

Sweep Synchronization as a Global Propagation Mechanism

Nicolas Beldiceanu

SICS, Uppsala, Sweden

Mats Carlsson

SICS, Uppsala, Sweden

Sven Thiel

MPI für Informatik, Saarbrücken, Germany

Overview of the talk

- what is a constraint program?
- sweep paradigm
- why synchronize?
- synchronized sweep algorithm
- a timetabling application
- benchmarks
- conclusion

What is a CP?

Constraint program:

- variables and domain constraints:
e.g. $X \in Dom(X) = [0, 3], Y \in Dom(Y) = [2, 4], Z \in Dom(Z) = [1, 5]$
- high level constraints:
e.g. $Alldiff(X, Y, Z) \wedge X + Z \geq 3 \wedge |Y - Z| \leq 2$

Solution:

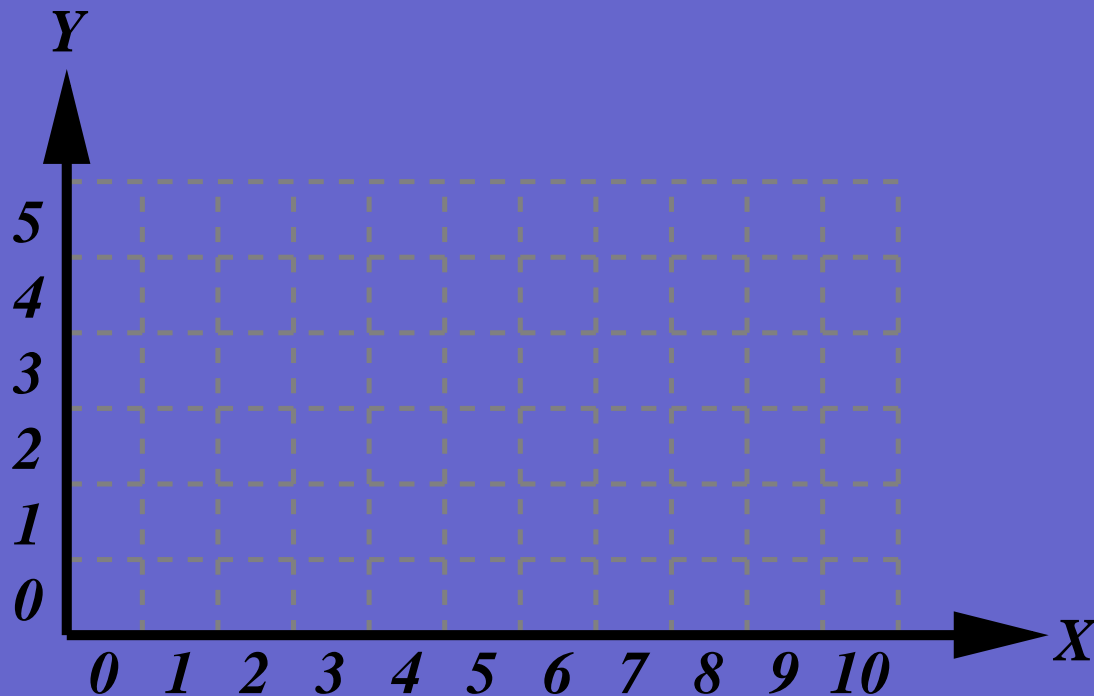
variable assignment fulfilling all constraints

e.g. $(X = 1, Y = 3, Z = 4)$

Sweep: Forbidden regions

$$C_{X,Y} = C_1(X, Y, \dots) \wedge C_2(X, Y, \dots) \wedge C_3(X, Y, \dots)$$

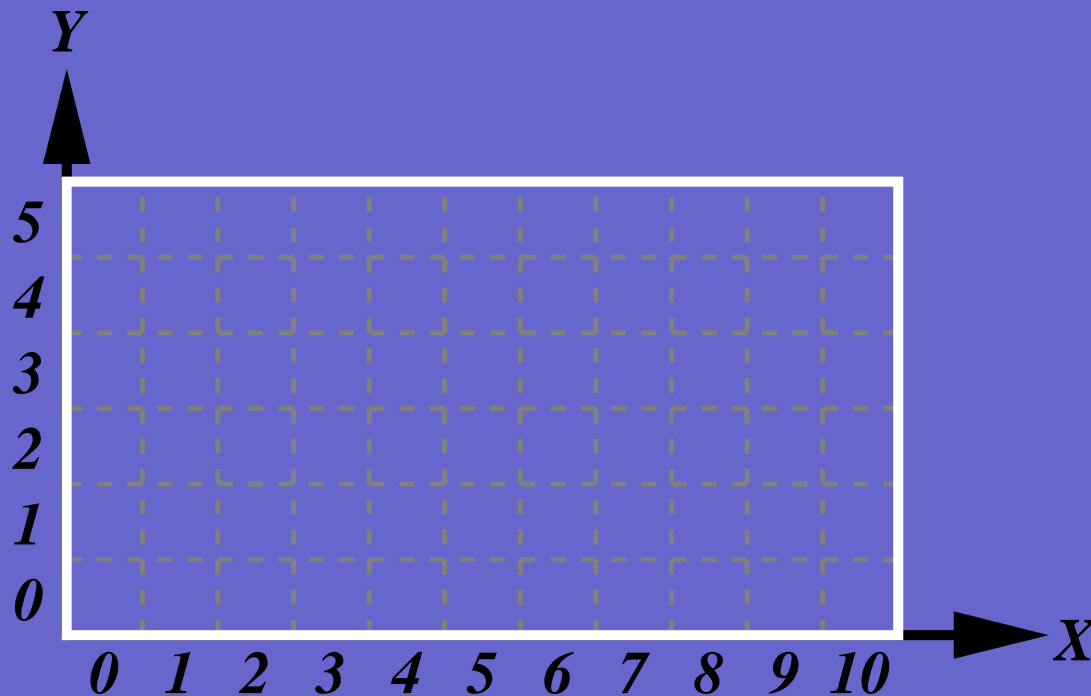
Problem: Find minimum value x s.th. solution for $X = x$ may exist!



