Efficient analysis of multi-clock synchronous specifications

V. Papailiopoulou, 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

weakly endochronous program

embedded implementation (real-time, distributed)

- multi-synchronous (polychronous) paradigm
- execution in cycles, multiple rates
- deterministic concurrency

- global synchronization
  → clear notion of signal absence/presence

Confluence [R. Milner]
Monotony [G. Kahn]
Mazurkiewicz traces

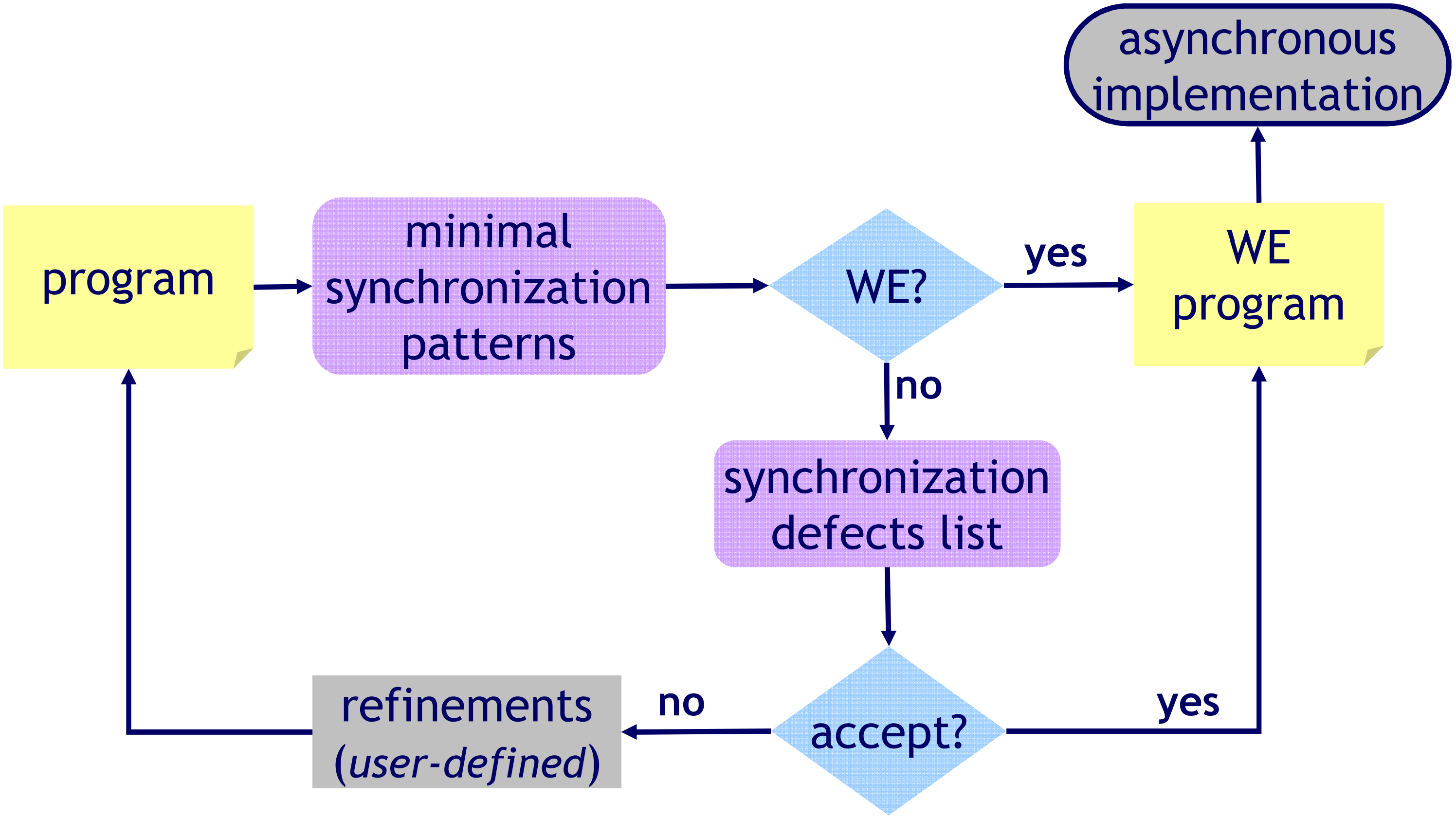
scheduling-independent
latency-insensitive
delay-insensitive
speed-independent

