Translation from Quantitative Intentional Automata into Markov Chains

Young-Joo Moon

CWI

October 23, 2007

Contents

- Motivation
- Related work
- Reo and Intentional Automata
- Work flow
- Example
- Conclusion

Motivation I

Existing Formalisms and Tools

- Reo language
  - a channel-based glue language for coordination models
- Constraint Automata
  - operational semantics for Reo language
- Variations of Reo language and Constraint Automata
  - Quantitative Reo language
  - Quantitative Constraint Automata(QCA)

However, these formalisms do not explain quantitative aspects derived from the environment, for example,

- Throughput
- Response time

Motivation II

Markov Chains(MCs)

- Stochastic model for performance evaluation
- Memoryless property
- Continuous-time MC and Discrete-time MC

The translation from Reo language into MCs is considered in order to

- account for quantitative aspects from the environment
- implement an integrated tool for modeling functionality and performance evaluation

Related work

Measure Specification Language(MSL) provides

- specification of performance measures in component-oriented way
- mixed approach
  - compositional framework by Stochastic Process Algebra(SPA)
  - performance evaluation by Action-labeled Continuous Time Markov Chains(ACTMCs)

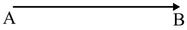
Comparison to our methodology

- compositional framework by Quantitative Reo language
- performance evaluation by derived MC

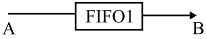
  ⇒ The derived MC has compact state space because of the information of synchronous behavior

Reo language

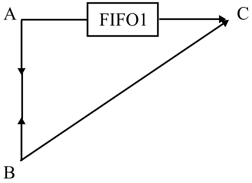
- Reo language
  - a channel-based “glue language”
  - primitive channels and complex application called connectors
  - synchronousy and asynchronousy behavior
- Quantiative Reo language
  - variation of Reo language
  - compositional specification of a system behavior with the quantity (i.e., data flow delay)



Synchronous channel



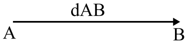
FIFO channel



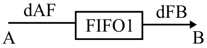
Ordering connector

Reo language

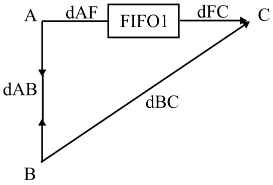
- Reo language
  - a channel-based “glue language”
  - primitive channels and complex application called connectors
  - synchronousy and asynchronousy behavior
- Quantitative Reo language
  - variation of Reo language
  - compositional specification of a system behavior with the quantity (i.e., data flow delay)



Synchronous channel



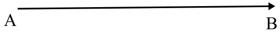
FIFO channel

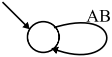


Ordering connector

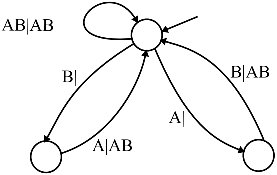
Intentional Automata

- Intentional Automata(IA)
  - specification of a system behavior with the environment information
  - data arrivals at ports and processing between ports





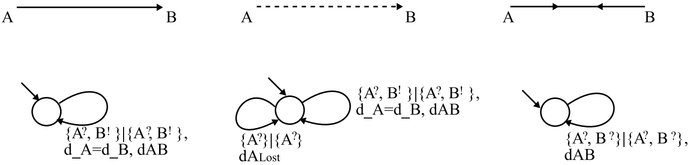
Constraint Automata(CA)



Intentional Automata(IA)

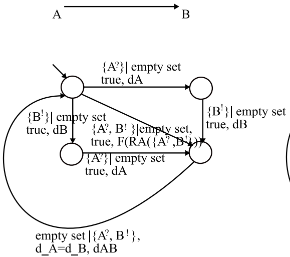
Quantitative Intentional Automata(QIA)

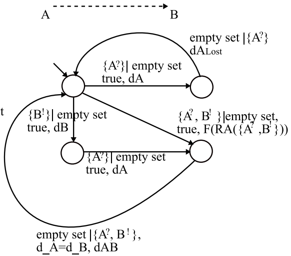
- Concept of IA and quantity
- Separation input and output ports
- Processing delay(dAB,dAF,dFB) is given.
  - Q-algebra for delay calculation

