Abstraction of Clocks in Synchronous Dataflow Systems

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

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

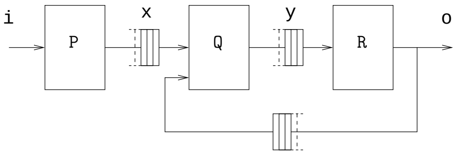
(2) LRI, Univ. Paris-Sud 11, Orsay, France and INRIA Saclay

(3) Institut Universitaire de France

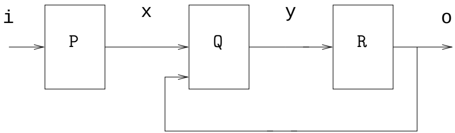
APLAS - December 2008 - Bengaluru, India

Programming Kahn Networks

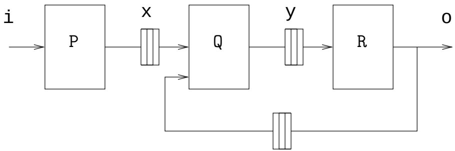
Kahn Networks (unbounded buffers):



Synchronous Kahn Networks (no buffer):



n-Synchronous Kahn Networks (bounded buffers):



Contribution

Previous work:

- n-synchrony can be checked on periodic clocks

Motivation:

- dealing with long patterns in periodic clocks
- dealing with jitter ("almost periodic" clocks)
- modeling nodes execution time

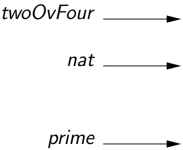
Contribution:

- n-synchrony can be checked by abstracting clocks

Overview

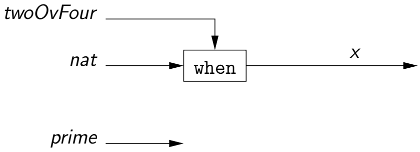
1. Synchronous and n-synchronous models
2. Abstraction of clocks
   - definition
   - abstract relations
   - abstract operators
3. Applications

Synchronous Dataflow Languages (Lustre, Signal, Lucid Synchrone)



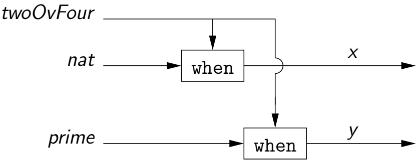
name	value							clock
nat	0	1	2	3	4	5	...	$(1)^\omega$
prime	2	3	5	7	11	13	...	$(1)^\omega$
twoOvFour	1	1	0	0	1	1	...	$(1)^\omega$
$x = nat$ when $twoOvFour$	0	1			4	5	...	$(1)^\omega$ *on twoOvFour*
$y = prime$ when $twoOvFour$	2	3			11	13	...	$(1)^\omega$ *on twoOvFour*
$z = x + y$	2	4			15	18	...	$(1)^\omega$ *on twoOvFour*
$oneOvTwo = even\ x$	1	0			1	0	...	$(1)^\omega$ *on twoOvFour*
$t = x$ when $oneOvTwo$	0				4		...	$(1)^\omega$ *on twoOvFour on oneOvTwo*

Synchronous Dataflow Languages (Lustre, Signal, Lucid Synchrone)



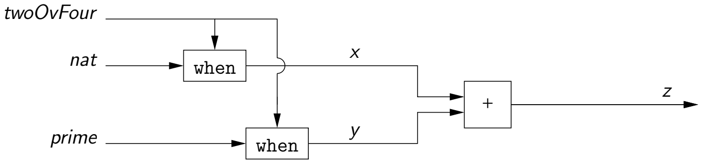
name	value							clock
nat	0	1	2	3	4	5	...	$(1)^\omega$
prime	2	3	5	7	11	13	...	$(1)^\omega$
twoOvFour	1	1	0	0	1	1	...	$(1)^\omega$
$x = nat$ `when` *twoOvFour*	0	1			4	5	...	$(1)^\omega$ *on twoOvFour*
$y = prime$ `when` *twoOvFour*	2	3			11	13	...	$(1)^\omega$ *on twoOvFour*
$z = x + y$	2	4			15	18	...	$(1)^\omega$ *on twoOvFour*
oneOvTwo = *even x*	1	0			1	0	...	$(1)^\omega$ *on twoOvFour*
$t = x$ `when` *oneOvTwo*	0				4		...	$(1)^\omega$ *on twoOvFour on oneOvTwo*