- HW/SW
- asynchronous
- asynchronous message passing

- platform-specific synchronization mechanisms
- reconstruction of global synchronization

Contribution

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

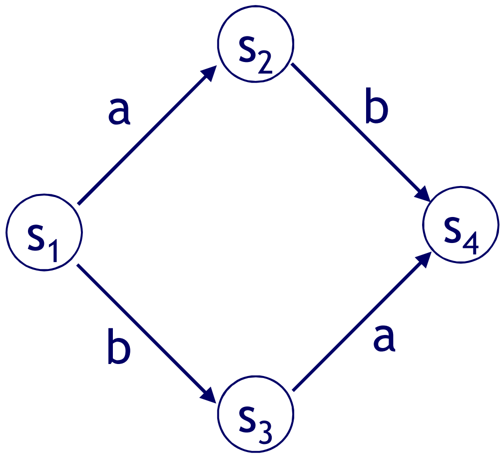


Outline

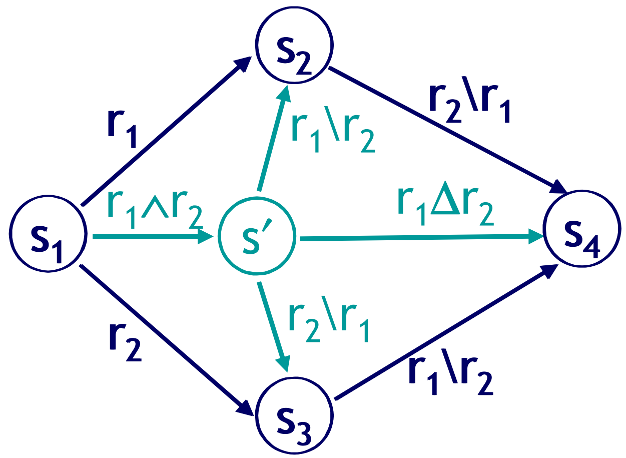
- Weak endochrony
  - definition
  - atomic reactions
  - signal absence constraints
- Checking weak endochrony
  - generator set
  - parallel composition
- Illustrative example
- Extensions
- Conclusion

Weakly endochronous systems

asynchronous system



weakly endochronous system



r_1, r_2 : non-contradictory transitions

→ closure under $\vee$, $\wedge$ and \

($r_1 \vee r_2$ = values of r_1 or values of r_2

$r_1 \wedge r_2$ = common values of r_1, r_2

$r_1\backslash r_2$ = only values of r_1)

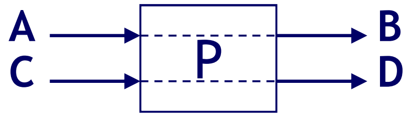
Weakly endochronous systems

- 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

- P=(|B:=A|D:=C|)



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

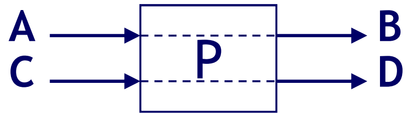
  AS(P)

  - $r_4 = (A^u, B^u, C^w, D^w)$, $u \in D_A$ and $w \in D_C$
- r_2, r_3 = independent
  - no synchronization needed for the execution
  - concurrent execution is possible

