Abstraction of Clocks in Synchronous Data-flow Systems

A. Cohen [1] L. Mandel [2] F. Plateau [2] M. Pouzet[23]

(1) INRIA Saclay - Ile-de-France, Orsay, France

(2) LRI, Univ. Paris-Sud 11, Orsay, France and INRIA Saclay

(3) Institut Universitaire de France

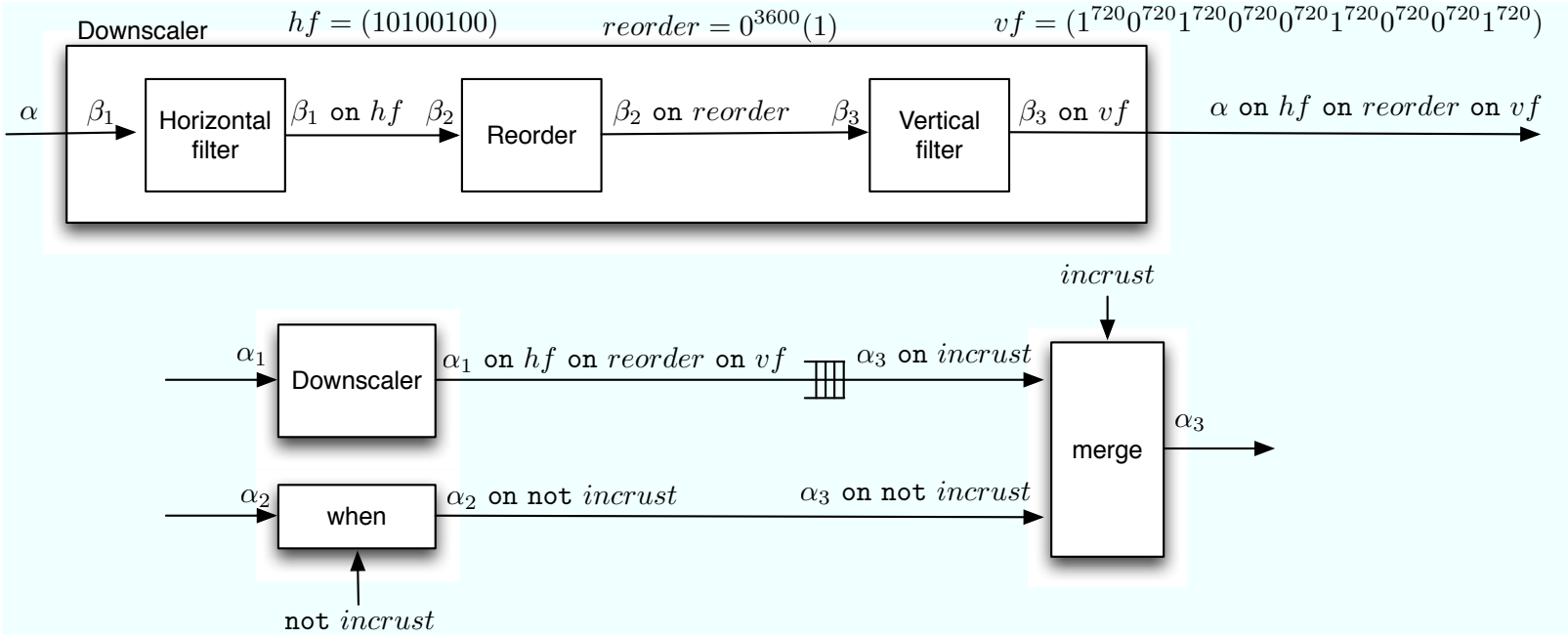
Synchron Workshop - 2 December 2008

N-Synchronous Model: A Relaxed Notion of Synchrony

- 0-synchronous : no buffers ($h_a = h_b$).
  For instance $(1010) \neq (0110)$.
- n-synchronous : buffer of size n ($h_a <: h_b$).
  For instance $(1010) <: (0110)$

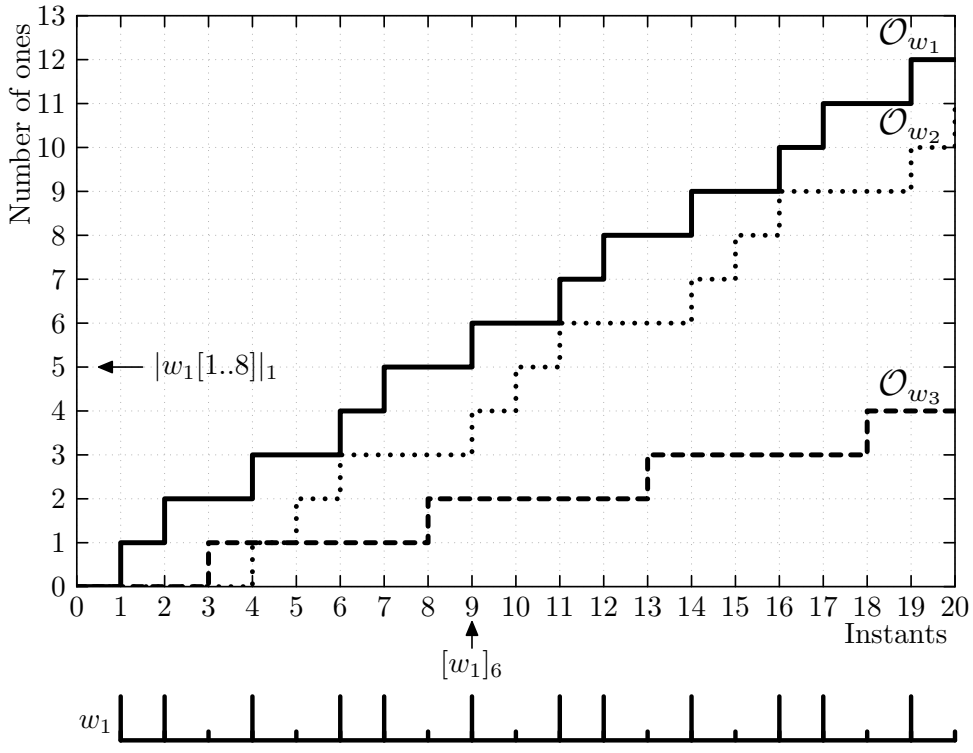
Interest:
more flexible, as much guaranties as in synchronous model.