Synchronous Dataflow Languages (Lustre, Signal, Lucid Synchrone)



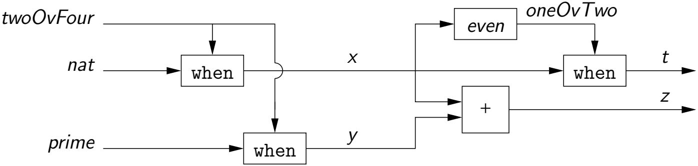
name	value							clock
nat	0	1	2	3	4	5	...	$(1)^\omega$
prime	2	3	5	7	11	13	...	$(1)^\omega$
twoOvFour	1	1	0	0	1	1	...	$(1)^\omega$
$x = nat$ `when` *twoOvFour*	0	1			4	5	...	$(1)^\omega$ *on twoOvFour*
$y = prime$ `when` *twoOvFour*	2	3			11	13	...	$(1)^\omega$ *on twoOvFour*
$z = x + y$	2	4			15	18	...	$(1)^\omega$ *on twoOvFour*
oneOvTwo = *even x*	1	0			1	0	...	$(1)^\omega$ *on twoOvFour*
$t = x$ `when` *oneOvTwo*	0				4		...	$(1)^\omega$ *on twoOvFour on oneOvTwo*

Synchronous Dataflow Languages (Lustre, Signal, Lucid Synchrone)



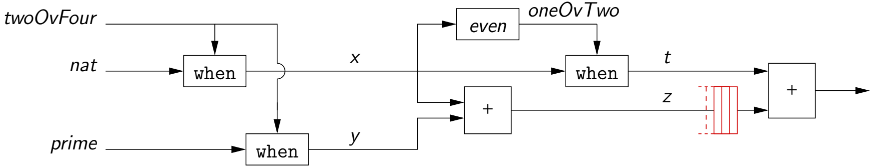
name	value							clock
nat	0	1	2	3	4	5	...	$(1)^\omega$
prime	2	3	5	7	11	13	...	$(1)^\omega$
twoOvFour	1	1	0	0	1	1	...	$(1)^\omega$
$x =$ *nat* `when` *twoOvFour*	0	1			4	5	...	$(1)^\omega$ *on twoOvFour*
$y =$ *prime* `when` *twoOvFour*	2	3			11	13	...	$(1)^\omega$ *on twoOvFour*
$z = x + y$	2	4			15	18	...	$(1)^\omega$ *on twoOvFour*
oneOvTwo = *even x*	1	0			1	0	...	$(1)^\omega$ *on twoOvFour*
$t = x$ `when` *oneOvTwo*	0				4		...	$(1)^\omega$ *on twoOvFour on oneOvTwo*

Synchronous Dataflow Languages (Lustre, Signal, Lucid Synchrone)



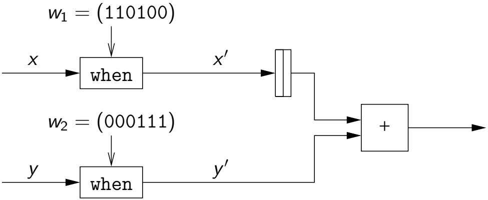
name	value							clock
nat	0	1	2	3	4	5	...	$(1)^\omega$
prime	2	3	5	7	11	13	...	$(1)^\omega$
twoOvFour	1	1	0	0	1	1	...	$(1)^\omega$
$x = nat$ `when` $twoOvFour$	0	1			4	5	...	$(1)^\omega$ *on twoOvFour*
$y = prime$ `when` $twoOvFour$	2	3			11	13	...	$(1)^\omega$ *on twoOvFour*
$z = x + y$	2	4			15	18	...	$(1)^\omega$ *on twoOvFour*
$oneOvTwo = even\ x$	1	0			1	0	...	$(1)^\omega$ *on twoOvFour*
$t = x$ `when` $oneOvTwo$	0				4		...	$(1)^\omega$ *on twoOvFour on oneOvTwo*