Sweep: Forbidden regions

$$C_{X,Y} = C_1(X, Y, \dots) \wedge C_2(X, Y, \dots) \wedge C_3(X, Y, \dots)$$

Problem: Find minimum value x s.th. solution for $X = x$ may exist!

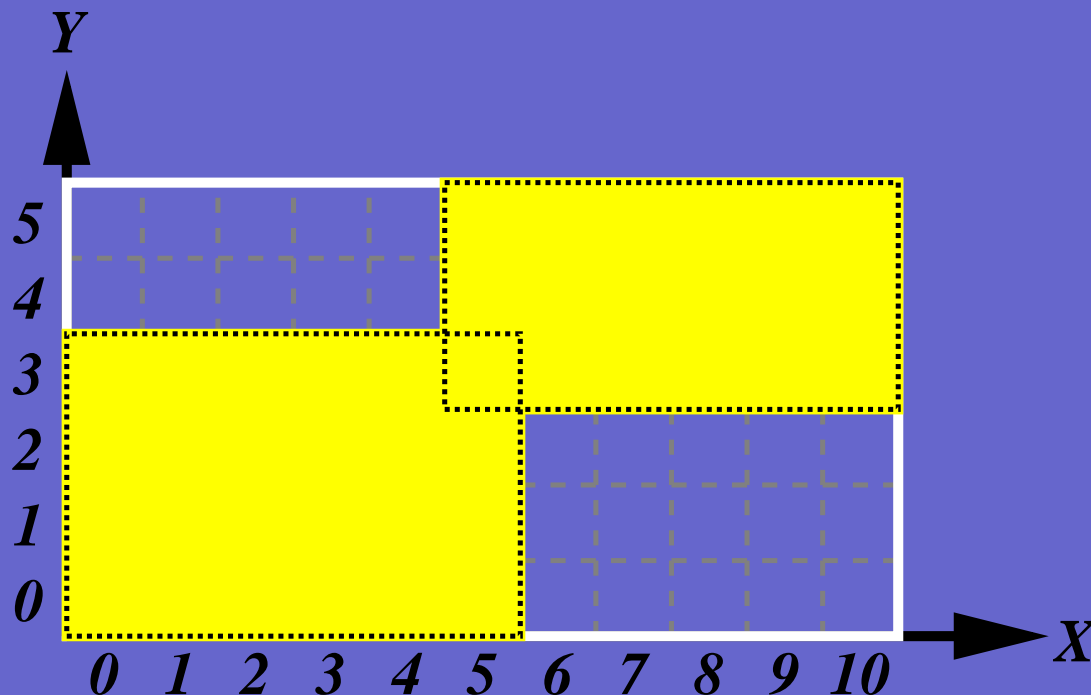


- $X \in [0, 10], Y \in [0, 5]$

Sweep: Forbidden regions

$$C_{X,Y} = C_1(X,Y,\dots) \wedge C_2(X,Y,\dots) \wedge C_3(X,Y,\dots)$$

Problem: Find minimum value x s.th. solution for $X = x$ may exist!

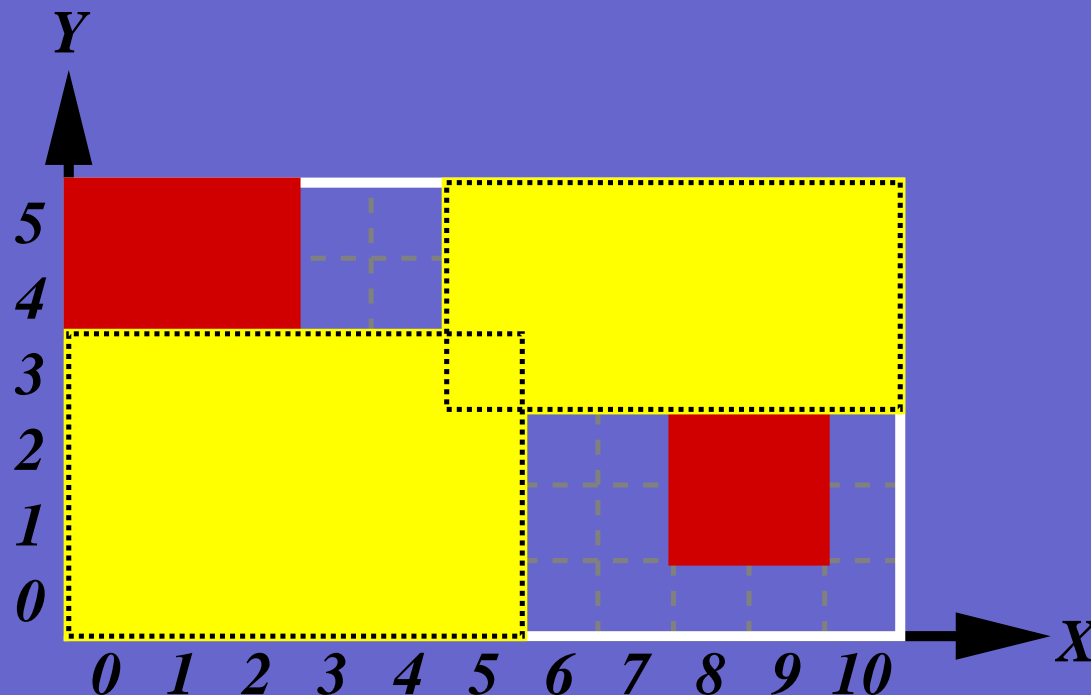


- $X \in [0, 10], Y \in [0, 5]$
- forbidden regions C_1

Sweep: Forbidden regions

$$C_{X,Y} = C_1(X,Y,\dots) \wedge C_2(X,Y,\dots) \wedge C_3(X,Y,\dots)$$

Problem: Find minimum value x s.th. solution for $X = x$ may exist!

