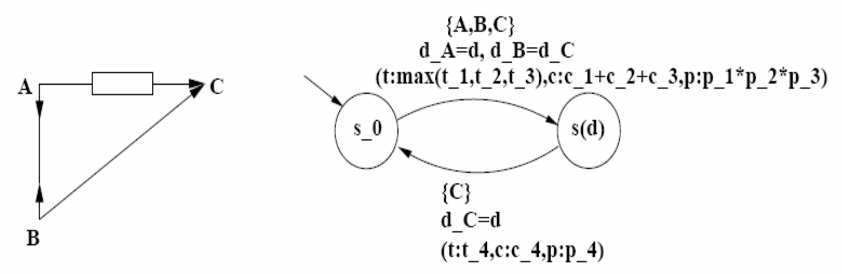
Example - 1/3

- QoS for basic Reo channels:
  - shortest time: $(\mathbb{R}_+ \cup \{\infty\}, min, +, max, \infty, 0)$
  - cost: $(\mathbb{R}_+ \cup \{\infty\}, min, +, +, \infty, 0)$
  - reliability: $([0,1], max, \times, \times, 0, 1)$
- QCA for 4 basic Reo channels:



Example - 2/3

- Consider the alternator circuit and its QCA



- The total cost of the connector is:

$$t = max(t_1, t_2, t_3) + t_4$$

$$c = \sum_{i=1}^{4} c_i$$

$$p = \prod_{i=1}^{4} p_i$$

Example - 3/3

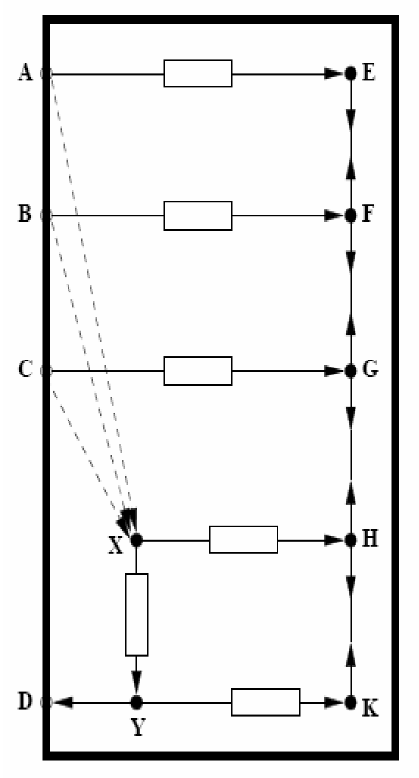
- Two available providers:

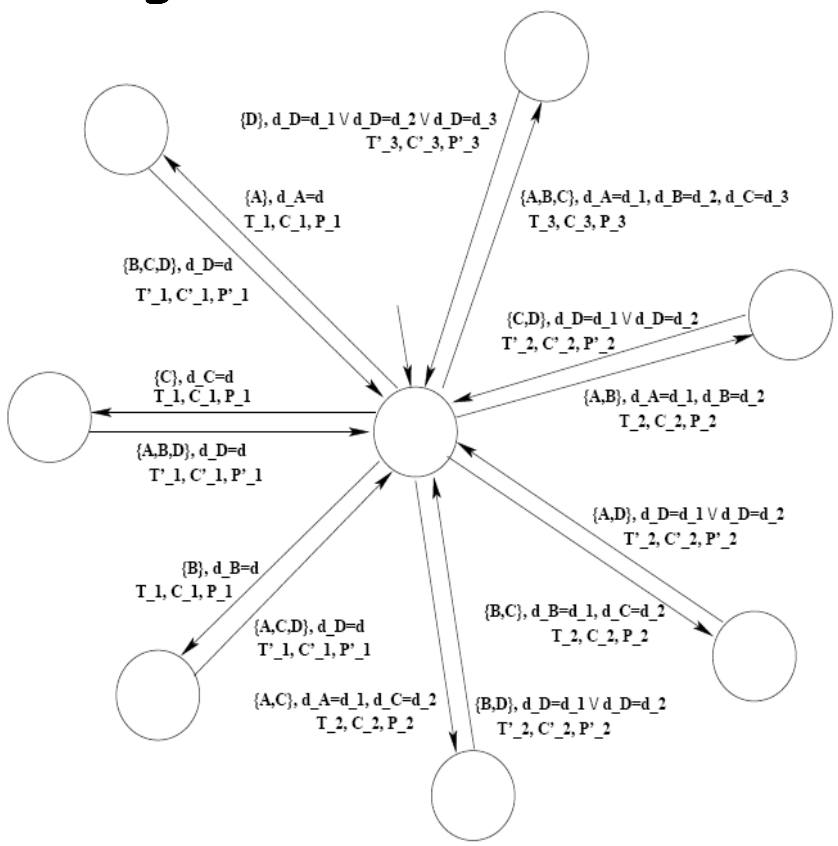
	provider1	provider2
SyncDrain	(t_1=0,c_1=3,p_1=1)	(t_1=0.1,c_1=2,p_1=1)
Sync	(t_2=1,c_2=2,p_2=0.95)	(t_2=1,c_2=8,p_2=0.99)
FIFO1	(t_3=1,c_3=2,p_3=0.9, t_4=0.5,c_4=2,p_4=0.9)	(t_3=1,c_3=5,p_3=1, t_4=1,c_4=5,p_4=0.99)
LossySync	(t_5=1,c_5=2,p_5=0.95, t_6=0.1,c_6=1,p_6=0.95)	(t_5=1,c_5=3,p_5=0.99, t_6=0.2,c_6=0.5,p_6=0.99)