N-Synchronous Model: picture in picture example



- Previous work: subtyping relation can be checked provided clocks are periodic.
- Motivations:
  1. dealing with long patterns in periodic clocks
  2. modelizing jitter
     How to deal with "almost periodic" clocks ? For instance
     α on w with $w = 00.(\ (10) + (01)\ )^{\omega}$
     (e.g. $w = 00\ 01\ 10\ 01\ 01\ 10\ 01\ 10\ldots$)

Clocks as Infinite Binary Words



$w ::= 0.w \mid 1.w$

Notations:

$w[i]$: element at index i

$[w]_j$: position of the j^{th} 1

$\mathcal{O}_w(i)$: number of 1s seen in w until index i.

buffer $size(w_1, w_2) = \max_{i \in \mathbb{N}}(\mathcal{O}_{w_1}(i) - \mathcal{O}_{w_2}(i))$

precedence $w_1 \preceq w_2 \overset{def}{\Leftrightarrow} \forall i,\ \mathcal{O}_{w_1}(i) \geq \mathcal{O}_{w_2}(i)$

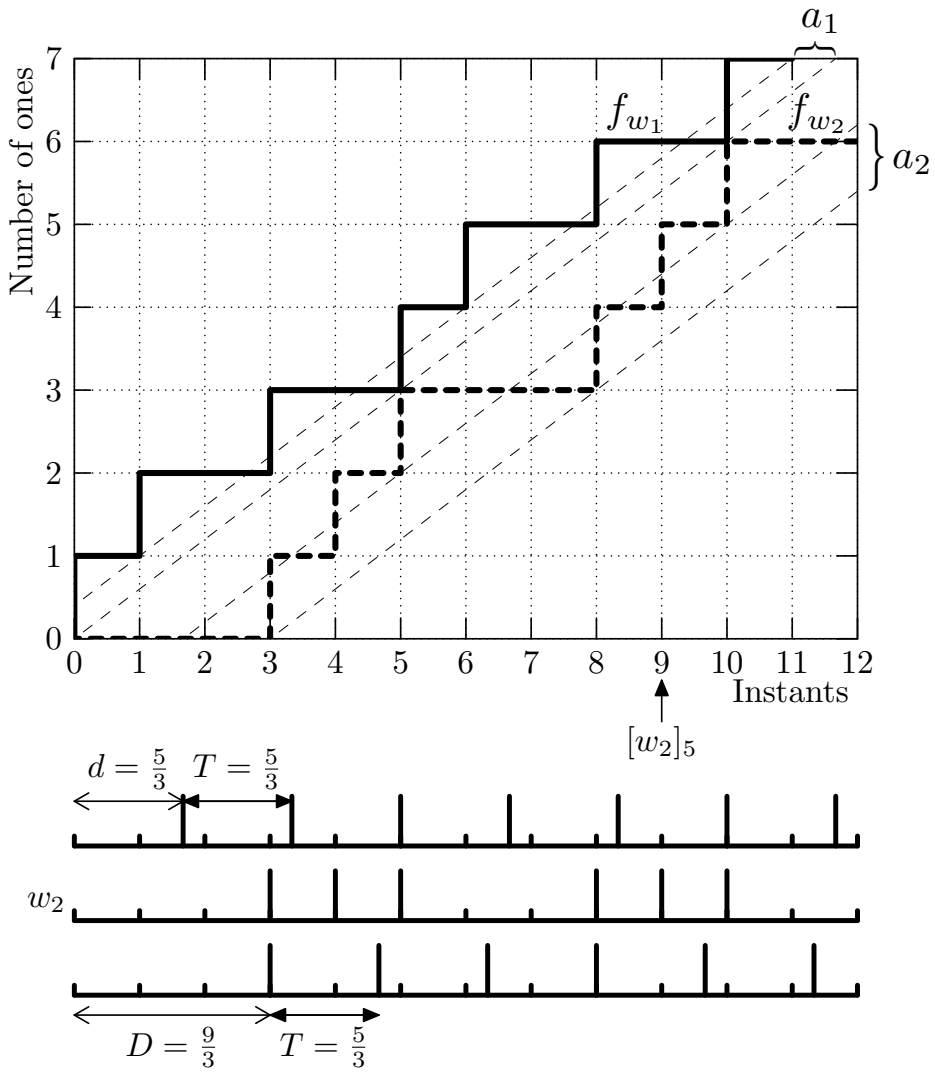
synchronizability $w_1 \bowtie w_2 \overset{def}{\Leftrightarrow} \exists b_1, b_2 \in \mathbb{Z}, \forall i,\ b_1 \leq \mathcal{O}_{w_1}(i) - \mathcal{O}_{w_2}(i) \leq b_2$