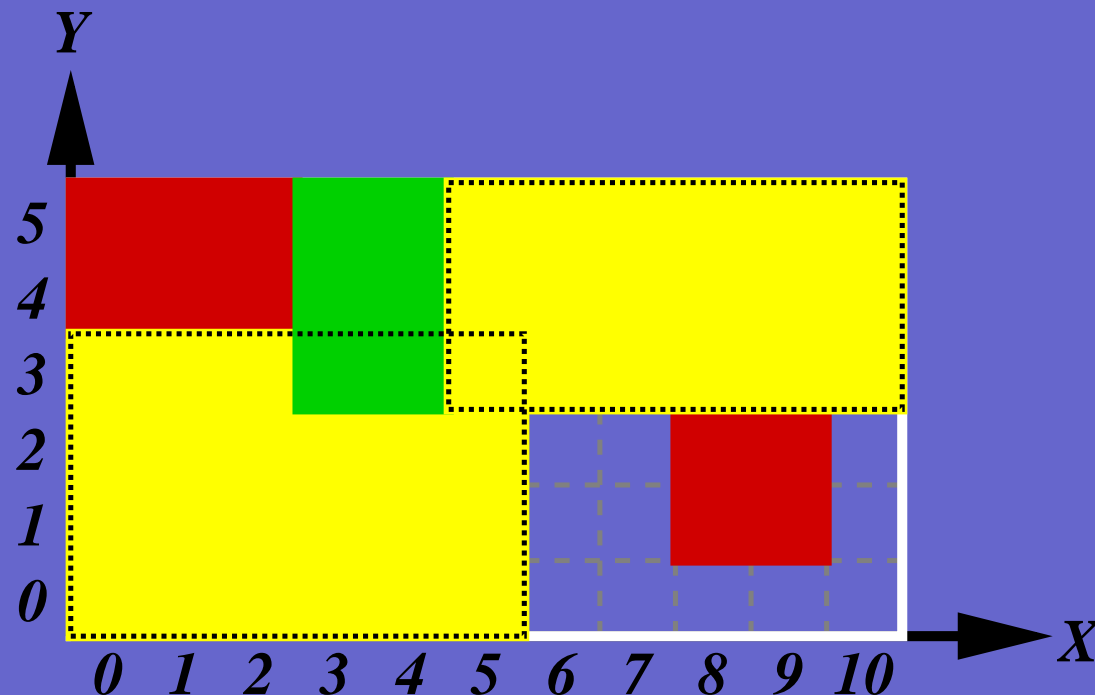


- $X \in [0, 10], Y \in [0, 5]$
- forbidden regions C_1
- forbidden regions C_2

Sweep: Forbidden regions

$$C_{X,Y} = C_1(X,Y,\dots) \wedge C_2(X,Y,\dots) \wedge C_3(X,Y,\dots)$$

Problem: Find minimum value x s.th. solution for $X = x$ may exist!