- Requirements:
  - Cost no more than 15
  - Reliability greater than 90%
- Alternatives:
  - All provider 1: t=1.5, c=9, p=0.7696
  - All provider 2: t=2, c=20, p=0.9801
  - Sync by provider 1; SyncDrain and FIFO1 by provider 2: t=2, c=14, p=0.9405

Example: Discriminator Circuit

- Composed QCA after hiding





Example: Alternative Deployments

- Composed cost values:

$$T_1 = max\{t_3, t_5\}$$
$$C_1 = 3 \times c_3 + c_5$$
$$P_1 = p_3^3 \times p_5$$
$$T_1' = max\{t_2, t_3, t_3 + t_4, t_6\} + max\{t_4, t_1 + t_4\}$$
$$C_1' = 4 \times c_1 + c_2 + 3 \times c_3 + 6 \times c_4 + 2 \times c_6$$
$$P_1' = p_1^4 \times p_2 \times p_3^3 \times p_4^6 \times p_6^2$$
$$T_2 = max\{t_3, t_5\}$$
$$C_2 = 4 \times c_3 + 2 \times c_5$$
$$P_2 = p_3^4 \times p_5^2$$
$$T_2' = max\{t_2, t_3, t_3 + t_4, t_6\} + max\{t_4, t_1 + t_4\}$$
$$C_2' = 4 \times c_1 + c_2 + 2 \times c_3 + 7 \times c_4 + c_6$$
$$P_2' = p_1^4 \times p_2 \times p_3^2 \times p_4^7 \times p_6$$
$$T_3 = max\{t_3, t_5\}$$
$$C_3 = 5 \times c_3 + 3 \times c_5$$
$$P_3 = p_3^5 \times p_5^3$$
$$T_3' = max\{t_2, t_3 + t_4\} + max\{t_4, t_1 + t_4\}$$
$$C_3' = 4 \times c_1 + c_2 + c_3 + 7 \times c_4$$
$$P_3' = p_1^4 \times p_2 \times p_3 \times p_4^7$$

- All provider 1:

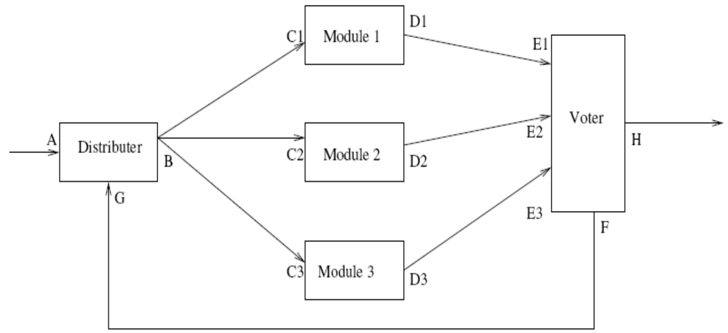
$$T_1 = T_2 = T_3 = 1, T_1' = T_2' = T_3' = 2,$$
$$C_1 = 8, C_2 = 12, C_3 = 16, C_1' = 34, C_2' = 33, C_3' = 30,$$
$$P_1 \approx 0.69, P_2 \approx 0.48, P_3 \approx 0.5, P_1' \approx 0.33, P_2' \approx 0.39, P_3' \approx 0.41$$