$\Rightarrow$ r_2, r_3 = *atomic reactions* $\rightarrow$ $r_4 = r_2 \vee r_3$

WE system representation = **unique** set of atomic reactions

Example 2: constraining absence



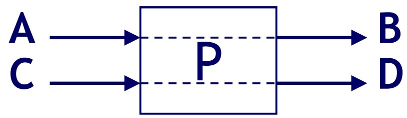
- P=(|B:=A|D:=C|A^#C|)
- Possible program reactions:
  - $r_1 = (A^{\perp}, B^{\perp}, C^{\perp}, D^{\perp})$
  - $r_2 = (A^{u}, B^{u}, C^{\perp}, D^{\perp})$, $u \in D_A$
  - $r_3 = (A^{\perp}, B^{\perp}, C^{w}, D^{w})$, $w \in D_C$
- r_2, r_3 = not independent
  - synchronizations are necessary

$\Rightarrow$ explicit absence: $\perp\!\!\!\perp$

- special synchronization message

Example 2: constraining absence

- P=(|B:=A|D:=C|A^#C|)



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

$$\begin{cases} r'_2 = (A^{u}, B^{u}, C^{\perp\!\!\!\perp}, D^{\perp\!\!\!\perp}),\ u \in D_A \\ r'_3 = (A^{\perp\!\!\!\perp}, B^{\perp\!\!\!\perp}, C^{w}, D^{w}),\ w \in D_C \end{cases}$$

$$\begin{cases} r''_2 = (A^{u}, B^{u}, C^{\perp}, D^{\perp}),\ u \in D_A \\ r''_3 = (A^{\perp\!\!\!\perp}, B^{\perp}, C^{w}, D^{w}),\ w \in D_C \end{cases}$$

- 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 $\perp$, $\perp\!\!\perp$
  - (least synchronized) no constraining absence($\perp\!\!\perp$) in $g \in$ GS can be replaced by a simple absence ($\perp$)
    - 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 → $GS_{p|q}$)
  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 $\perp\!\!\!\perp$ values from $GS'_{p|q}$
     - (potentially) discover concurrency

INRIA

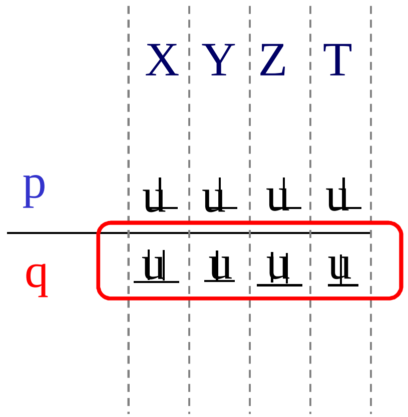
Parallel composition

p = (| X:=Y | 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

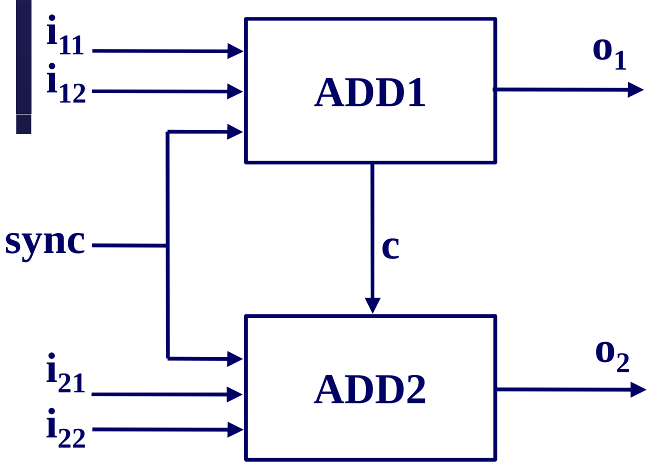
- Compute the GS for a synchronous program P
  - bottom-up
- Weak endochrony test:

$$\forall g_1, g_2 \in GS \text{ signal absence is not needed to distinguish } g_1 \text{ from } g_2$$
$$\Leftrightarrow$$
$$P \text{ is weakly endochronous}$$

- If P is weakly endochronous
  - asynchronous implementation is possible
- If not
  - existing $\perp\!\!\!\perp$ values provide a minimal synchronization solution

