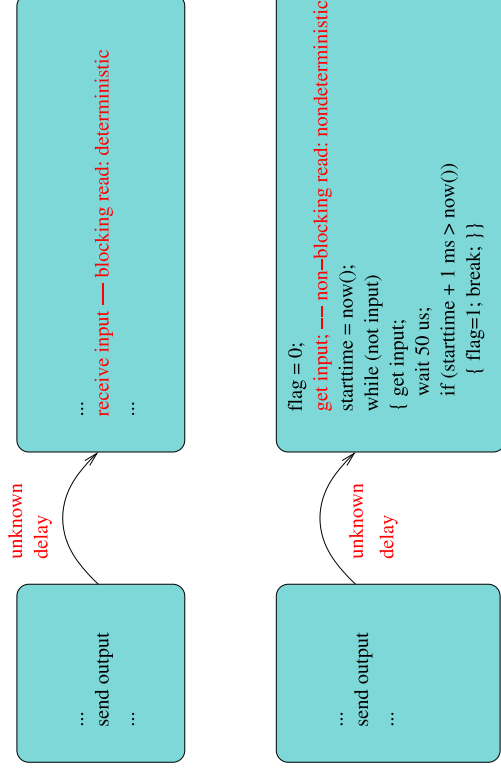


## Modeling Unknown Communication Delay



## System Modeling

- Introduction
- Rugby Meta-Model
- Finite State Machines
- Petri Nets
- Untimed Model of Computation
- Synchronous Model of Computation
- Timed Model of Computation
- Integration of Computational Models
- Tightly Coupled Process Networks

### Nondeterminism and Probability

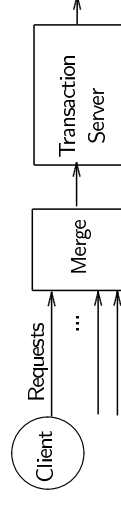


## Purpose of Nondeterminism

- Descriptive Purpose
  - ★ Modeling of an entity with limited knowledge (e.g. unknown communication time)
  - ★ Nondeterminism as an abstraction mechanism;
  - ★ All possibilities must be exercised for validation;
- Constraining Purpose
  - ★ The specification gives freedom to the implementation
  - ★ One possibility must be implemented



## Unspecified Merge of Inputs



The Merge model should have the following properties:

- Requests are not ordered by the Merge
- Absent request should not block the Merge
- The Merge should be fair



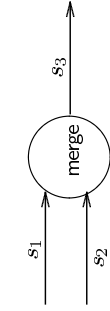
## Merge as an Untimed Process Relation

$$\text{merge}(s_1, \langle \rangle) = s_1$$

$$\text{merge}(\langle \rangle, s_2) = s_2$$

$$\text{merge}(e_1 \oplus s_1, e_2 \oplus s_2) = \{e_1 \oplus s_3 : s_3 \in \text{merge}(s_1, e_2 \oplus s_2)\}$$

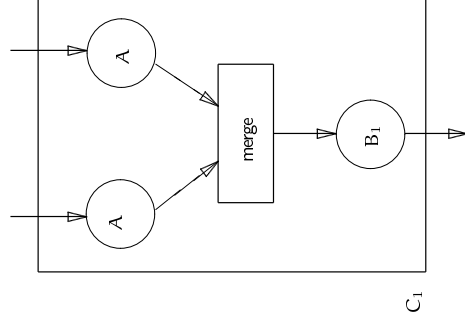
$$\cup \{e_2 \oplus s_3 : s_3 \in \text{merge}(e_1 \oplus s_2, s_2)\}$$



$s_1$	$s_2$	$s_3$
$\langle a, b \rangle$	$\langle 1, 2 \rangle$	$\{ \langle a, b, 1, 2 \rangle, \langle a, 1, b, 2 \rangle, \langle a, 1, 2, b \rangle, \langle 1, a, b, 2 \rangle, \langle 1, a, 2, b \rangle, \langle 1, 2, a, b \rangle \}$
$\langle c, d \rangle$	$\langle 3, 4 \rangle$	$\{ \langle c, d, 3, 4 \rangle, \langle c, 3, d, 4 \rangle, \langle c, 3, 4, d \rangle, \langle 3, c, d, 4 \rangle, \langle 3, c, 4, d \rangle, \langle 3, 4, c, d \rangle \}$