subtyping $w_1 <: w_2 \overset{def}{\Leftrightarrow} (w_1 \preceq w_2) / (w_2 \preceq w_2)$

composed clocks $c ::= w \mid not\ w \mid c\ on\ c$

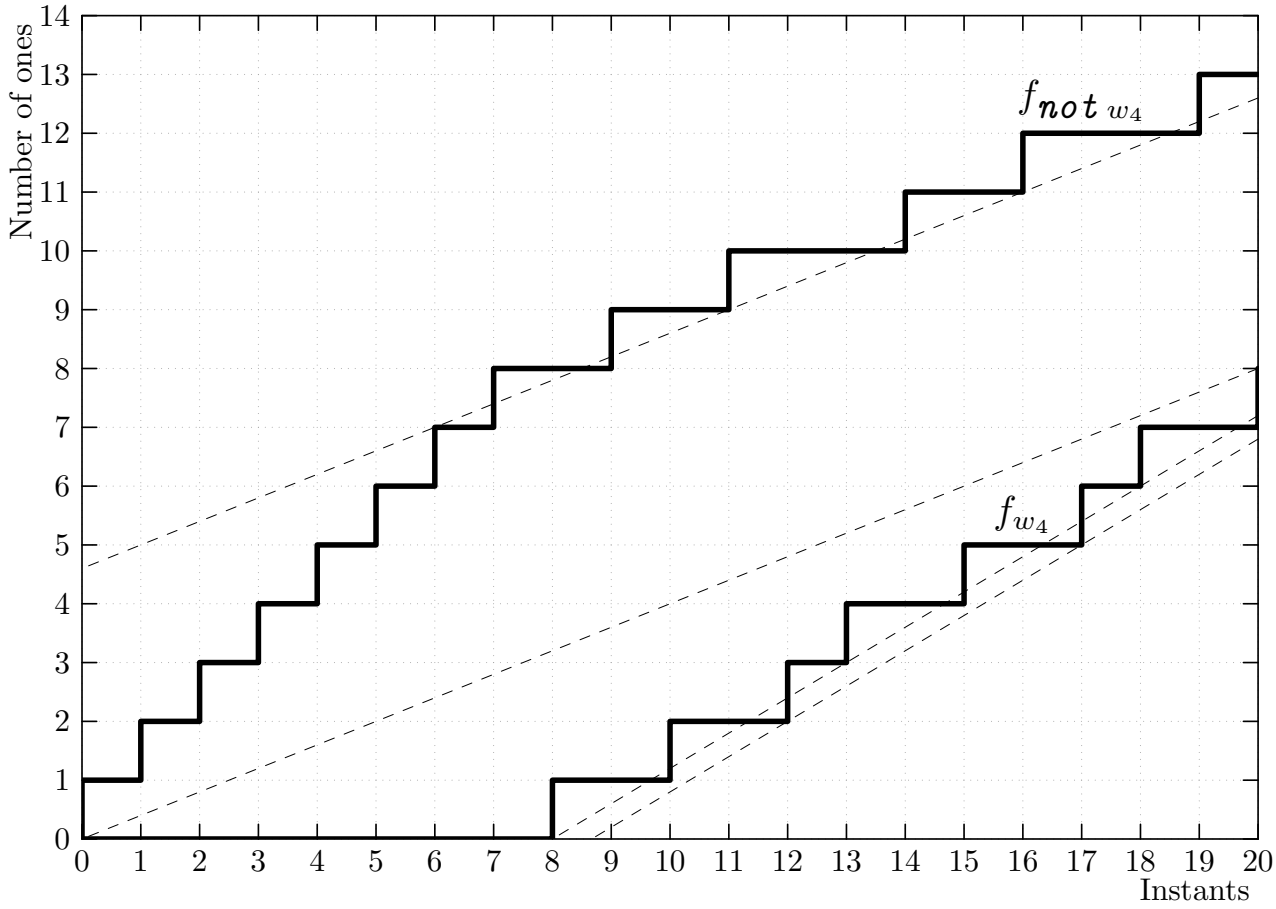
Abstraction of Infinite Binary Words [SYNCHRON'07, APLAS'08]: $abs(w) = [d, D](T)$

$$concr([d, D](T)) \stackrel{def}{\Leftrightarrow} \{w,\ \forall j \geq 0,\ T \times j + d \leq [w]_{j+1} \leq T \times j + D\}$$



Drawbacks

- sets of 1s in prefix are badly abstracted.