- $X \in [0, 10], Y \in [0, 5]$

- forbidden regions C_1

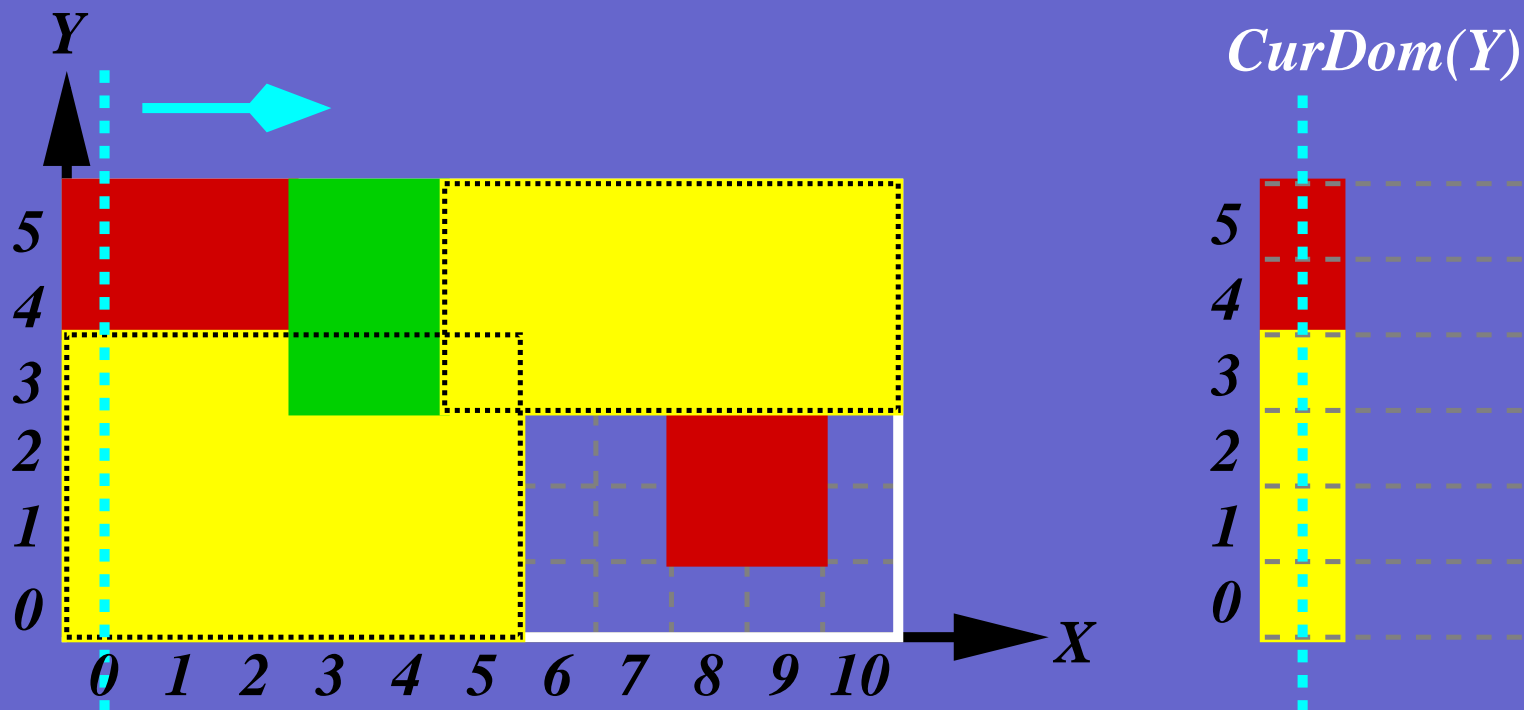
- forbidden regions C_2

- forbidden regions C_3

$$\Rightarrow x = 6$$

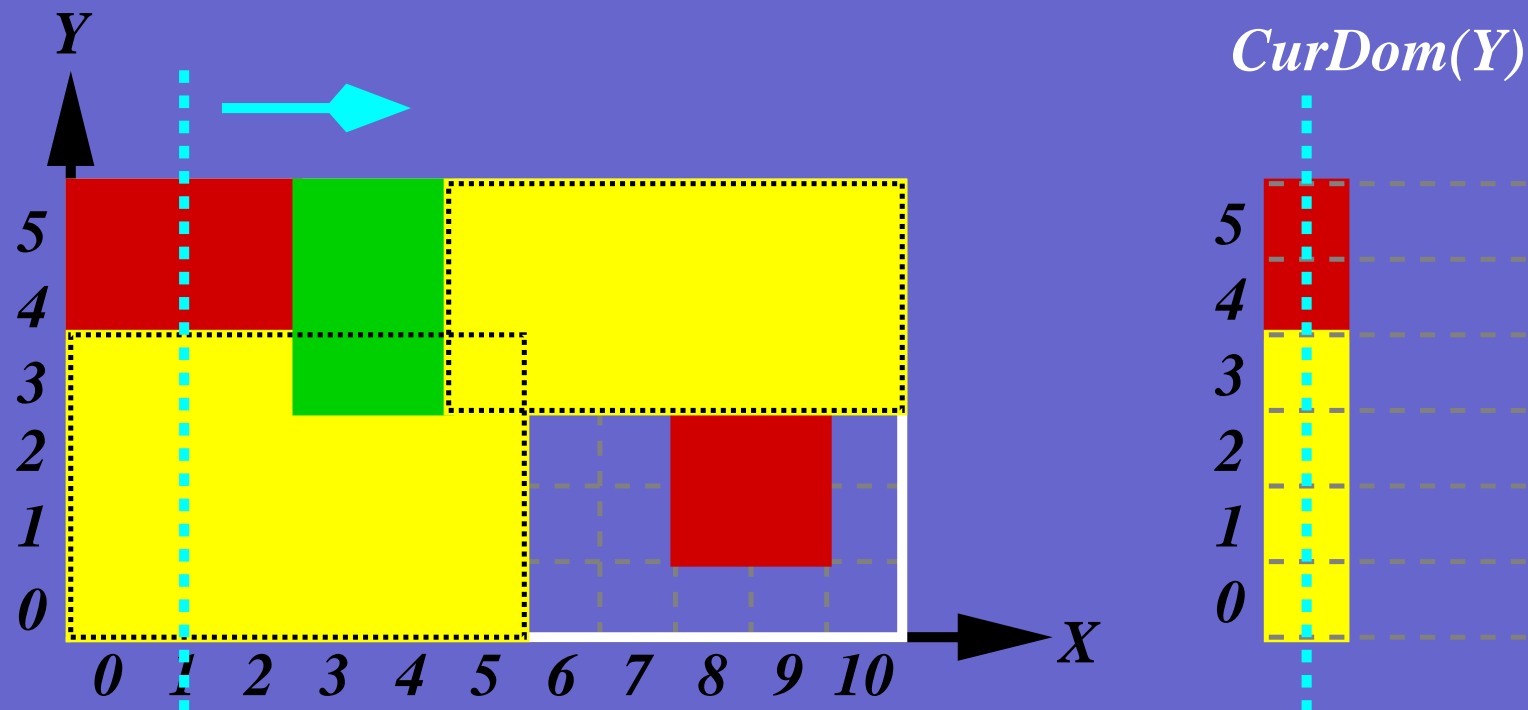
Sweep: Concept

- move **sweepline** from left to right



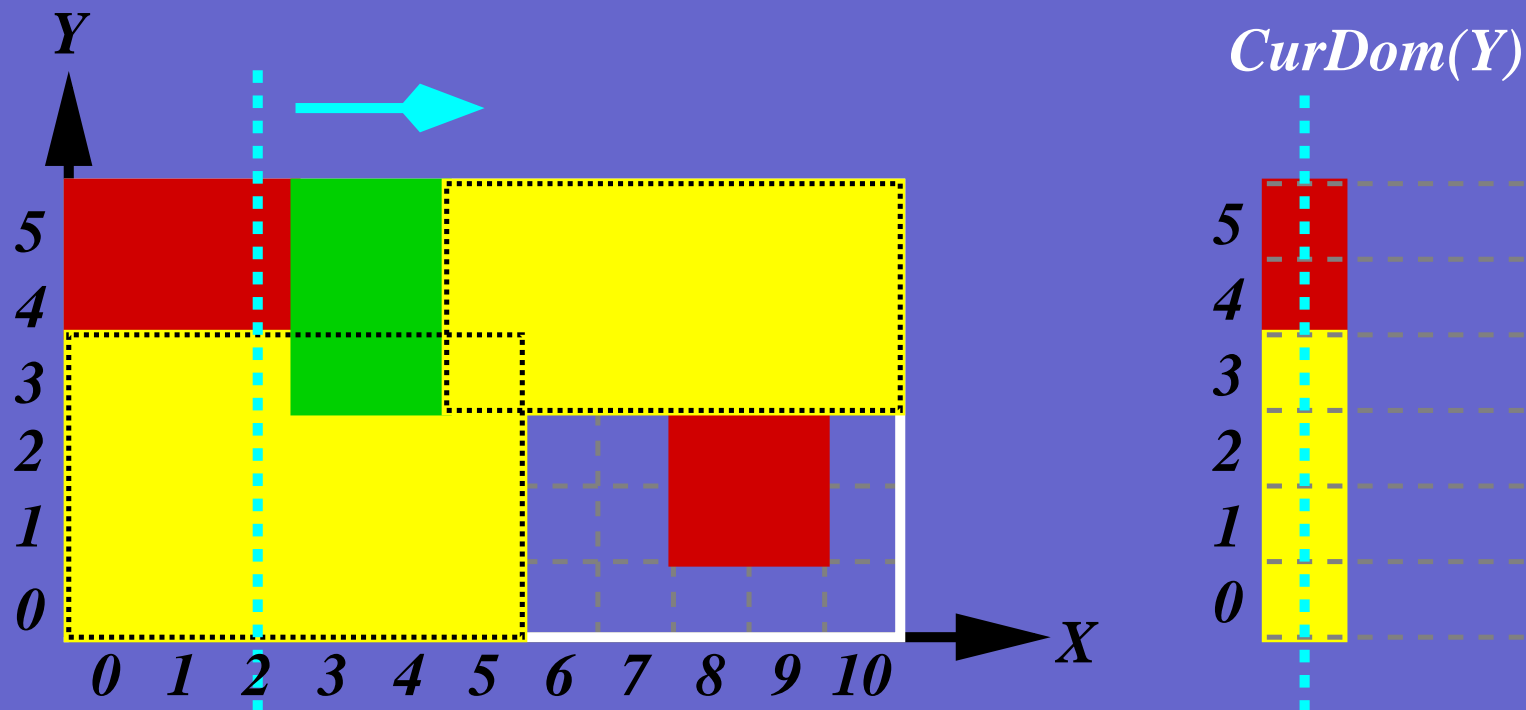
Sweep: Concept

- move **sweepline** from left to right