Synchronous Dataflow Languages (Lustre, Signal, Lucid Synchrone)



name	value							clock
nat	0	1	2	3	4	5	...	$(1)^\omega$
prime	2	3	5	7	11	13	...	$(1)^\omega$
twoOvFour	1	1	0	0	1	1	...	$(1)^\omega$
x = *nat* `when` *twoOvFour*	0	1			4	5	...	$(1)^\omega$ *on twoOvFour*
y = *prime* `when` *twoOvFour*	2	3			11	13	...	$(1)^\omega$ *on twoOvFour*
z = *x* + *y*	2	4			15	18	...	$(1)^\omega$ *on twoOvFour*
oneOvTwo = *even x*	1	0			1	0	...	$(1)^\omega$ *on twoOvFour*
t = *x* `when` *oneOvTwo*	0				4		...	$(1)^\omega$ *on twoOvFour on oneOvTwo*

Relaxing the synchronous condition

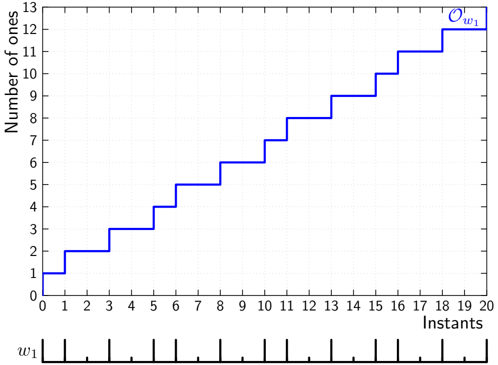


x'	x_0	x_1	x_3			x_6
y'			y_3	y_4	y_5	
$buffer(x') + y'$			$x_0 + y_3$	$x_1 + y_4$	$x_3 + y_5$	

n-synchronous model:

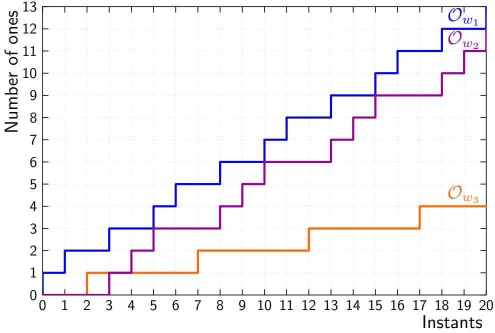
- communication through bounded buffers
- if $w_1 <: w_2$ a buffer can be inserted
- Example: $(110100) <: (000111)$

Clocks as Infinite Binary Words



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

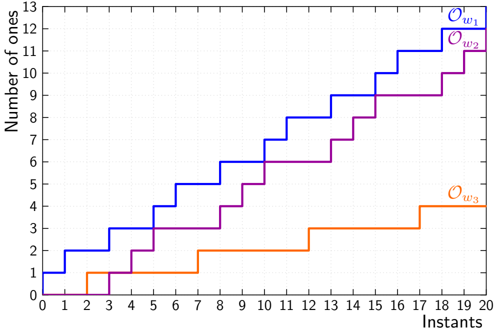
Clocks as Infinite Binary Words



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

subtyping $\quad w_1 <: w_2 \overset{def}{\Leftrightarrow} \exists n \in \mathbb{N}, \forall i,\ 0 \leq \mathcal{O}_{w_1}(i) - \mathcal{O}_{w_2}(i) \leq n$

Clocks as Infinite Binary Words



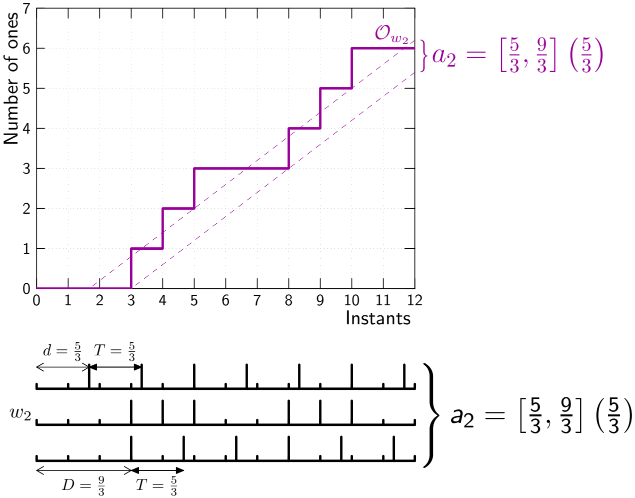
buffer $\quad size(w_1, w_2) = \max_{i \in \mathbb{N}}(\mathcal{O}_{w_1}(i) - \mathcal{O}_{w_2}(i))$

subtyping $\quad w_1 <: w_2 \overset{def}{\Leftrightarrow} \exists n \in \mathbb{N}, \forall i,\ 0 \leq \mathcal{O}_{w_1}(i) - \mathcal{O}_{w_2}(i) \leq n$

