Efficient analysis of multi-clock synchronous specifications

V. Papailiopoulos, D. Potop-Butucaru
INRIA Paris-Rocquencourt

SYNCHRON 2010
Fréjus, France
Motivation

- functional specifications (synchronous)
- preserve correctness & determinism
- embedded implementation (real-time, distributed)

- multi-synchronous (polychronous) paradigm
- execution in cycles, multiple rates
- deterministic concurrency
- global synchronization
  - clear notion of signal absence/presence

- HW/SW
- asynchronous
- platform-specific synchronization mechanisms
- reconstruction of global synchronization
Motivation

- functional specifications (synchronous)
- preserve correctness & determinism
- embedded implementation (real-time, distributed)
- multi-synchronous (polychronous) paradigm
- execution in cycles, multiple rates
- deterministic concurrency
- global synchronization → clear notion of signal absence/presence
- HW/SW
- asynchronous
- asynchronous message passing
- platform-specific synchronization mechanisms
- reconstruction of global synchronization

Confluence [R. Milner]
Monotony [G. Kahn]
Mazurkiewicz traces
scheduling-independent latency-insensitive delay-insensitive speed-independent
Contribution

- Method of checking weak endochrony
  - application formalism: SIGNAL language subset

![Diagram]

- Minimal synchronization patterns
- WE program
- Asynchronous implementation
- Synchronization defects list
- Refinements (user-defined)
- Accept?
Outline

- Weak endochrony
  - definition
  - atomic reactions
  - signal absence constraints
- Checking weak endochrony
  - generator set
  - parallel composition
- Illustrative example
- Extensions
- Conclusion
Weakly endochronous systems

asynchronous system

weakly endochronous system

\[ s_1 \rightarrow a \rightarrow s_2 \rightarrow b \rightarrow s_3 \rightarrow b \rightarrow s_4 \rightarrow a \rightarrow s_1 \]

\[ r_1, r_2 : \text{non-contradictory transitions} \]

\[ \rightarrow \text{closure under } \lor, \land \text{ and } \setminus \]

\( (r_1 \lor r_2 = \text{values of } r_1 \text{ or values of } r_2 \)

\( r_1 \land r_2 = \text{common values of } r_1, r_2 \)

\( r_1 \setminus r_2 = \text{only values of } r_1 \)
Weakly endochronous systems

- Absence = don’t care value
  - not needed in computations

- Decisions over the value of a message (not presence/absence)
  - deterministic (scheduling-independent) implementation

- Transitions not sharing common signals are independent
  - execution in any order or simultaneously

- Finite stateless abstraction
  - SIGNAL compiler
  - free of high-level synchronization constraints
Example 1: simple absence

\[ \mathcal{P} = \left( \begin{array}{c|c} B := A \mid D := C \end{array} \right) \]

Possible program reactions:

- \[ r_1 = (A^\perp, B^\perp, C^\perp, D^\perp) \]
- \[ r_2 = (A^u, B^u, C^\perp, D^\perp), \ u \in D_A \]
- \[ r_3 = (A^\perp, B^\perp, C^w, D^w), \ w \in D_C \]
- \[ r_4 = (A^u, B^u, C^w, D^w), \ u \in D_A \text{ and } w \in D_C \]

\[ r_2, r_3 = \text{independent} \]

- no synchronization needed for the execution
- concurrent execution is possible

\[ \Rightarrow r_2, r_3 = \text{atomic reactions} \Rightarrow r_4 = r_2 \lor r_3 \]

WE system representation = unique set of atomic reactions
Example 2: constraining absence

\[ P = ( \mid B := A \mid D := C \mid A ^ \# C ) \]

Possible program reactions:
- \( r_1 = (A^\perp, B^\perp, C^\perp, D^\perp) \)
- \( r_2 = (A^u, B^u, C^\perp, D^\perp), \ u \in D_A \)
- \( r_3 = (A^\perp, B^\perp, C^w, D^w), \ w \in D_C \)

\( r_2, r_3 = \text{not independent} \)
- synchronizations are necessary

\( \Rightarrow \) explicit absence: \( \perp \)
- special synchronization message
Example 2: constraining absence

- $P = (B := A \mid D := C \mid A \wedge \neg C)$

Possible reactions:
- $r_1 = (A^\perp, B^\perp, C^\perp, D^\perp)$
- $r_2 = (A^u, B^u, C^\perp, D^\perp), \ u \in D_A$
- $r_3 = (A^\perp, B^\perp, C^w, D^w), \ w \in D_C$

