# Formal Modeling of Clearing and Settlement Overview-Tutorial

Michael Alexander
Wirtschaftsuniversität Wien
malexand@wu-wien.ac.at

August 24, 2007

# Agenda

Overview

Process Algebras

Program Transformation

Model Checking

Project Overview

Selected References

# Overview

# Properties of Clearing and Settlement



✔ Clearing and Settlement Process are

- ✘ Stateful with Considerable State Spaces
- ✘ *Concurrent*
- ✘ Asynchronous
- ✘ Processes and Software need to be Operationally Correct
- ✘ Inherent Systemic Risk
- ✘ Difficult to Debug, Costly Software Bugs
- ✘ Interlinked

# Formal Languages Applications



- ✔ Modeling Systems and Processes Unambiguously
- ✔ Formal Semantics In Mathematical Notation
- ✔ Facilitate Spotting Conditions Such As Deadlocks, Livelocks And Resource Starvation
- ✔ Can Nullify The Possibility Of Unforeseen Scenarios in PSPACE-Complete Problems
- ✔ Testing of State Permutations
- ✔ Assure Correctness In Complex Interactions Between Agents
- ✔ Specify - and Test Adherence To Overall Systems Constraints

# Formal Languages Applications II

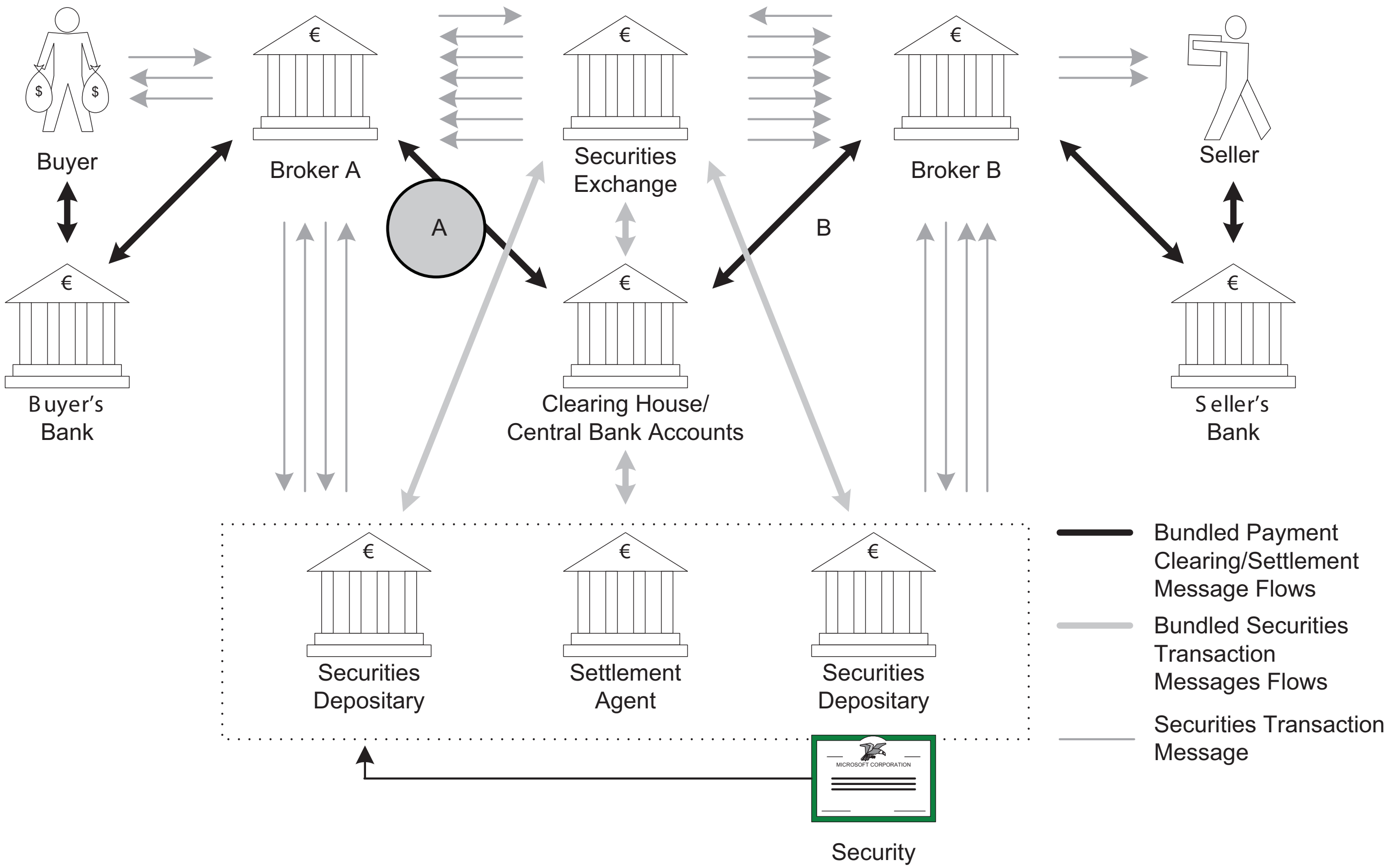


- ✔ Applications Target Target Behavioral Properties
  - ✘ Formally Describe (*not invent*) a Specification
  - ✘ Test for Correctness of Properties of a Specificationy