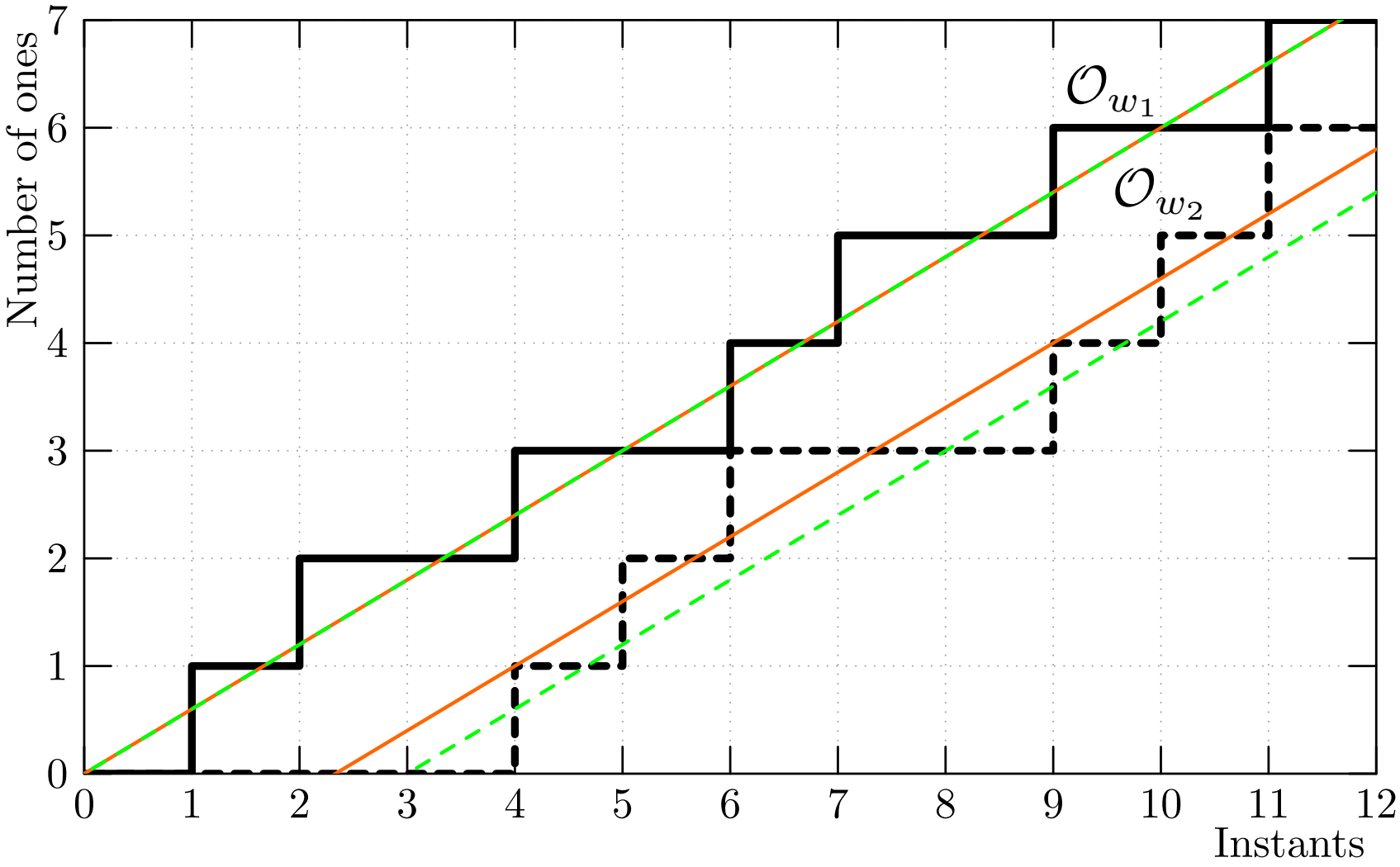
$\Rightarrow a \subseteq not^{\sim}\ not^{\sim}\ a$.

- words with finite number number of 1s cannot be abstracted.
  $\Rightarrow not^{\sim}$ can not be applied to 000(1).