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

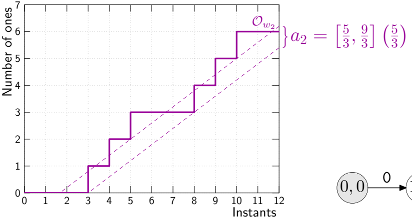
precedence $\quad w_1 \preceq w_2 \overset{def}{\Leftrightarrow} \forall i,\ \mathcal{O}_{w_1}(i) \geq \mathcal{O}_{w_2}(i)$

Abstraction of Infinite Binary Words: $abs(w) = [d, D]\,(T)$

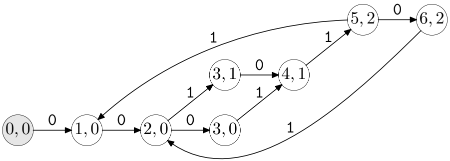


$$concr\,([d, D]\,(T)) \stackrel{def}{\Leftrightarrow} \{w,\ \forall j \geq 0,\ T \times j + d \leq [w]_{j+1} \leq T \times j + D\}$$

Abstract Clocks as Automata



$a_2 = \left[\frac{5}{3}, \frac{9}{3}\right]\left(\frac{5}{3}\right)$

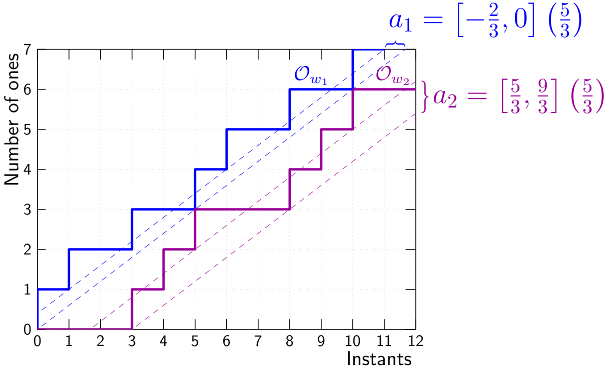


- Set of states $\{(i, j) \in \mathbb{N}^2\}$: coordinates in the 2D-chronogram
- Finite number of state equivalence classes.
- Transition function δ:

$$\begin{array}{lcll} \delta(1,(i,j)) & = & nf(i+1, j+1) & \text{if } T \times j + d \leq i \leq T \times j + D \\ \delta(0,(i,j)) & = & nf(i+1, j) & \text{if } i+1 \leq T \times j + D \end{array}$$

- Allows to check/generate clocks

Abstract Relations

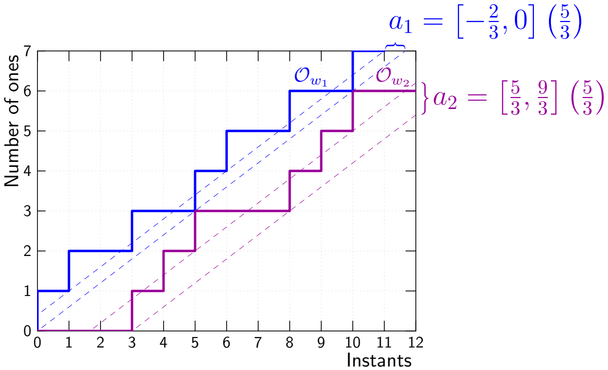


Synchronizability:

$$[d_1, D_1]\,(T_1) \bowtie^{\sim} [d_2, D_2]\,(T_2) \Leftrightarrow T_1 = T_2$$

- proposition: $abs(c_1) \bowtie^{\sim} abs(c_2) \Leftrightarrow c_1 \bowtie c_2$