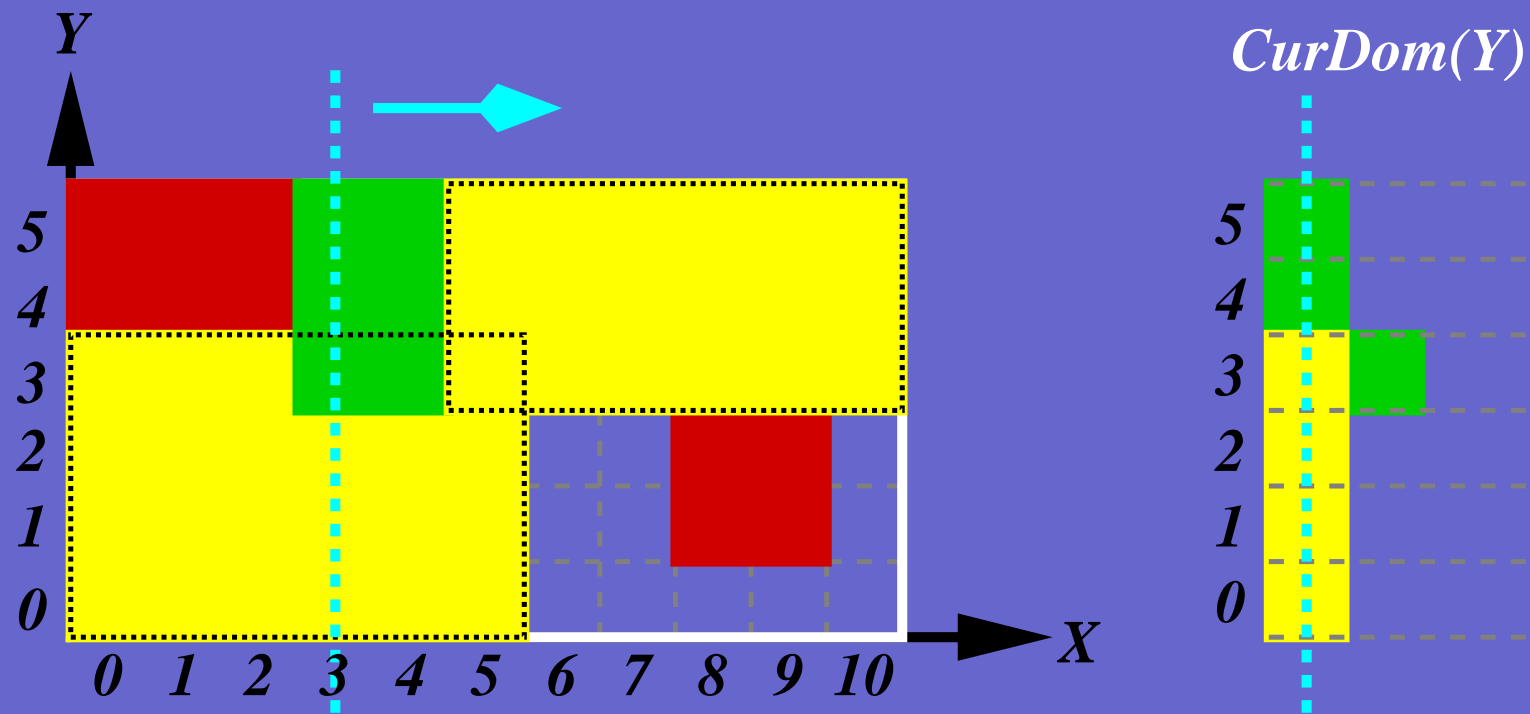
Sweep: Concept

- move **sweepline** from left to right



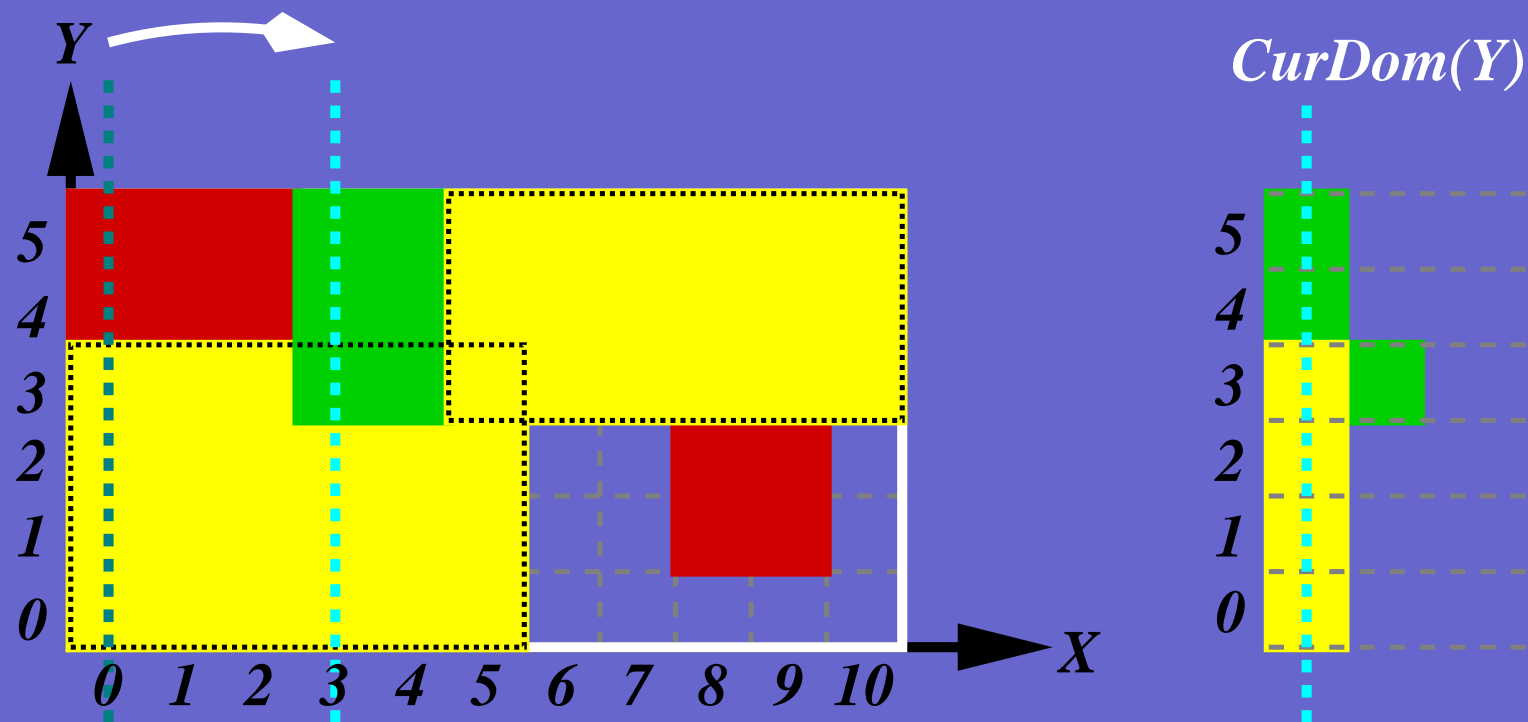
Sweep: Concept

- move **sweepline** from left to right



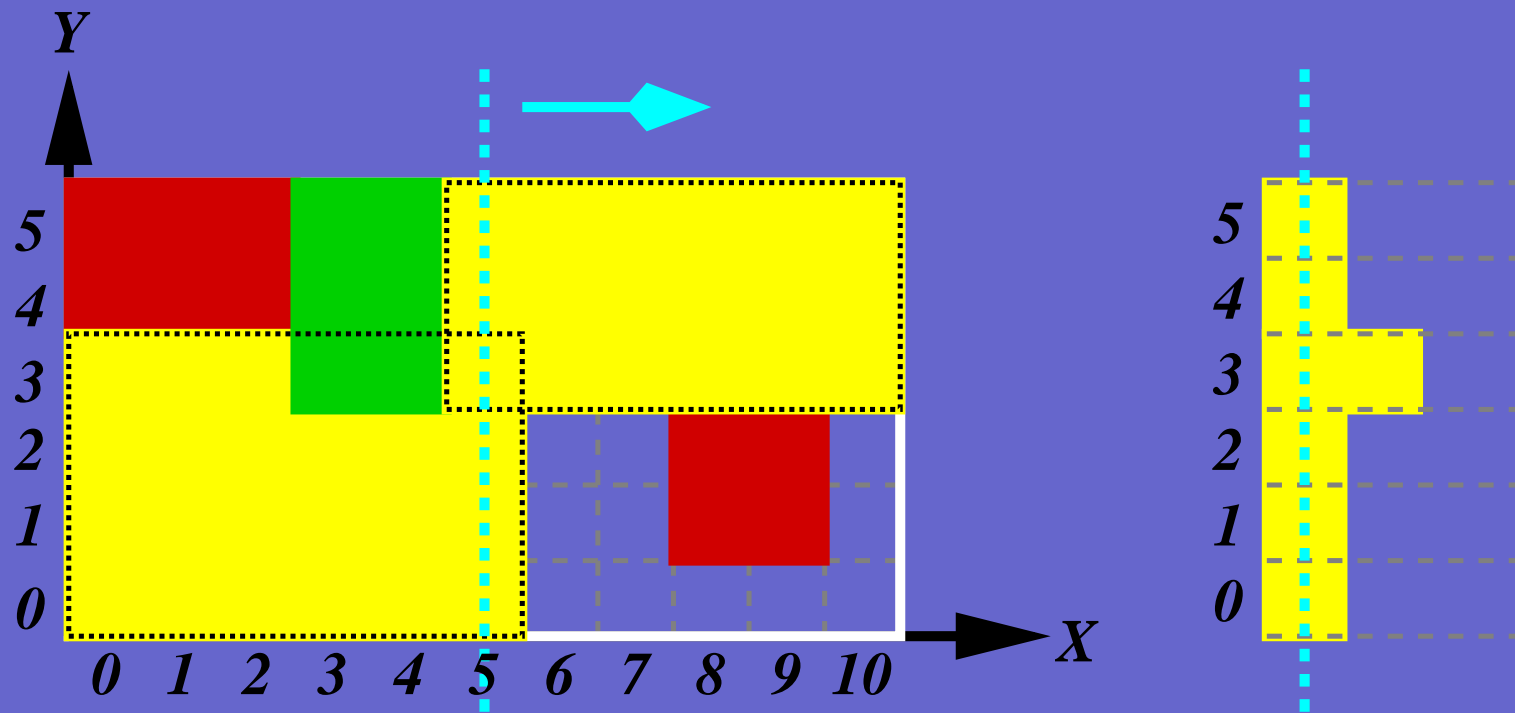
Sweep: Concept

- move **sweepline** from left to right
- stop only when events occur



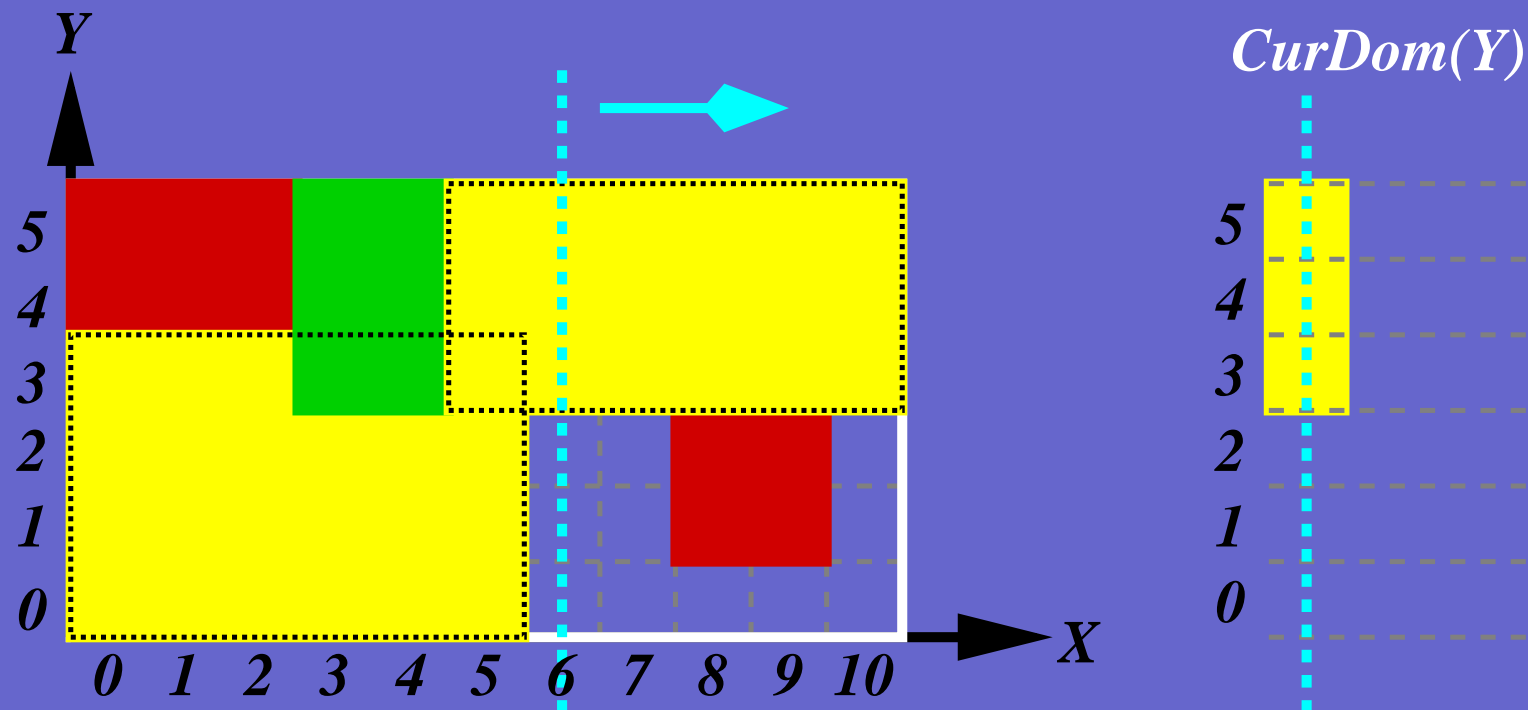
Sweep: Concept