Illustrative example

- Two adders
  - synchronized, when sync=present
  - independent, when sync=absent

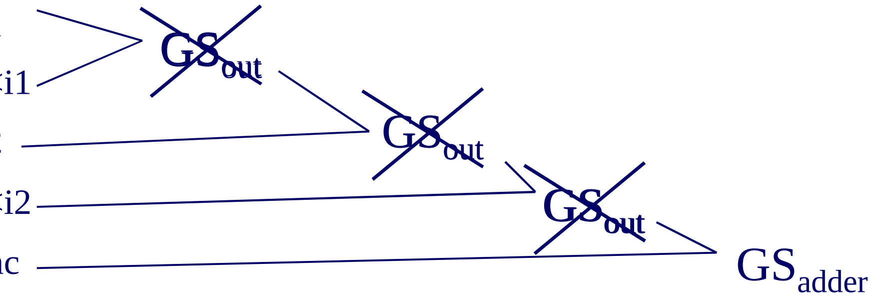



```
process adder =
(? integer i11,i12,i21,i22; event sync
 ! integer o1,o2)
(| (o1,c):=ADD1(i11,i12,sync)
 | o2:=ADD2(i21,i22,sync,c)
 |)
where
  boolean c;
  process ADD1 = … ;
  process ADD2 = … ;
end;
```

Illustrative example

- Generator set computation

```
process adder = (? event sync, integer i1,i2;
 ! integer o1,o2)
(| i1 ^= o1       → GS_i1^=o1
 | sync ^< i1     → GS_sync^<i1
 | i2 ^= o2       → GS_i2^=o2
 | sync ^< i2     → GS_sync^<i2
 | c ^= sync      → GS_c^=sync
 |)
where
  boolean c
end;
```