Abstract Relations



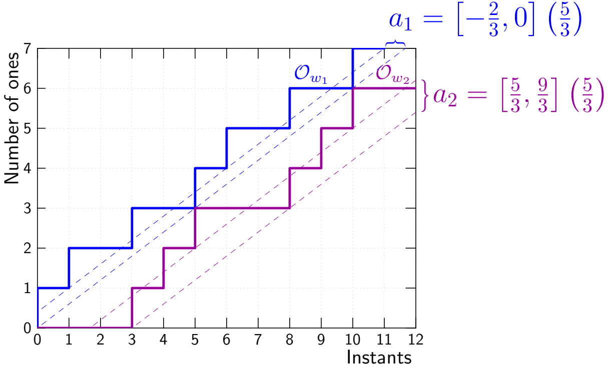
Precedence:

$$a_1 \preceq^{\sim} a_2 \Leftrightarrow \frac{K_1}{n} - \frac{k_2}{n} \leq 1 - \frac{1}{n}$$

with $a_1 = \left[\frac{k_1}{n}, \frac{K_1}{n}\right] \left(\frac{l}{n}\right)$, $a_2 = \left[\frac{k_2}{n}, \frac{K_2}{n}\right] \left(\frac{l}{n}\right)$ and $gcd(l, n) = 1$

- proposition: $abs(c_1) \preceq^{\sim} abs(c_2) \Rightarrow c_1 \preceq c_2$

Abstract Relations



Subtyping:

$$a_1 <:^{\sim} a_2 \Leftrightarrow a_1 \bowtie^{\sim} a_2 \wedge a_1 \preceq^{\sim} a_2$$

buffer size:

$$size(a_1, a_2) = \left\lceil \frac{K_2 - (n-1) - k_1}{l} \right\rceil$$

Abstract Operators

Composed clocks: $c ::= w \mid \mathit{not}\ w \mid c\ \mathit{on}\ c$

- Abstraction of a composed clock:

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

- Correctness property:

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

- Definition of $\mathit{on}^{\sim}$:

$$[d_1, D_1]\,(T_1)\ \mathit{on}^{\sim}\ [d_2, D_2]\,(T_2) = [d_{12}, D_{12}]\,(T_1 \times T_2)$$

with $d_{12} = d_1 + d_2 \times T_1$ and $D_{12} = D_1 + D_2 \times T_1$

Applications

Abstraction of periodic clocks with long patterns:

- Downscaler example:

  $abs((10100100)\ \mathit{on}\ 0^{3600}(1)\ \mathit{on}\ (1^{720}0^{720}1^{720}0^{720}0^{720}1^{720}0^{720}0^{720}1^{720}))$

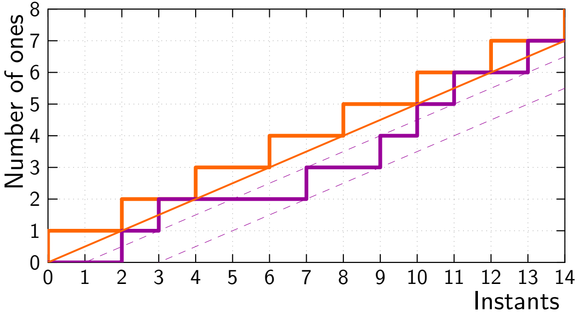
  $= [-\frac{2}{3}, 0]\ (\frac{8}{3})\ \mathit{on}^{\sim}\ [3600, 3600]\ (1)\ \mathit{on}^{\sim}\ [\frac{-4315}{4}, \frac{3600}{4}]\ (\frac{9}{4}) = [6723, 12000]\ (6)$

Dealing with jitter:

- For instance $w \in 00.(\ (10) + (01)\ )^{\omega}$ can be specified by:

$$abs(w) = [2, 3]\ (2)$$

Applications



Modeling execution time:

