When 1 Clock Is Not Enough

Gerald Lüttgen *(University of York, UK)*

www.cs.york.ac.uk/~luettgen/

Joint work with Michael Mendler *(Bamberg University, D)*

Clocks & Multiple Clocks

Clocks as an abstract synchronisation concept:

- Orchestrating computation within its scope into a well-defined sequence of successive phases
- Qualitative rather than quantitative, where time passes *between* (and not *on*) clock ticks

Why multiple clocks?

- Spatial distribution of system components
  - Systems-on-chip: accuracy of clock signal
  - GALS: loose synchronisation between remote processes
- Complex coordination between system components
  - Scheduling in DSP: phase-based, nesting & encapsulation
  - Timing constraints: each clock implementing one constraint

Literature on Multi-Clock Process Algebras

- Is relatively scarce given the practical relevance of multi-clock settings
  - But rich literature on timed process algebras
- Extends Nicollin & Sifakis' ATP in various ways:
  - PMC - Process algebra with Multiple Clocks (ESOP '94)
  - CSA - Calculus of Synchrony and Asynchrony (CONCUR '97)
  - CaSE - Calculus of Synchrony and Encapsulation (CONCUR '03)
- Differs in choice of operators and semantics
  - Although only bisimulation-based semantics are considered
- What are the underlying design choices?

Proposed Unified Semantic Framework

A multi-clock process is an LTS

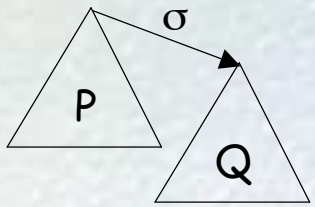
$$p = (\mathcal{A}_p, C_p, act_p, clk_p, \Sigma_p, \Pi_p) \in \mathcal{P}$$

with

- Initial actions $\mathcal{A}_p \subseteq \mathcal{A}$ (CCS-style handshake)
- Initial clocks $C_p \subseteq C$ (CSP-style broadcast)
- Action transitions act_p: $\mathcal{A}_p \rightarrow 2^{\mathcal{P}}$
- Clock transitions clk_p: $C_p \rightarrow \mathcal{P}$ (Time determinism)
- Instability set $\Sigma_p \subseteq C \backslash C_p$ (Holding up clocks)
- Urgency relation Π_p: $\mathcal{A}_p \rightarrow 2^C$ (Clock scoping)

Explicit Control Of Clock Phasing

- Cock phases entirely defined by (in)stability of state
  - Clocks stopped by default, via insistent prefixing, i.e., $\Sigma_p = C$
  - Clock σ can only tick when explicitly specified, e.g., via the ATP-style timeout operator $[P]\sigma(Q)$, i.e., $\Sigma_{[P]\sigma(Q)} = \Sigma_p \backslash \{\sigma\}$



- Urgency relation plays no role, i.e., $\Pi_p(\alpha) = \varnothing$ for all $\alpha \in \mathcal{A}_p$
- Point of view adopted in PMC

Implicit Control Of Clock Phasing

- Relaxed view on process stability
  - $\Sigma_p = \varnothing$, unless $\tau \in \mathcal{A}_p$ when $\Sigma_p = \Pi_p(\tau)$
  - Process p is stable unless it is preempted by a τ-transition (within its scope)
- Urgency relation used to implement maximal progress
  - P|Q can engage in a joint σ-transition only if there is no pending communication within σ's scope, i.e., $\sigma \notin \Pi_p(a) \cap \Pi_q(\bar{a})$ for any a
  - For calculi with a single clock, such as Hennessy & Regan's TPL, this is the classic form of *global* maximal progress
  - For calculi with multiple clocks with clock scoping, this is the refined form of *local* maximal progress
- Point of view adopted in CSA

Mixed Implicit & Explicit Control

- Implicit control via relaxed process stability and maximal progress
- Explicit control via dynamic time-stop operators Δ_σ
  - Operator Δ_σ stops clock σ in current state p, i.e., adds σ to Σ_p
- Point of view adopted in CaSE
- System verification:
  - A time-stopped process often indicates the violation of a real-time constraint

Controlling Action Urgency

How to decouple a clock σ within P, when composing P with Q to form P|Q, i.e., how to define the scope of clocks?

Three possible ways:

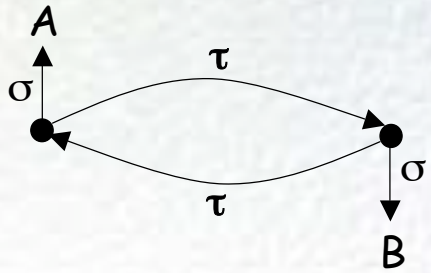
1. Detach σ from Q, using the ignore operator $\uparrow\sigma$, i.e., P | (Q $\uparrow\sigma$)
   - Override σ-transitions in Q
   - Add σ-loops at every reachable state of Q
   - Adopted in PMC and CSA
2. Attach σ to P, using the attach operator @σ, i.e., (P@σ) | Q
   - Adopted in CSA@
3. Hide σ in P, using a hiding operator
   - (P<σ>) | Q : <σ> turns σ-transitions into non-synchronising, non-urgent transitions (adopted in CSA@)
   - (P/σ) | Q : /σ turns σ-transitions into non-synchronising, but urgent transitions (adopted in CaSE)

Conclusions

- The proposed unified framework:
  - Captures the "design space" for multi-clock process algebras
  - Is able to express the existing multi-clock process algebras
- Technical results:
  - A complete axiomatisation of strong bisimulation for *regular* processes in PMC and CSA
  - Fully-abstract characterisation of observational congruence in PMC, CSA and CaSE
  - A complete axiomatisation of observational congruence for *finite* processes in PMC

Future Work

- Detail the sketched unified approach
  - Bisimulation-based semantics
  - Applications to "real" languages, e.g., multi-clock Esterel
- Axiomatise observational congruence
  - Problem for regular processes since time determinism prevents the elimination of unguardedness



- Investigate other multi-clock semantics
  - Based on testing or failures

Thank You!

Selected references:

- H.R. Andersen and M. Mendler. An asynchronous process algebra with multiple clocks. In ESOP '94, LNCS 788, pp. 58–73.
- R. Cleaveland, G. Lüttgen, and M. Mendler. An algebraic theory of multiple clocks. In CONCUR '97, LNCS 1243, pp. 166–180.
- M. Hennessy and T. Regan. A process algebra for timed systems. Information and Computation, 117:221–239, 1995.
- M. Kick. Modelling synchrony and asynchrony with multiple clocks. Master's thesis, University of Passau, D, 1999.
- X. Nicollin and J. Sifakis. The algebra of timed processes, ATP: Theory and application. Information and Computation, 114:131–178, 1994.
- B. Norton. A Process Algebraic Theory for Synchronous Software Composition. PhD thesis, University of Sheffield, UK, 2005.
- B. Norton, G. Lüttgen, and M. Mendler. A compositional semantic theory for synchronous component-based design. In CONCUR 2003, LNCS 2761, pp. 461–476.