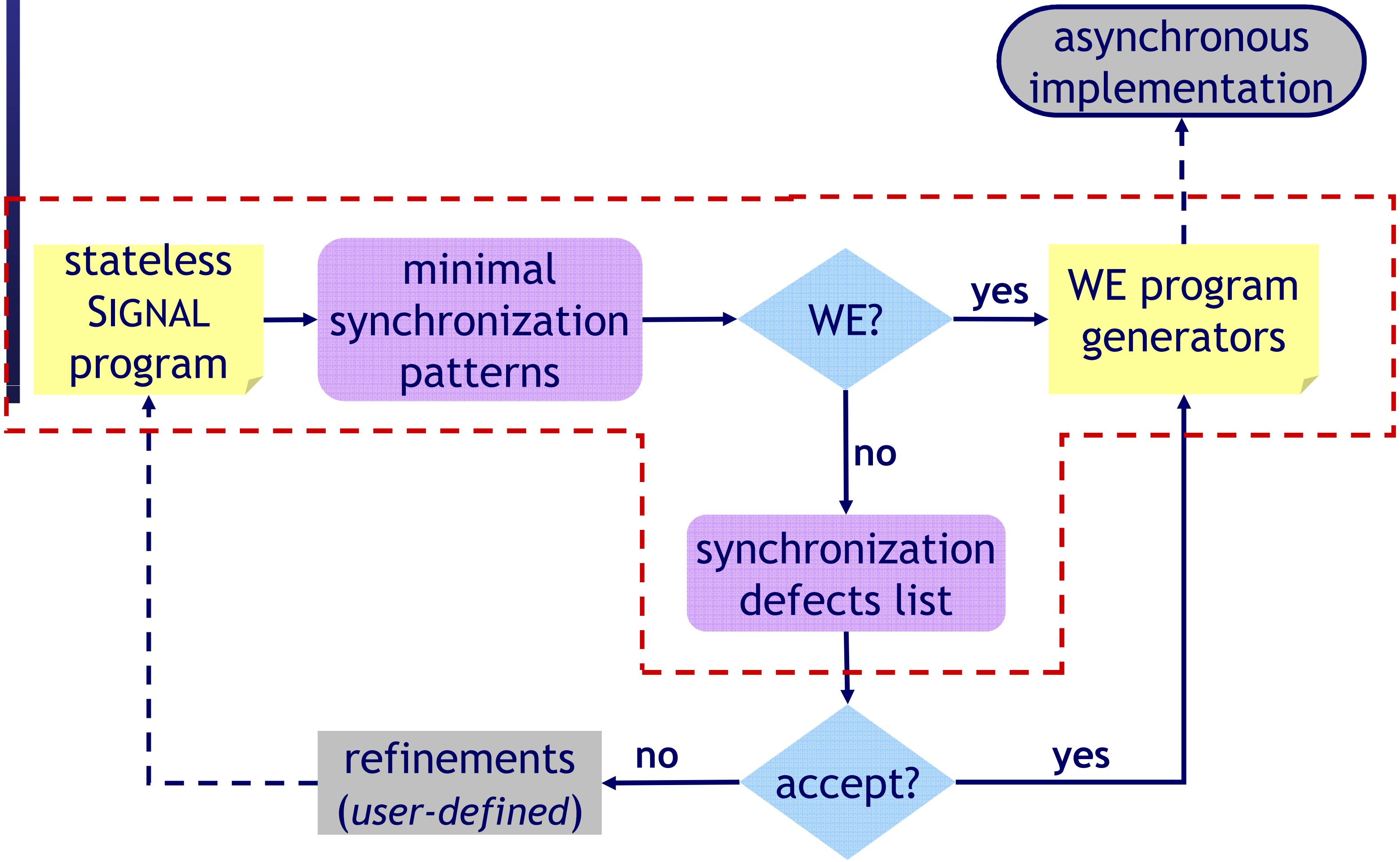
Illustrative example

- Generator set
  - [ADD1] $r_1 = (i_1, o_1, sync{=}\perp\!\!\!\perp)$
  - [ADD2] $r_2 = (i_2, o_2, sync{=}\perp\!\!\!\perp)$
  - [ADD1+ADD2] $r_3 = (i_1, o_1, sync, c, i_2, o_2)$

not weakly endochronous

- Introduce new input signals
  - boolean sync1 at ADD1
    - sync1=0 : no synchronization needed
    - sync1=1 : ADD1 and ADD2
  - boolean sync2 at ADD2
    - sync2=0 : no synchronization needed
    - sync2=1 : ADD1 and ADD2
  - $sync1{=}1 \wedge sync2{=}1 \Leftrightarrow sync{=}present$