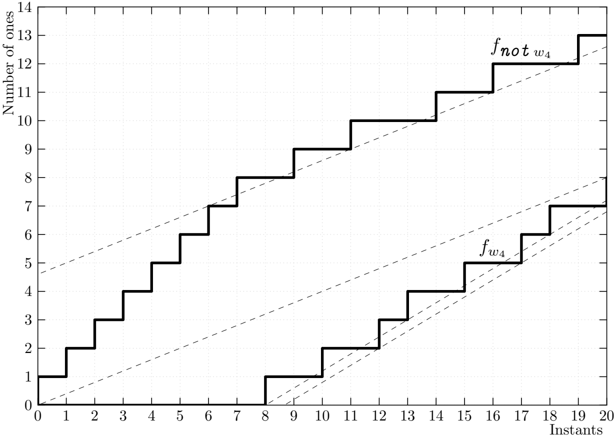
- $f :: \alpha \; \texttt{on}^{\sim} \; [0,0]\,(2) \rightarrow \alpha \; \texttt{on}^{\sim} \; [1,3]\,(2)$
- composed twice:
  $f \circ f :: \alpha \; \texttt{on}^{\sim} \; [0,0]\,(2) \rightarrow \alpha \; \texttt{on}^{\sim} \; [2,6]\,(2)$

Conclusion

- Abstracting clocks allows a more flexible composition of nodes in Synchronous Dataflow Systems.
- Correctness of abstract relations and operators have been proved in Coq.
- Current work:
  integration in the Lucid Synchrone dataflow synchronous language.

Drawbacks

- sets of $1s$ in prefix are badly abstracted.

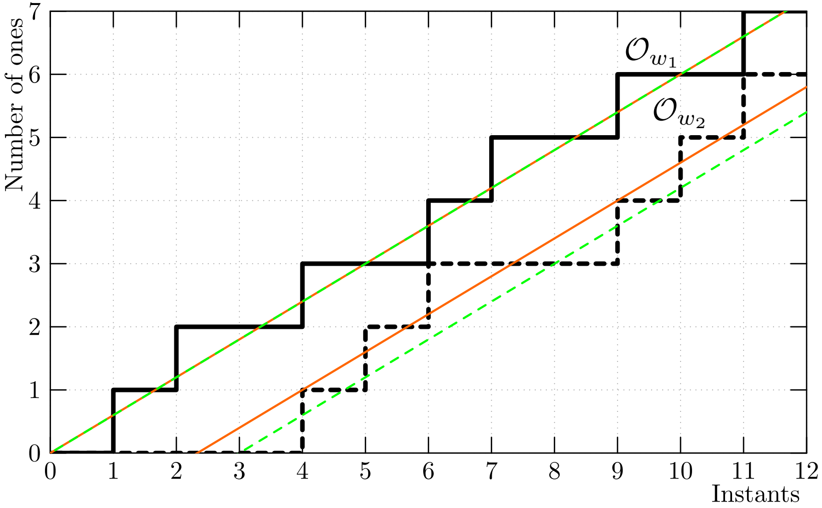


- words with finite number number of $1s$ cannot be abstracted.

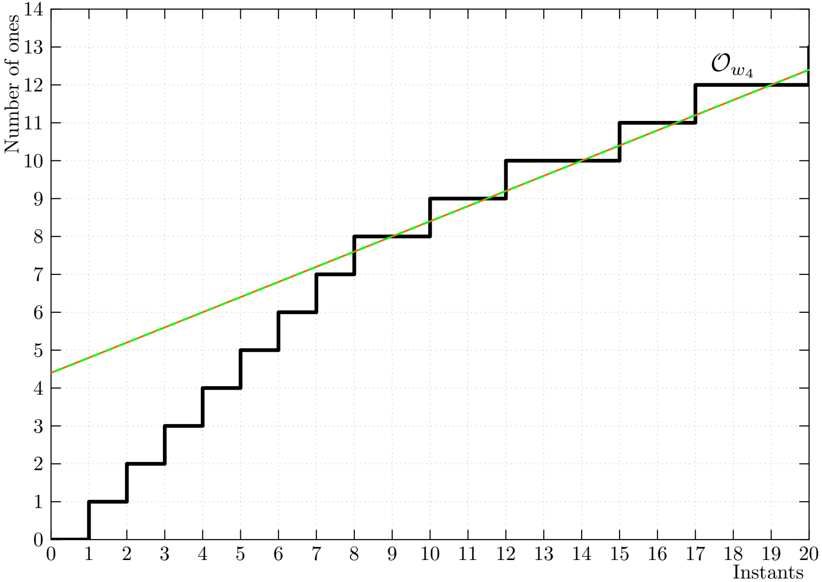
New Abstraction:

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

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



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



- Clocks with a nul rate can be abstracted.