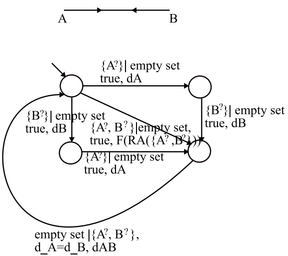


Extended QIA(EQIA)

- Representation explicit request arrivals
- Separation request arrivals and data flow processing
- Given set of request inter-arrival time







Work flow

Final Goals

- Translation from Quantitative Reo circuit to MC
- Integrate tool implementation from specification of a system behavior to performance evaluation

- Intermediate steps
  - Quantitative Reo circuit into QIA
  - QIA into MC
- Extending existing tools and implementing its translation

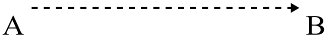
QIA into MC

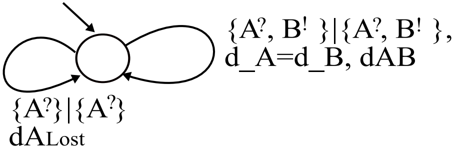
- Assumptions
  - The order of processing delays can be deduced.
    - $d_1; d_2$: d_2 follows d_1.
    - $d_1 \parallel d_2$: d_1 and d_2 happen in parallel.
  - The delay distribution is exponentially distributed.
  - The synchronous behaviors happen atomically.
  - Decision of which reaction is instantaneous.
- QIA transition $\rightarrow_{QIA}$
  - request arrivals of an atomic behavior
    - single arrival in non-deterministic way
    - parallel arrivals
  - processing of an atomic behavior
- MC transition $\rightarrow_{MC}$
  - single event
    - single request arrival at a port
    - single processing for an atomic behavior

QIA into MC

Translation

- extending QIA
- adding missing arrivals
- keeping single data arrival and single processing
- adding intermediate transitions for prallel processing
- dealing with parallel request arrivals

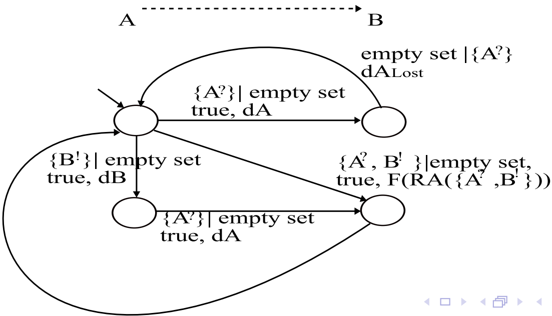




QIA into MC

Translation

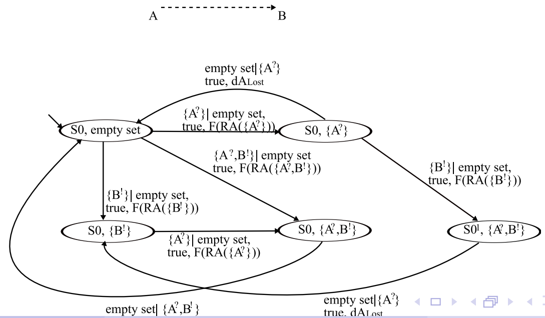
- extending QIA
- adding missing arrivals
- keeping single data arrival and single processing
- adding intermediate transitions for prallel processing
- dealing with parallel request arrivals



QIA into MC

Translation

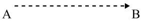
- extending QIA
- adding missing arrivals
- keeping single data arrival and single processing
- adding intermediate transitions for prallel processing
- dealing with parallel request arrivals

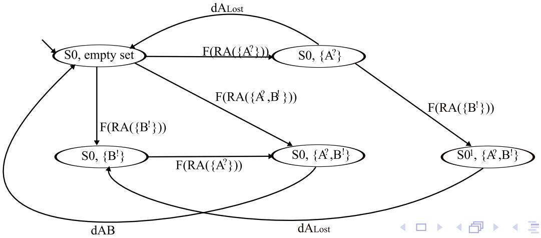


QIA into MC

Translation

- extending QIA
- adding missing arrivals
- keeping single data arrival and single processing
- adding intermediate transitions for prallel processing
- dealing with parallel request arrivals