## The Brock - Ackerman Anomaly - 1



$$C_1(s_1, s_2) = B_1(\text{merge}(A(s_1), A(s_2)))$$

$$A(\langle \rangle) = \langle \rangle$$

$$A(e \oplus s) = \langle e, e \rangle$$

$$B_1(\langle \rangle) = \langle \rangle$$

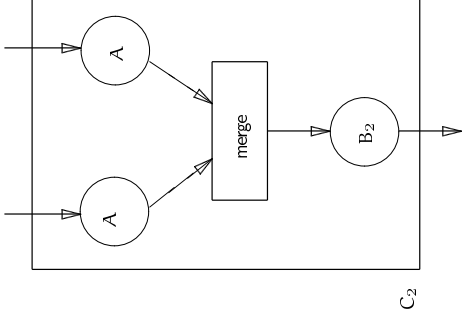
$$B_1(\langle e \rangle \oplus s) = \langle e \rangle \oplus f_1(s)$$

$$f_1(\langle \rangle) = \langle \rangle$$

$$f_1(\langle e' \rangle \oplus s) = \langle e' \rangle$$



## The Brock - Ackerman Anomaly - 2



$$C_2(s_1, s_2) = B_2(\text{merge}(A(s_1), A(s_2)))$$

$$A(\langle \rangle) = \langle \rangle$$

$$A(e \oplus s) = \langle e, e \rangle$$

$$B_2(\langle \rangle) = \langle \rangle$$

$$B_2(\langle e \rangle \oplus s) = f_2(e, s)$$

$$f_2(e, \langle \rangle) = \langle \rangle$$

$$f_2(e, \langle e' \rangle \oplus s) = \langle e, e' \rangle$$



## The Brock - Ackerman Anomaly - 3

$$C_1(s_1, s_2) = B_1(\text{merge}(A(s_1), A(s_2)))$$

$$A(\langle \rangle) = \langle \rangle$$

$$A(e \oplus s) = \langle e, e \rangle$$

$$B_1(\langle \rangle) = \langle \rangle$$

$$B_1(\langle e \rangle \oplus s) = \langle e \rangle \oplus f_1(s)$$

$$f_1(\langle \rangle) = \langle \rangle$$

$$f_1(\langle e' \rangle \oplus s) = \langle e' \rangle$$

$$C_2(s_1, s_2) = B_2(\text{merge}(A(s_1), A(s_2)))$$

$$A(\langle \rangle) = \langle \rangle$$

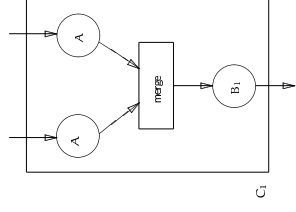
$$A(e \oplus s) = \langle e, e \rangle$$

$$B_2(\langle \rangle) = \langle \rangle$$

$$B_2(\langle e \rangle \oplus s) = f_2(e, s)$$

$$f_2(e, \langle \rangle) = \langle \rangle$$

$$f_2(e, \langle e' \rangle \oplus s) = \langle e, e' \rangle$$



$$C_i(\langle \rangle, \langle \rangle) = \langle \rangle$$

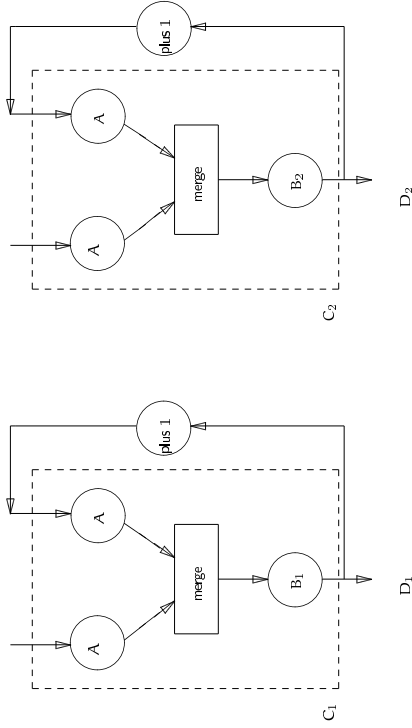
$$C_i(e \oplus s, \langle \rangle) = \langle e, e \rangle$$