- move **sweepline** from left to right
- stop only when events occur



Sweep: Concept

- move **sweepline** from left to right
- stop only when events occur



Sweep: Algorithm

Function: Sweep($X, Y; \mathcal{C}$)

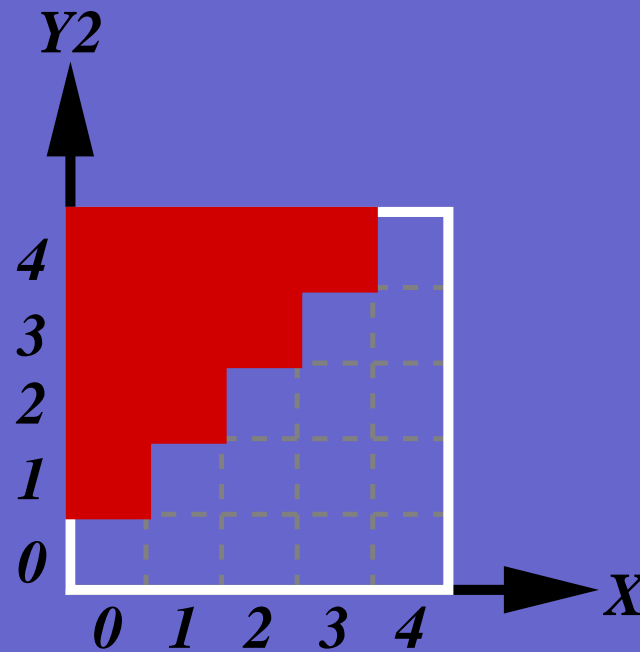
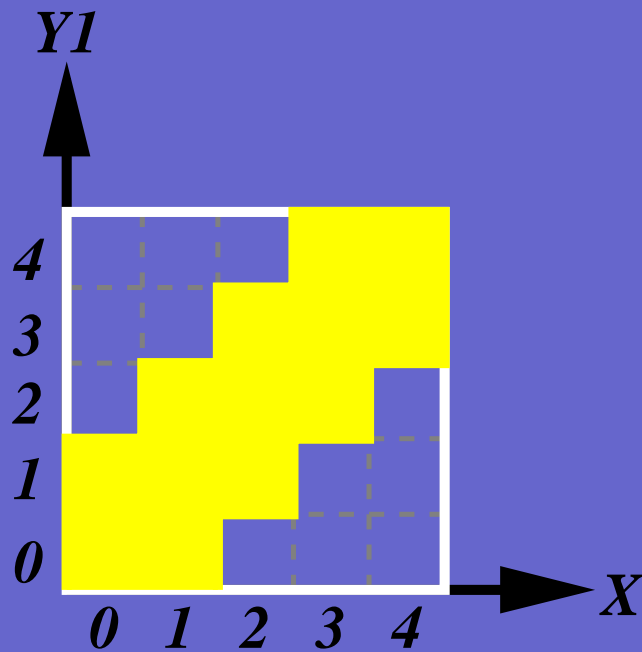
- 1: $Q \leftarrow \emptyset$ // event queue