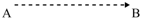


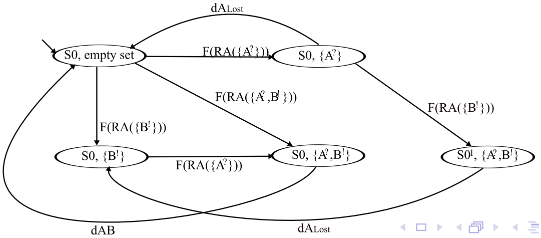


QIA into MC

Translation

- extending QIA
- adding missing arrivals
- keeping single data arrival and single processing
- adding intermediate transitions for prallel processing
- dealing with parallel request arrivals

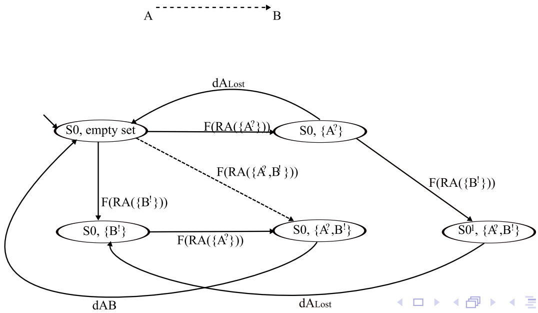




QIA into MC

Translation

- extending QIA
- adding missing arrivals
- keeping single data arrival and single processing
- adding intermediate transitions for prallel processing
- dealing with parallel request arrivals



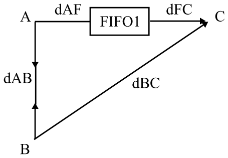
QIA into MC

Translation for parallel processing $s_1 \xrightarrow{\emptyset, N, g, d} s_2$

1. If d is a single delay, then add $s_1 \xrightarrow{d} s_2$.
2. If d $= d_1 \parallel d_2 \parallel \cdots \parallel d_k$, then for each transition,
   - $\forall\, d_i,\ s_1 \xrightarrow{d_i} ts_i$
   - $\forall\, d_j,\ ts_i \xrightarrow{d_j} ts_{ij}$ where $i \neq j$
   - $\ldots$
   - $\forall\, d_k,\ ts_{ij\ldots l} \xrightarrow{d_k} s_2$

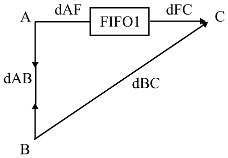
   go back to step 1.
3. If d $= d_1 ; d_2 ; \cdots ; d_k$, then for each transition, $s_1 \xrightarrow{d_1} ts_1, ts_1 \xrightarrow{d_2} ts_2, \cdots, ts_{k-1} \xrightarrow{d_k} s_2$, go back to step 1.

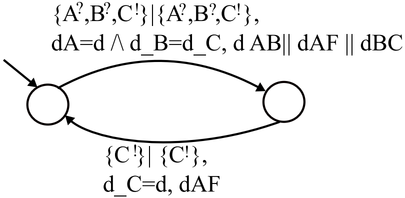
Example2



- variables for configuration : A, B, C, dAB, dBC, dAF, dFC
- number of states of MC : $2^7 = 128$ states
  - port variables : ready for processing
  - delay variables : in processing