$$C_i(\langle \rangle, e' \oplus s') = \langle e', e' \rangle$$

$$C_i(e \oplus s, e' \oplus s') = \{ \langle e, e \rangle, \langle e, e' \rangle, \langle e', e \rangle, \langle e', e' \rangle \}$$



### The Brock - Ackerman Anomaly - 4

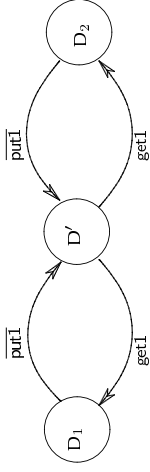


$$D_1((5)) = \{(5, 5), (5, 6)\}$$

$$D_2((5)) = \{(5, 5)\}$$



### Weak Determinacy



$$D' \stackrel{\text{def}}{=} \text{get1}.D_1 + \text{get1}.D_2$$

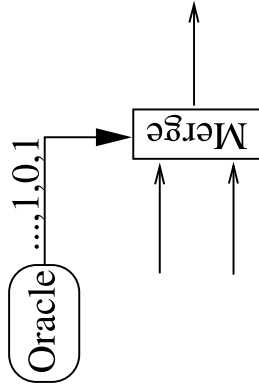
$$D_1 \stackrel{\text{def}}{=} \overline{\text{put1}}.D'$$

$$D_2 \stackrel{\text{def}}{=} \overline{\text{put1}}.D'$$

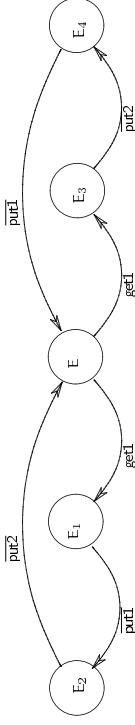
If a system behaves deterministically as far as it can be observed from the environment, the system is weakly determinate.



### Oracle Based Nondeterminism



### Weak Confluence



If, for every two possible actions, the occurrence of one can never preclude the other, the system is weakly confluent.



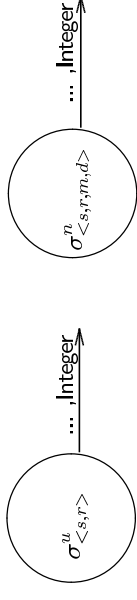
## Modeling with Non-deterministic Components

- Weak determinacy or weak confluence is desirable
- Design rules to achieve determinacy and and confluence are necessary
- Example rules:
  - ★ Deploying **blocking read** ensures deterministic system behaviour in the presence of non-deterministic communication and computation time.
  - ★ Process composition of processes with disjoint inputs preserves weak confluence.



## The $\sigma$ Process

Stochastic processes can be used for descriptive and constraining purpose.



$$\text{sigmaC}(min, max, w_0) = \sigma^u_{<w_0, [min, max]>}$$

where

$$\sigma^u_{<w_0, [min, max]>}() = s'$$

$$\forall e' \in s' : min \leq val(e') \leq max$$

$$\wedge \forall v \in [min, max] : p_v(e') = \frac{1}{max - min + 1}$$

$p_v(e')$ ...probability that the value of  $e'$  is  $v$



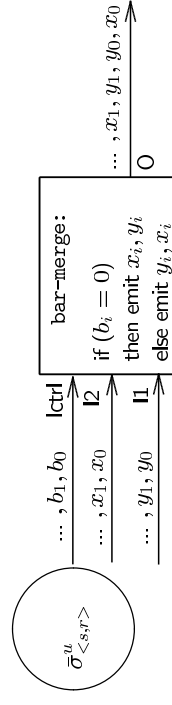
## Modeling Approach with Deterministic Components