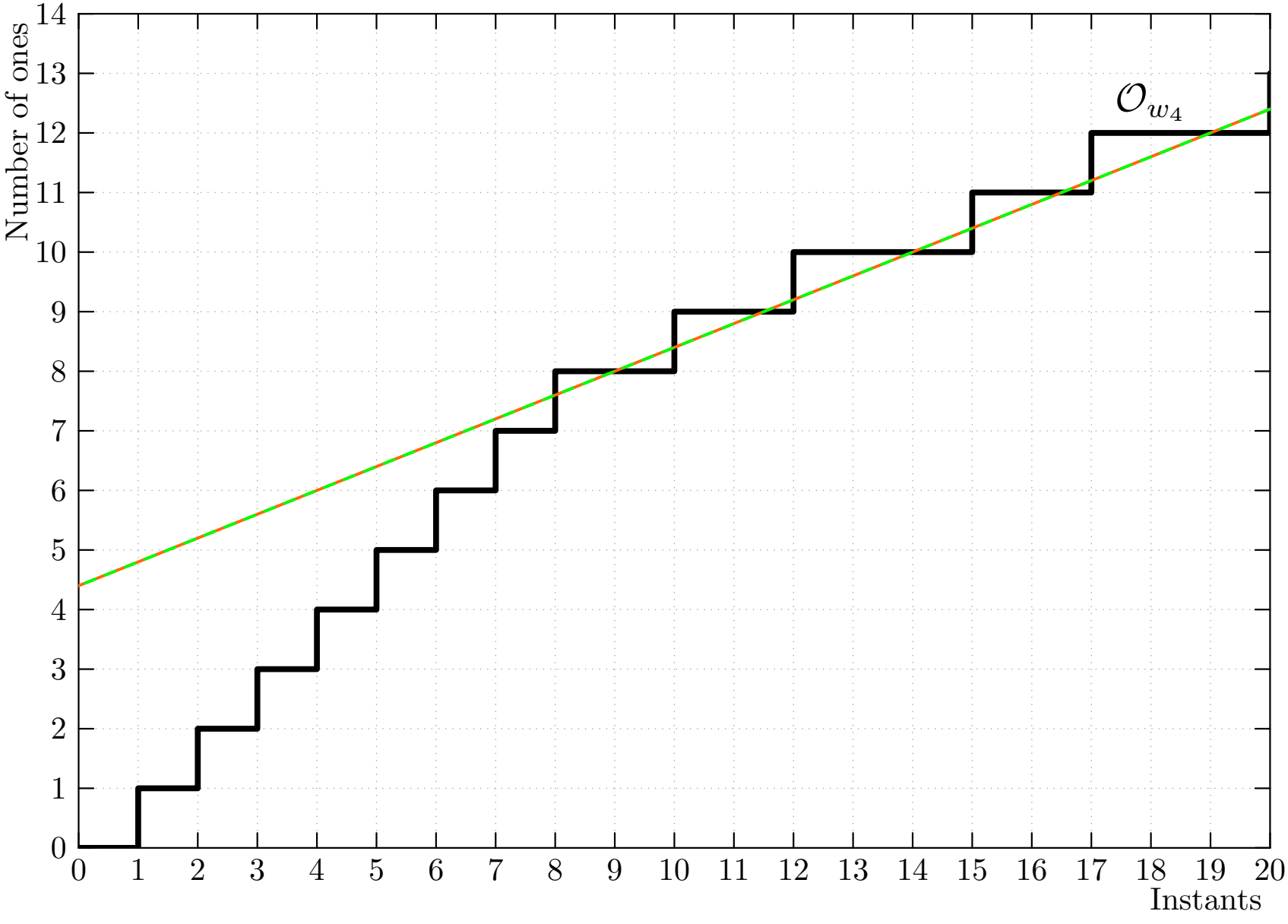
New Abstraction [JFLA'09]:

$$concr\left((b^0, b^1, r)\right) \overset{def}{\Leftrightarrow}$$

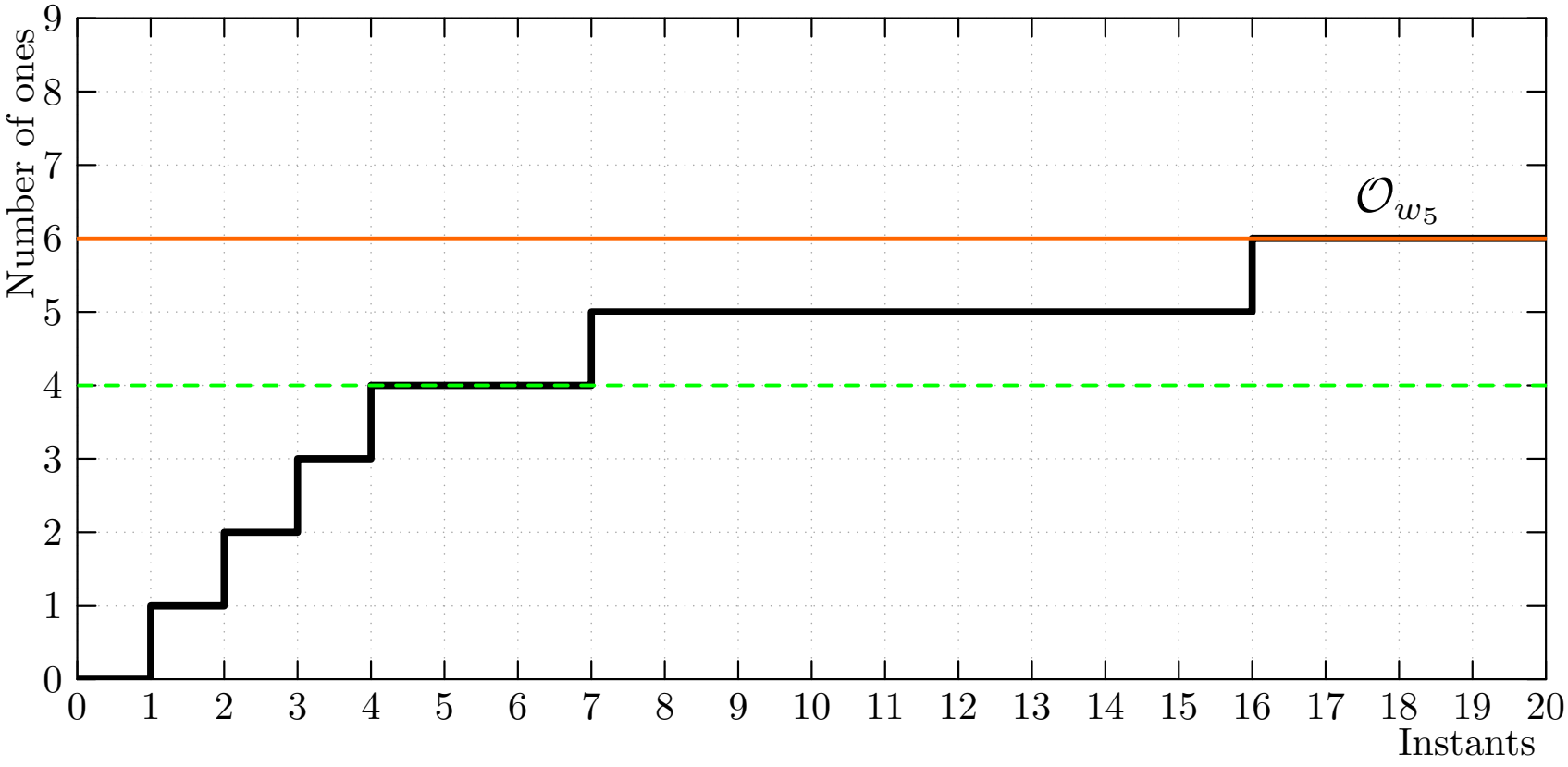
$$\left\{ w,\ \forall i \geq 1,\quad \begin{array}{lcll} w[i]=1 & \Rightarrow & \mathcal{O}_w(i-1) & < r \times i + b^1 \\ \wedge\ w[i]=0 & \Rightarrow & \mathcal{O}_w(i-1) & \geq r \times i + b^0 \end{array} \right\}$$