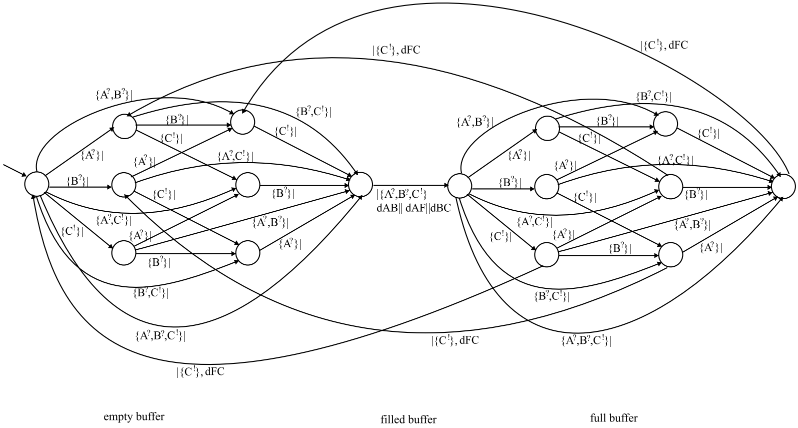
Example2



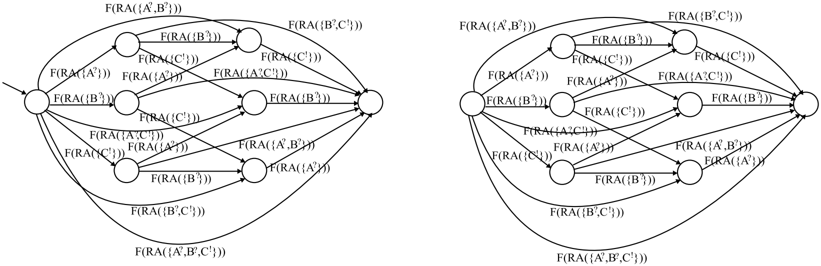

$\{A^{?},B^{?},C^{!}\}|\{A^{?},B^{?},C^{!}\}$,
dA=d /\ d_B=d_C, d AB|| dAF || dBC

$\{C^{!}\}|\ \{C^{!}\}$,
d_C=d, dAF

Example2


$|\{C^!\}$, dFC
$|\{C^!\}$, dFC
$\{A^?,B^?\}|$
$\{B^?\}|$
$\{C^!\}|$
$\{B^?,C^!\}|$
$\{C^!\}|$
$\{A^?\}|$
$\{A^?\}|$
$\{A^?,C^!\}|$
$\{B^?\}|$
$\{C^!\}|$
$\{B^?\}|$
$|\{A^?,B^?,C^!\}$
dAB|| dAF||dBC
$\{A^?,C^!\}|$
$\{C^!\}|$
$\{A^?\}|$
$\{A^?,B^?\}|$
$\{A^?\}|$
$\{B^?\}|$
$\{B^?,C^!\}|$
$\{A^?,B^?,C^!\}|$
$|\{C^!\}$, dFC
$|\{C^!\}$, dFC
$\{A^?,B^?\}|$
$\{B^?,C^!\}|$
$\{B^?\}|$
$\{C^!\}|$
$\{C^!\}|$
$\{A^?\}|$
$\{A^?\}|$
$\{A^?,C^!\}|$
$\{B^?\}|$
$\{C^!\}|$
$\{B^?\}|$
$\{A^?,C^!\}|$
$\{C^!\}|$
$\{A^?\}|$
$\{A^?,B^?\}|$
$\{B^?\}|$
$\{A^?\}|$
$\{B^?,C^!\}|$
$\{A^?,B^?,C^!\}|$
empty buffer
filled buffer
full buffer

Example2

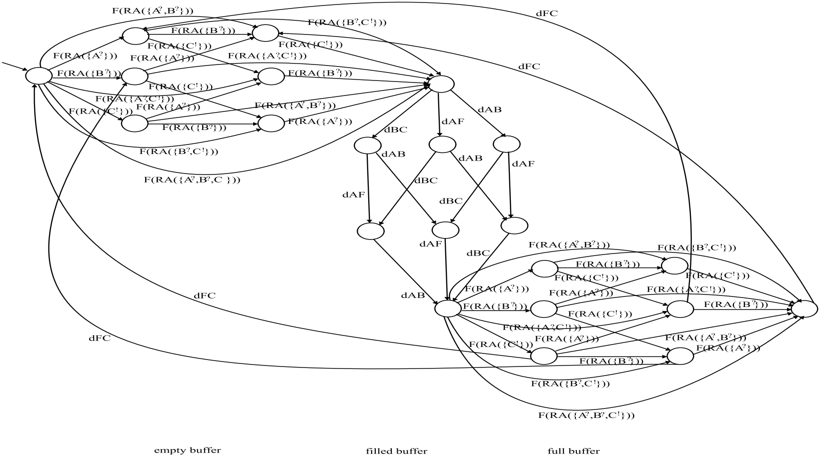


empty buffer

filled buffer

full buffer

Example2



- In total, 22 states

Conclusion

- Reo language provides
    - compositional specification of a system behavior
    - synchronousy information

  ,but can not explain the environment.
- By the translation from Reo into MC
    - accounting for the environment with quantity
    - implementing an integrated tool for modeling functionality and performance evaluation