- Restrict all modeling elements to be deterministic
- Constrain implementation such that it is equivalent with the specification model
- Examples:
  - ★ Blocking read
  - ★ Synchronous assumptions



## Usage of Stochastic Processes

- An implemented **sigma bar process**  $\bar{\sigma}$  can generate any of the possible outputs of a  $\sigma$ .
- An implemented **sigma tilde processes**  $\tilde{\sigma}$  must exhibit the statistical properties of  $\tilde{\sigma}$ .
- Verification follows the interpretation of implementation for  $\sigma$  processes.



## Select Based Process Constructors

$$\begin{aligned} \text{select}(\delta, f, g) &= f & \text{if } \delta = 0 \\ &= g & \text{if } \delta = 1 \end{aligned}$$

$$\begin{aligned} \text{selMapS}(f_1, f_2, r_0) &= p \\ &\text{with} \\ & p(\bar{s}) = p_1(p_2(\sigma_{<r_0, [0,1]}^u, \bar{s})) \\ & p_1 = \text{mapS}(f) \\ & p_2 = \text{zipS}() \\ & f((\delta, \bar{e})) = \text{select}(\delta, f_1, f_2)(\bar{e}) \end{aligned}$$



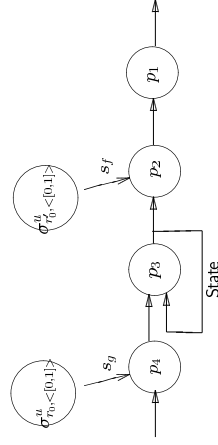
## Consolidation Based Constructor

$$\begin{aligned} \text{conMooreS}(g, f, r_0, r'_0, w_0) &= p \\ &\text{with} \\ & p(\bar{s}) = p_1(p_2(s_f, p_3(p_4(s_g, \bar{s})))) \\ & s_f = \sigma_{<r'_0, [0,1]}^u \\ & s_g = \sigma_{<r_0, [0,1]}^u \\ & p_1 = \text{mapS}(f) \\ & p_2 = \text{zipS}() \\ & p_3 = \text{scanS}(g, w_0) \\ & p_4 = \text{zipS}() \end{aligned}$$



## Select Based Moore Constructor

$$\begin{aligned} \text{selMooreS}(g_1, g_2, f_1, f_2, r'_0, w_0) &= p \\ &\text{with} \\ & p(\bar{s}) = p_1(p_2(s_f, p_3(p_4(s_g, \bar{s})))) \\ & s_f = \sigma_{<r'_0, [0,1]}^u \\ & s_g = \sigma_{<r_0, [0,1]}^u \\ & p_1 = \text{mapS}(f) \\ & p_2 = \text{zipS}() \\ & p_3 = \text{scanS}(g, w_0) \\ & p_4 = \text{zipS}() \\ & f((\delta, \bar{e})) = \text{select}(\delta, f_1, f_2)(\bar{e}) \\ & g(w, (\delta, \bar{e})) = \text{select}(\delta, g_1, g_2)(w, \bar{e}) \end{aligned}$$



## Summary

- Descriptive Purpose - Constraining Purpose
- Modeling the unknown
- Deterministic and Nondeterministic Merge
- Modeling Nondeterminism with History Relations
- Nondeterministic Components and Deterministic Systems
- Usage of stochastic processes
  - ★ To model the system's environment;
  - ★ To constrain the system under design;
  - ★ For modeling testbenches and input stimuli



## System Modeling - Summary

- Modeling Basics: State, Event, Next State and Output Encoding Function, Continuous and Discrete State/Time Models
- Rugby Meta-model
- Finite State Machines
- Petri Nets
- Untimed Model of Computation
- Synchronous Model of Computation
- Discrete Time Model of Computation
- Hierarchical Model of Computation with Heterogeneous Time Domains
- Coupling Effects in Process Networks
- Nondeterminism and Stochastic Processes