Absence constraints must be preserved

No single solution

Representation by generators (atoms + absence constraints)
- “compact” representation
Generator set (GS)

- Minimal set that contains all possible reactions of a multi-clock program
- For a program P with a set of reactions R,
  - (generation) GS generates R by union
  - (non-interference) if $g_1, g_2 \in GS$ have common present values and they are not different (no conflict), then $g_1 = g_2$
  - (non-void) no $g \in GS$ is exclusively composed of $\bot, \perp$
  - (least synchronized) no constraining absence($\perp$) in $g \in GS$ can be replaced by a simple absence ($\bot$)
    - discover as much concurrency as possible to reduce the need for synchronization
Computing the generator set of a program

- Library of generator sets for basic SIGNAL operators (dataflow operators + constraints)
- Algorithm for computing the generators of a parallel composition \((GS_p, GS_q \rightarrow GS_{p|q})\)
  1. Compute \(GS’_{p|q}\) (not necessarily minimal)
     - Match generators of \(GS_p\), \(GS_q\) by combining their common parts (correspondence problem)
  2. Remove useless \(\bot\) values from \(GS’_{p|q}\)
     - (potentially) discover concurrency
Parallel composition

\[ p = (| X := Y \mid Z := T |) \]
\[ q = (| X^\# Z |) \]
\[ GS_p = \{ (X^u, Y^u) \mid u \in D_X \} \cup \{ (Z^u, T^u) \mid u \in D_Z \} \]
\[ GS_q = \{ (X^u, Z^\perp) \mid u \in D_X \} \cup \{ (X^\perp, Z^u) \mid u \in D_Z \} \cup \{ (Y^u) \mid u \in D_Y \} \cup \{ (T^u) \mid u \in D_T \} \]
\[ GS_{p|q} = \{ (X^u, Y^u, Z^\perp, T^\perp) \mid u \in D_X \} \cup \{ (X^\perp, Y^\perp, Z^u, T^u) \mid u \in D_Z \} \]
Checking weak endochrony

- Compute the GS for a synchronous program P
  - bottom-up

- Weak endochrony test:
  \[ \forall g_1, g_2 \in \text{GS signal absence is not needed to distinguish } g_1 \text{ from } g_2 \]
  \[ \iff \]
  P is weakly endochronous

- If P is weakly endochronous
  - asynchronous implementation is possible

- If not
  - existing \[\downarrow\] values provide a minimal synchronization solution
Illustrative example

- Two adders
  - synchronized, when \( \text{sync} = \text{present} \)
  - independent, when \( \text{sync} = \text{absent} \)

```
process adder =
  (? integer \( i_{11}, i_{12}, i_{21}, i_{22} \); event \( \text{sync} \)
  ! integer \( o_1, o_2 \))
  ( (o_1, c) := ADD1(\( i_{11}, i_{12}, \text{sync} \))
    | o_2 := ADD2(\( i_{21}, i_{22}, \text{sync}, c \))
  )
where
  boolean c;
  process ADD1 = ... ;
  process ADD2 = ... ;
end;
```
Illustrative example

- Generator set computation

```plaintext
process adder = (? event sync, integer i_1, i_2;
  ! integer o_1, o_2)
  (i_1 = o_1 ⇒ GS_{i_1=\text{true}}
   sync < i_1 ⇒ GS_{-sync<\text{false}_1}
   i_2 = o_2 ⇒ GS_{i_2=\text{true}_2}
   sync < i_2 ⇒ GS_{sync<\text{false}_2}
   c = sync ⇒ GS_{c=\text{true}}
  )
where
  boolean c
end;
```
Illustrative example

- **Generator set**
  - \( r_1 = (i_1, o_1, \text{sync}=\perp) \)
  - \( r_2 = (i_2, o_2, \text{sync}=\perp) \)
  - \( r_3 = (i_1, o_1, \text{sync}, c, i_2, o_2) \)

- **Introduce new input signals**
  - boolean \( \text{sync}_1 \) at ADD1
    - \( \text{sync}_1=0 \) : no synchronization needed
    - \( \text{sync}_1=1 \) : ADD1 and ADD2
  - boolean \( \text{sync}_2 \) at ADD2
    - \( \text{sync}_2=0 \) : no synchronization needed
    - \( \text{sync}_2=1 \) : ADD1 and ADD2
  - \( \text{sync}_1=1 \land \text{sync}_2=1 \iff \text{sync}=\text{present} \)
Prototype V1