# Critical Process Segment Modeling



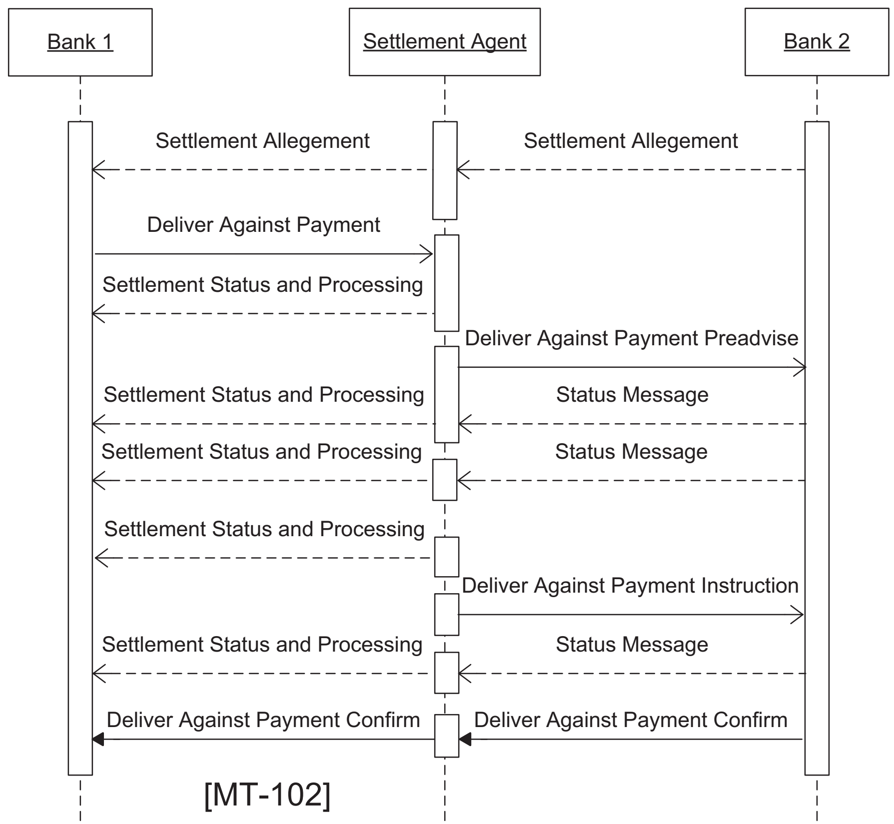
## ✔ Step 1: Focus Paths {A, B}



# Process Segment Modeling II



✔ Step 2: Identifying Agents, (Sub) Processes, Messages and Dependencies

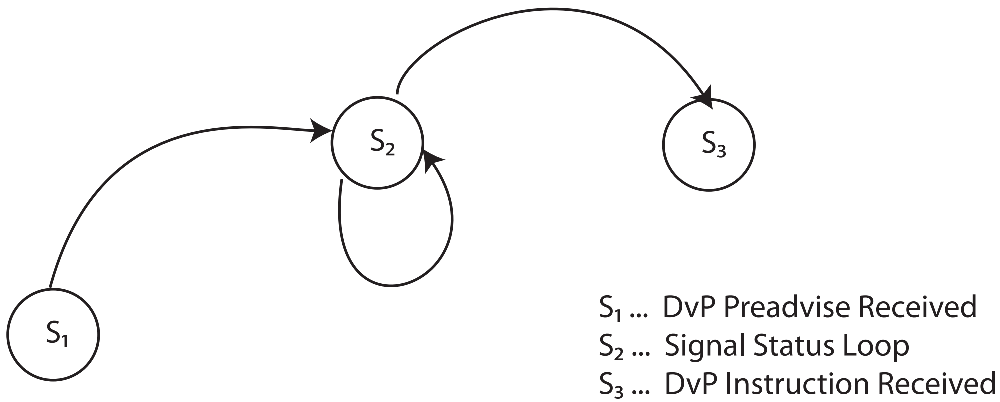


# Process Segment Modeling III



✔ Step 3: Modeling of the State Machine(s)

✘ e.g. Statecharts




✔ Step 4: Algebraic Model: Abstract or IT

✘ State-Based to Event-Based Logic Transform

# Process Algebras

# Heritage



- ✔ Petri Nets [Petri, 1962]
  - ✘ Graph-based Concurrency
- ✔ Trace Theory - Trace Sets [Mazurkiewicz, 1970]
- ✔ Semantic Logic of Computer Programs
- ✔ CCS [Millner, 1980], followed by:
  - ✘ $\pi$- Calculus
  - ✘ LOTOS
  - ✘ Communicating Sequential Processes (CSP) et al.
  - ✘ Presently: Many Variations Including Stochastic, Timed and Mobile Calculi

# Related Approaches



- ✔ Integrated Definition Methods (IDEF)
- ✔ Business Process Modeling (BPM)
  - ✘ Unified Modelling Language - Activity Diagrams (UML)
  - ✘ Business Process Modeling Language (BPML)
  - ✘ XPDL/WfXML, BPEL, XLANG et al.
- ✔ Petri Nets