Prototype V1



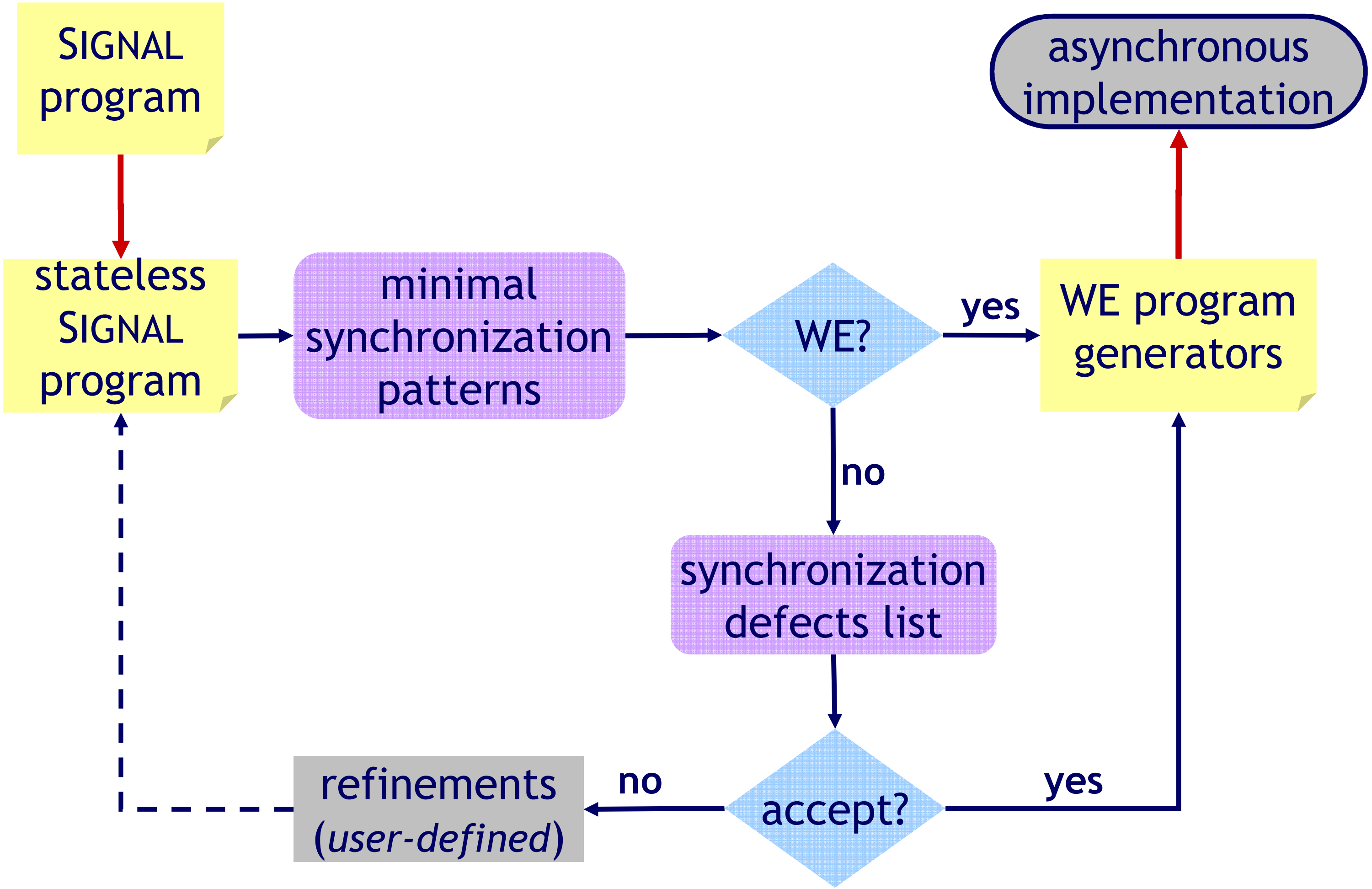
Extensions

- Compact representation of generators
  - groups of atoms with the same synchronization patterns
  - for signals $V=\{s_1,s_2,\ldots,s_n\}$, $g = \langle(u_1,\ldots,u_n), \{c_1,\ldots,c_m\}\rangle$
    - u_i : $\perp$, $\perp\!\!\!\perp$ 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

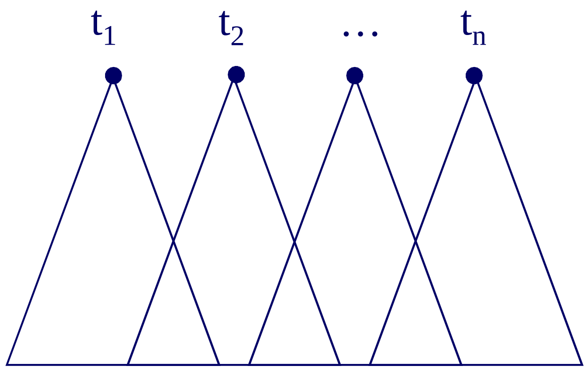
- Example:
  - $GS_{X:=Y default Z}=\{<(X^{D}, Y^{D}, Z^{\perp}, W^{\perp}), \{1=2\}>,$
    $<(X^{D}, Y^{\perp\!\!\!\perp}, Z^{D}, W^{\perp}), \{1=3\}>,$
    $<(X^{\perp}, Y^{\perp}, Z^{\perp}, W^{D'})>\}$
- New definition: $g_1 \vee g_2$
  - domain intersection of the signals
  - union of the constraints
- New algorithms