1. **Stateless SIGNAL program**
2. **Minimal synchronization patterns**
3. **WE?**
   - yes: **WE program generators**
   - no: **Synchronization defects list**
4. **Refinements (user-defined)**
5. **Accept?**
   - yes
   - no
Extensions

Compact representation of generators

- groups of atoms with the same synchronization patterns
- for signals \( V = \{s_1, s_2, \ldots, s_n\} \), \( g = \langle u_1, \ldots, u_n \rangle, \{c_1, \ldots, c_m\} \rangle \)
  - \( u_i : \perp, \bot \) or subset of the data domain of \( s_i \)
  - \( c_i : \) constraint over the signal values (\( c_i = (1=2) \rightarrow u_1 = u_2 \))
- no empty data domain
- only equality constraints (for the moment)
Extensions

Example:

- $G_{S_x:=y \text{default}_z} = \{(X^D, Y^D, Z^\perp, W^\perp), \{1=2\}\}$,
  $\{(X^D, Y^\perp, Z^D, W^\perp), \{1=3\}\}$,
  $\{(X^\perp, Y^\perp, Z^\perp, W^{D'})\}$

New definition: $g_1 \vee g_2$

- domain intersection of the signals
- union of the constraints

New algorithms
Work in progress

- SIGNAL program
- stateless SIGNAL program
- minimal synchronization patterns
- WE?
- WE program generators
- asynchronous implementation
- refinements (user-defined)
- synchronization defects list
- accept?

Yes: WE program generators
No: refinements (user-defined)
Work in progress

- Optimization of generator sets
  - hierarchical and symbolic representation
    - decision trees

- $t_i \rightarrow \text{leaves}$
Conclusion

- Method of analyzing synchronous programs
  - compact representation
  - take advantage of the potential concurrency

- Weak endochrony criterion
  - ensure asynchronous implementation
  - efficient code generation

- Future work
  - symbolic representation
  - evaluation on real SIGNAL programs
Finite-data stateless abstraction

- Finite data types
- No delay equations
  - $X:=Y \ $init \ v_0 \rightarrow X^=Y$
- Clock constraints
  - Equality: $X^=Y$
  - Inclusion: $X^<Y$
  - Exclusiveness: $X^\#Y$
Checking weak endochrony

- Compute the synchronization configurations
- Construct minimal sets
  - Generator sets
- Check the weak endochrony properties
  - require that all choices are over input signal values
Basic generator sets

- $V = \{X, Y, Z, W\}$

- $G_{X:=Y_{\text{default}Z}} = \{(X^D, Y^D, Z^\perp, W^\perp), \{1=2\}\}$,
  \[(X^U, Y^\parallel, Z^u, W^\perp), \{1=3\}\],
  \[(X^\perp, Y^\perp, Z^\perp, W^D)\}\}

- $G_{X:=Y_{\text{when}Z}} = \{(X^D, Y^D, Z^{[1]}, W^\perp), \{1=2\}\}$,
  \[(X^\parallel, Y^D, Z^{[0]}, W^\perp)\],
  \[(X^\parallel, Y^D, Z^\parallel, W^\perp)\],
  \[(X^\perp, Y^\perp, Z^\perp, W^D)\}\}

- $G_{X^\wedge Y} = \{(X^D, Y^D, Z^\perp, W^\perp), (X^\perp, Y^\perp, Z^D, W^\perp), (X^\perp, Y^\perp, Z^\perp, W^D)\}$

- $G_{X^\wedge Y} = \{(X^U, Y^D, Z^\perp, W^\perp), (X^\parallel, Y^D, Z^\perp, W^\perp), (X^\perp, Y^\perp, Z^D, W^\perp)\}$
  \[(X^\perp, Y^\perp, Z^\perp, W^D)\]\}
Illustrative example

Two adders
- synchronized, when $\text{sync}=$present
- independent, when $\text{sync}=$absent

```
process ADD1 =
  (? integer $i_1$, $i_2$; event sync
  ! integer $o$; boolean $c$)
  ( $i_1$^=$i_2
    sync^<$i_1
    c^=sync
    $o$=$((i_1$ when sync) + ($i_2$ when sync)
  )
  )

process ADD2 =
  (? integer $i_1$, $i_2$; event sync ; boolean $c$
  ! integer $o$)
  ( $i_1$^=$i_2
    sync^<$i_1
    c^=sync
    $o$=($i_1$ when sync) + ($i_2$ when sync) + $c$
  )
  default ($i_1$+$i_2$)
) ;
```