- ✔ Simulation Modeling
- ✔ Model Driven Architecture (MDA)

# Calculus of Communicating Systems (CCS)



- ✔ Proposed by Milner, 1980 [4]
- ✔ Limited Set of Primitives/Constructs
  - ✘ Abstraction of Communications of Concurrent Systems
  - ✘ Agents, Actions, Choice, Parallel Composition, Restriction
- ✔ Insufficient Concreteness for Modeling Payment Systems
  - ✘ No Value-Passing in Default Specification

# $\pi$- Calculus



- ✔ Developed by [Milner 1993] as follow-up to CCS
- ✔ Base Set of Constructs such as Process, Channel, Message
- ✔ Many Calculi and Languages Derive from $\pi$-Calculus
  - ✘ e.g. BPML, occam-pi
- ✔ Sample on DvP Statemachines:
  - ✘ $S_2$ Loop Process receiving DvP Instruction Message on a Channel $m$: $m(MT5x).P$

# Communicating Sequential Processes (CSP)



- ✔ Expressive Process Algebra Introduced by Hoare, 1978 [3]
  - ✘ Formalizes Processes, Events, Traces, Multiple Parallel/Choice Operators, Hiding, Deterministic/Nondeterministic Choice etc.
- ✔ Failures/Divergences Model Synergistic to Model Checking
- ✔ Selected Basis for Timed Clearing/Settlement Extensions