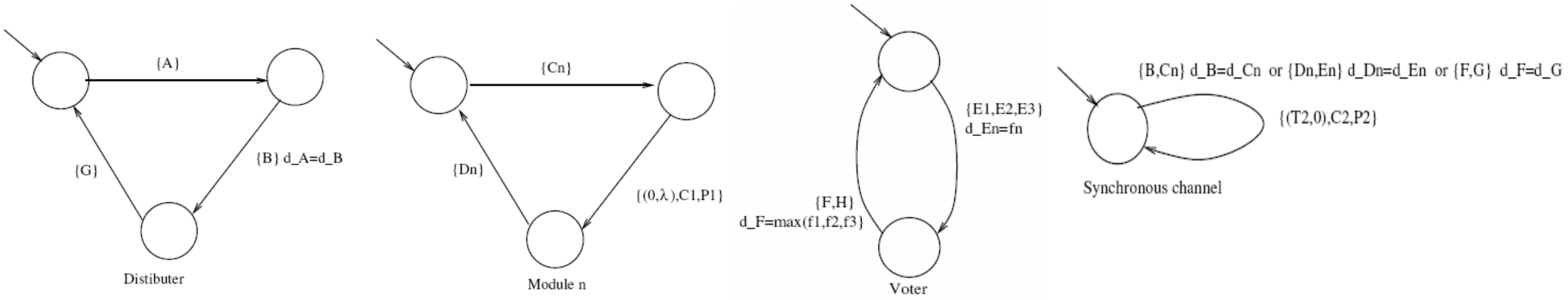
- All provider 2:

$$T_1 = T_2 = T_3 = 1, T_1' = T_2' = T_3' = 3.1,$$
$$C_1 = 18, C_2 = 26, C_3 = 34, C_1' = 62, C_2' = 61.5, C_3' = 56,$$
$$P_1 = 0.99, P_2 \approx 0.98, P_3 \approx 0.97, P_1' \approx 0.89, P_2' \approx 0.89, P_3' \approx 0.92$$

Stochastic Q-Constraint Automata

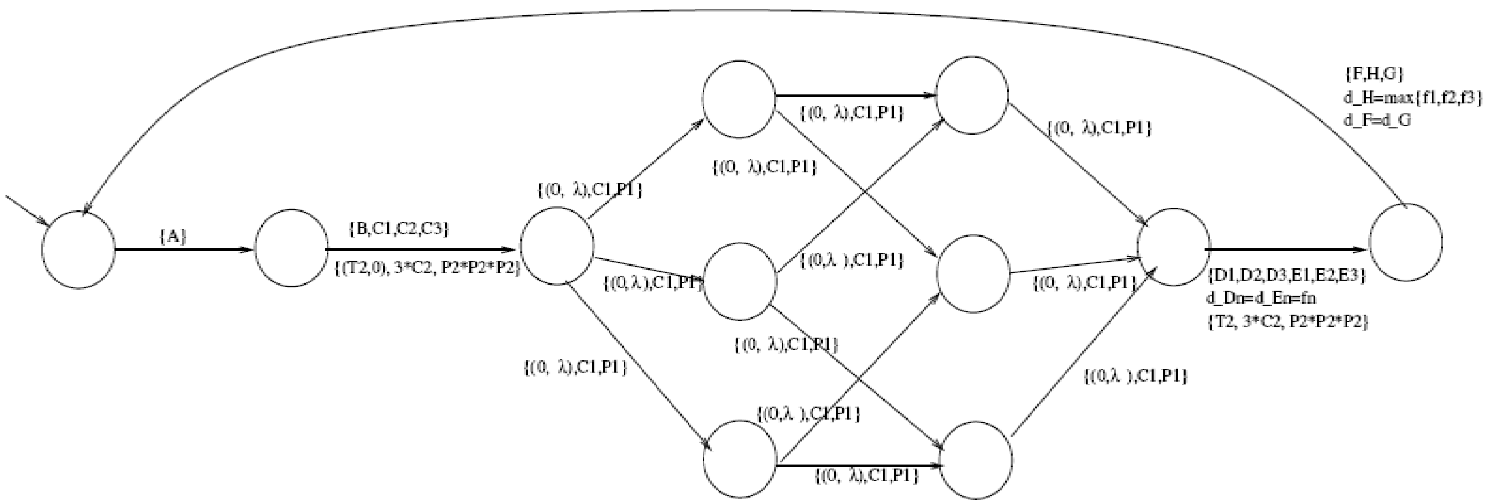
- Example: Triple modular redundancy system





Composed SQCA for TMR System

- How to hide internal states involving stochastic transition?



Summary

- Reo offers a powerful structural framework for composition of QoS properties.
- QCA (stochastic or otherwise) serve as good models for both Reo circuits and components/subsystems with QoS properties.
- Choices of actual composition operators for Q-algebras in some domains is non-trivial.
- Hiding of intermediate states/transitions of composed SQCA?