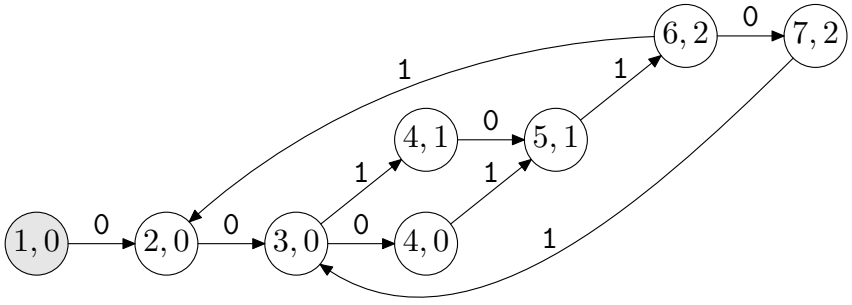
- Initial sets of $1s$ are well abstracted.



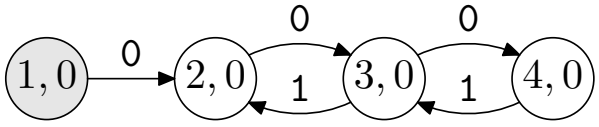
- Clocks with a nul rate can be abstracted.



Abstract Clocks as Automata



$$a_2 = \left(-\frac{9}{5}, -\frac{7}{5}, \frac{3}{5}\right)$$



$$a_4 = \left(-\frac{3}{2}, -\frac{2}{2}, \frac{1}{2}\right)$$

$A_a = \langle Q, \Sigma, \delta, q_o \rangle$ with:

- The set of states $Q = \{(i,j) \in \mathbb{N}^2\}$: coordinates in the 2D-chronograms.
  Finite number of state equivalence classes.
- The initial state q_o is $(0,0)$
- The alphabet Σ is $\{0,1\}$
- The transition function δ is defined by:
  $\delta(1,(i,j)) = nf(i+1, j+1)$ if $\mathcal{O}_w(i-1) < r \times i + b^1$
  $\delta(0,(i,j)) = nf(i+1, j)$ if $\mathcal{O}_w(i-1) \geq r \times i + b^0$
  It is undefined otherwise.

Checking Clocks are in an Abstraction

```
type rat = { num: int; den: int; }
(* "+:", "<:" ... are operations over rat *)

let norm { num = n; den = l; } i j =
  if i >= l && j >= n
  then (i - l, j - n)
  else (i, j)

let node check (b0, b1, r) clk = ok where
  rec i, j = (1,0) fby norm r (i+1) (if clk then j + 1 else j)
  and ok =
    if clk
    then (rat_of_int j) <: r *: (rat_of_int i) +: b1
    else (rat_of_int j) >=: r *: (rat_of_int i) +: b0
```

Generating Clocks in an Abstraction

```
let node generate choice (b0, b1, r) = clk where
  rec i, j = (1,0) fby norm r (i+1) (if clk then j + 1 else j)
  and one = (rat_of_int j) <: (r *: (rat_of_int i) +: b1)
  and zero = (rat_of_int j) >=: (r *: (rat_of_int i) +: b0)
  and clk = choice one zero
```