# Timed Process Algebra



- ✔ Time Constructs for Calculi as Extensions or Native Algebrae
- ✔ Continous and Discrete Time
- ✔ Absolute Time Sources and Relative Time
- ✔ Main Branches
  - ✘ Timed CSP [Reed and Roscoe, 1986]
  - ✘ CSP +T
  - ✘ Timed and Temporal CCS
  - ✘ $\text{ACP}_{\varphi}$, $\text{ACP}_{dat}$
  - ✘ $\varphi$ SDL

# Program Transformation

# Computer Code from Process Algebras



- ✔ Process Algebras are Turing Complete
  - ✘ Calculi can be Tranformed in Executable Computer Code
- ✔ Language Based on CSP: OCCAM
- ✔ Transformation is Complex
  - ✘ Calcui are Precise, yet not Congruent with Mainstream Languages such as C++
- ✔ Paradigm Proposed by Co-Investigator:
  - ✘ Selective Formalsim [1]
  - ✘ CSP++ Software Synthesis Framework

# Model Checking

# Overview



- ✔ Checking Very Large State Spaces through:
  - ✘ Assertions on Properties According to a Specification (no Magic)
- ✔ CSP Synergistic Model Checker FDR2 [Formal Systems Europe, 2005]. Main Mode:
  - ✘ Failures-Divergences Refinement
  - ✘ Basic Idea: Events Occuring when Exploring the State Space must also be *Possible* to Occur by the Specification [2]

# Sample Process



- ✔ Modified for the Context of Payment Systems based on [2]:
  - ✘ Ingress: 4 Message Sending Processes Sharing a Single Data Channel Interleaved:

$$INGRESS = \underset{i \in 1 \ldots N}{|||} S_i$$

$$EGRESS = \underset{i \in 1 \ldots N}{|||} R_i$$

# Sample Process II



$$LHS = (INGRESS \underset{X}{||} (SM \,|||\, RA)) \backslash X$$

$$RHS = (EGRESS \underset{Y}{||} (RM \,|||\, SA)) \backslash Y$$

with:

$X = \{|mux,\ admx|\}$ $Y = \{|dmx, amux|\}$

taken together being:

$$SYSTEM = (LHS \underset{Z}{||} RHS) \backslash Z$$

with $Z = \{|mess, ack|\}$

Processes $\{SM,\ RA,\ RM,\ SA\}$, Channels $\{mux,\ admx,\ dmx,\ amux\}$

# Sample Process III



✔ Assertion $SPEC \sqsubseteq SYSTEM$

Excerpt from the CSPm Machine-Readable Model Checker Source:

```
Copy(i)=left.i ? x->right.i ! x->Copy(i)
Spec = ||| i:Tag @ Copy(i)

assert Spec [FD= System
```

# Sample Process IV



FDR Model Checker Run:

```
Refinement check:
Refine checked 1,404 states
With 4056 transitions
Took 0(0+0) seconds
Allocated a total of 6 pages of size 128K
Compaction produced 0 chunks of 16K.
true
```

# Project Overview

# Project Synopsis I



- ✔ Selected Processes
  - ✘ Domestic Interbank Retail Payment
  - ✘ Deferred Net Settlement
- ✔ 2 Year Research Project
  - ✘ Partnership Dept. of Computing and Info Sciences, U.of Guelph,
  - ✘ Dept. of Information Systems and Process Management, WU Wien

# Project Synopsis II



- ✔ Research Questions on Modeling Time Aspects of Clearing/Settlement
- ✔ Grant by WU Wien
- ✔ Supported by an Austrian Bank
- ✔ Outputs of Student Theses and Select Payment/Clearing/Settlement Applications

# Selected References

# Selected References



[1] William B. Gardner. Converging csp specifications and c++ programming via selective formalism. *Trans. on Embedded Computing Sys.*, 4(2):302–330, 2005.

[2] M. Goldsmith. *FDR2 User's Manual version 2.82*, June 2005.

[3] C. A. R. Hoare. Communicating sequential processes. *Commun. ACM*, 21(8):666–677, August 1978.

[4] R. Milner. *Communication and concurrency*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.