Work in progress



Work in progress

- Optimization of generator sets
  - hierarchical and symbolic representation
    - decision trees
- $t_i \rightarrow$ leaves | $\perp$



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

SIGNAL subset

- Finite-data stateless abstraction
  - Finite data types
  - No delay equations
    - `X:=Y $init v`$_0$ → `X^=Y`
  - Clock constraints
    - Equality: `X^=Y`
    - Inclusion: `X^<Y`
    - Exclusiveness: `X^#Y`

Checking weak endochrony

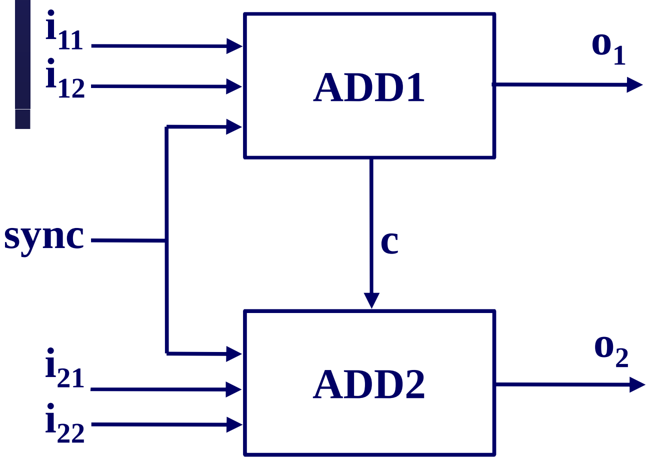
- Compute the synchronization configurations
- Construct minimal sets
  - Generator sets
- Check the weak endochrony properties
  - require that all choices are over input signal values

Basic generator sets

- $V = \{X, Y, Z, W\}$
- $G_{X:=Y\mathrm{default}Z} = \{<(X^D, Y^D, Z^\perp, W^\perp), \{1=2\}>,$
  $<(X^u, Y^{\perp\!\!\!\perp}, Z^u, W^\perp), \{1=3\}>,$
  $<(X^\perp, Y^\perp, Z^\perp, W^D)>\}$
- $G_{X:=Y\mathrm{when}Z} = \{<(X^D, Y^D, Z^{\{1\}}, W^\perp), \{1=2\}>,$
  $<(X^{\perp\!\!\!\perp}, Y^D, Z^{\{0\}}, W^\perp)>,$
  $<(X^{\perp\!\!\!\perp}, Y^D, Z^{\perp\!\!\!\perp}, W^\perp)>,$
  $<(X^\perp, Y^\perp, Z^\perp, W^D)>\}$
- $G_{X\hat{}=Y} = \{<(X^D, Y^D, Z^\perp, W^\perp)>, <(X^\perp, Y^\perp, Z^D, W^\perp)>, <(X^\perp, Y^\perp, Z^\perp, W^D)>\}$
- $G_{X\hat{}<Y} = \{<(X^u, Y^D, Z^\perp, W^\perp)>, <(X^{\perp\!\!\!\perp}, Y^D, Z^\perp, W^\perp)>, <(X^\perp, Y^\perp, Z^D, W^\perp)>,$
  $<(X^\perp, Y^\perp, Z^\perp, W^D)>\}$

Illustrative example

- Two adders
  - synchronized, when sync=present
  - independent, when sync=absent




```
process ADD1 =
(? integer i1,i2; event sync
 ! integer o; boolean c)
(| i1^=i2
 | sync^<i1
 | c^=sync
 | o:=(i1when sync) + (i2when sync)
 |) ;
process ADD2 =
(? integer i1,i2; event sync ; boolean c
 ! integer o)
(| i1^=i2
 | sync^<i1
 | c^=sync
 | o:=((i1when sync) + (i2when sync) + c)
         default (i1+i2)
 |) ;
```