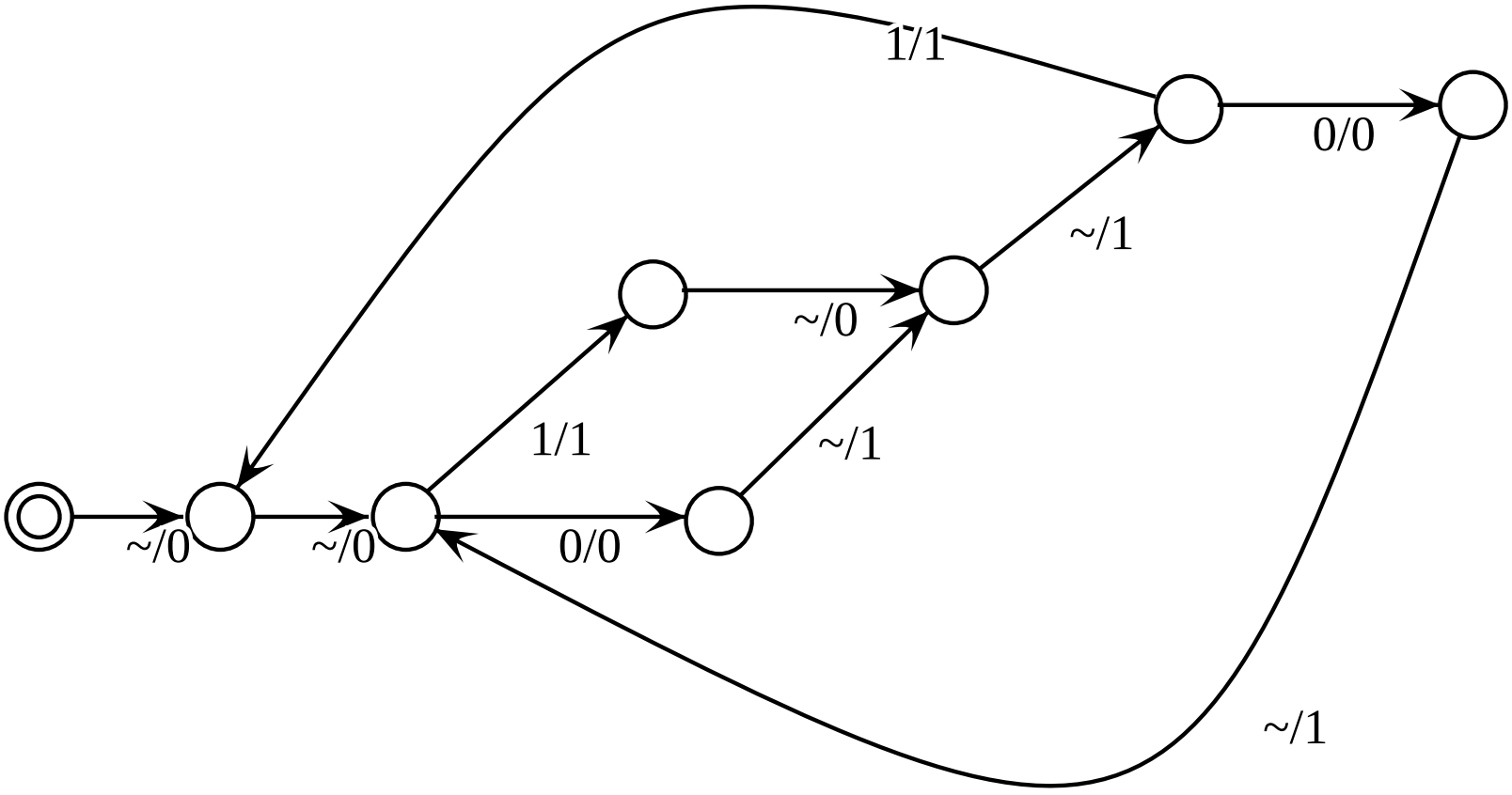
```
let node early a = generate (fun x y -> x) a
let node late a = generate (fun x y -> not y) a
```

Generating Clocks in an Abstraction with Lustre Compiler (Pascal Raymond)

```
node generate(x : bool) returns (clk: bool);
let
  -- we simply copy the input on the output
  clk = x ;
  -- but we enforce the correctness of x
  assert check(-9, -7, 3, 5, x) ;
tel

node early() returns (clk: bool);
let
  clk = generate(true);
tel

node late() returns (clk: bool);
let
  clk = generate(false);
tel
```


1/1
0/0
~/1
~/0
1/1
~/1
~/0
~/0
0/0
~/1

Abstract Operators:

- abstract not operator:

$$not^{\sim}\left(\left(b^0, b^1, r\right)\right) = \left(-b^1 - \varepsilon, -b^0 - \varepsilon, 1 - r\right)$$

with $r = \frac{n}{l}, \quad b^0 = \frac{k^0}{l}, \quad b^1 = \frac{k^1}{l}, \quad \varepsilon = \frac{l-1}{l}$

Property: $a = not^{\sim}\ not^{\sim}\ a$

- abstract on operator:

$$\left(b^0{}_1, b^1{}_1, r_1\right)\ on^{\sim}\ \left(b^0{}_2, b^1{}_2, r_2\right) = \left(b^0{}_{12}, b^1{}_{12}, r_{12}\right)$$

with:

$r_{12} = r_1 \times r_2,$
$b^0{}_{12} = b^0{}_1 \times r_2 + b^0{}_2,$
$b^1{}_{12} = b^1{}_1 \times r_2 + b^1{}_2 + \varepsilon.$

Abstraction of a composed clock

- $abs(c)$

$$
\begin{array}{lcl}
abs(\mathit{not}\ w) & \stackrel{def}{\Leftrightarrow} & \mathit{not}^{\sim}\ abs(w) \\
abs(c_1\ \mathit{on}\ c_2) & \stackrel{def}{\Leftrightarrow} & abs(c_1)\ \mathit{on}^{\sim}\ abs(c_2)
\end{array}
$$

Proposition: $c \in concr(abs(c))$

Abstract Relations

- synchronizability: $\left(b^0{}_1, b^1{}_1, r_1\right) \bowtie^{\sim} \left(b^0{}_2, b^1{}_2, r_2\right) \Leftrightarrow r_1 = r_2$
  Proposition: $abs(c_1) \bowtie^{\sim} abs(c_2) \Leftrightarrow c_1 \bowtie c_2$

- precedence: $b^0{}_1 \geq b^1{}_2 \Rightarrow a_1 \preceq^{\sim} a_2$
  Proposition: $abs(c_1) \preceq^{\sim} abs(c_2) \Rightarrow c_1 \preceq c_2$

- subtyping: $a_1 <:^{\sim} a_2 \Leftrightarrow (a_1 \bowtie^{\sim} a_2) \wedge (a_1 \preceq^{\sim} a_2)$
  $\Rightarrow$ can be checked in constant time.

Proofs in Coq

$$concr\left(\left(b^0, b^1, r\right)\right) \overset{def}{\Leftrightarrow}$$

$$\left\{ w,\ \forall i \geq 1,\quad \begin{array}{lcll} & & w[i] = 1 & \Rightarrow \quad \mathcal{O}_w(i-1) < r \times i + b^1 \\ & \wedge & w[i] = 0 & \Rightarrow \quad \mathcal{O}_w(i-1) \geq r \times i + b^0 \end{array} \right\}$$

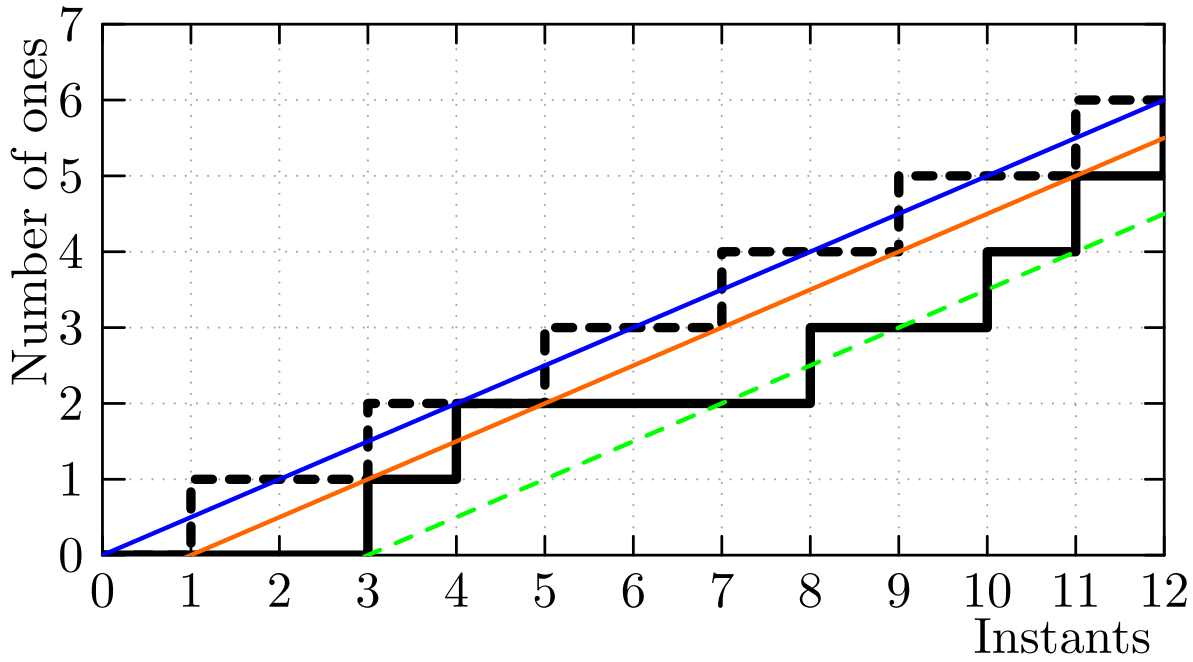
```
Definition in_abstractionj (w: ibw) (a:abstractionj) :=
  forall i: nat,
  forall H_i_ge_1: (i >= 1)%nat,
  (w i = true -> ones w (pred i) < absj_r a * i + absj_b1 a)
  /\
  (w i = false -> ones w (pred i) >= absj_r a * i + absj_b0 a).
```

Proofs in Coq

For instance, $on^{\sim}$ correctness:

```
Property on_correctness:
  forall (w1:ibw) (w2:ibw),
  forall H_wf_w1: well_formed_ibw w1,
  forall H_wf_w2: well_formed_ibw w2,
  forall (a1:abstractionj) (a2:abstractionj),
  forall H_wf_a1: well_formed_abstractionj a1,
  forall H_wf_a2: well_formed_abstractionj a2,
  forall (H_a1_eq_absj_w1: in_abstractionj w1 a1),
  forall (H_a2_eq_absj_w2: in_abstractionj w2 a2),
  in_abstractionj (on w1 w2) (on_absj a1 a2).
```

Other Applications - Modelizing Execution Time



$f :: \forall \alpha.\alpha \ \mathtt{on}^{\sim} \left(0, 0, \frac{1}{2}\right) \rightarrow \alpha \ \mathtt{on}^{\sim} \left(-\frac{3}{2}, -\frac{1}{2}, \frac{1}{2}\right)$

Composed twice:

$f \ o \ f :: \forall \alpha.\alpha \ \mathtt{on}^{\sim} \left(0, 0, \frac{1}{2}\right) \rightarrow \alpha \ \mathtt{on}^{\sim} \left(-\frac{6}{2}, -\frac{2}{2}, \frac{1}{2}\right)$

Other Applications

Modelizing Several Reads (or writes) at the Same Instant