- 2: **for all** forbidden regions $R = [x_s, x_e] \times [y_s, y_e]$ wrt. \mathcal{C} **do**
- 3: insert $E_s = (x_s, [y_s, y_e], start)$ and $E_e = (x_e + 1, [y_s, y_e], end)$ into Q
- 4: $x \leftarrow \min Dom(X)$; $CurDom(Y) \leftarrow Dom(Y)$
- 5: **while** $x \leq \max Dom(X)$ **do**
- 6: **for all** $E \in Q$ at position x **do**
- 7: extract E from Q and update $CurDom(Y)$
- 8: **if** $CurDom(Y) \neq \emptyset$ **then**
- 9: choose $y \in CurDom(Y)$; return (x, y)
- 10: **if** $Q \neq \emptyset$ **then** $x \leftarrow$ position of next event in Q **else** $x \leftarrow \infty$
- 11: return **failure**

Why synchronize?

Example:

$$X, Y_1, Y_2 \in [0, 4], Z \in [1, 2]$$

$$|X - Y_1| > Z \wedge Y_2 \leq X \wedge |Y_1 - Y_2| < 2$$

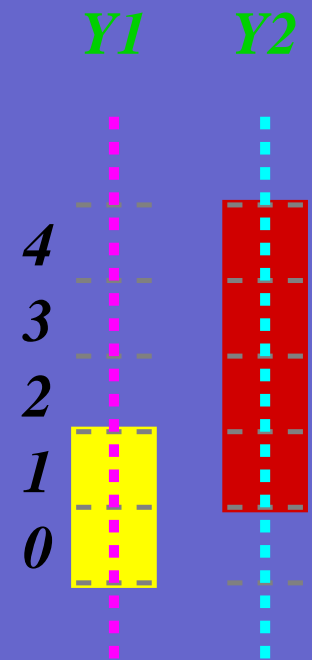
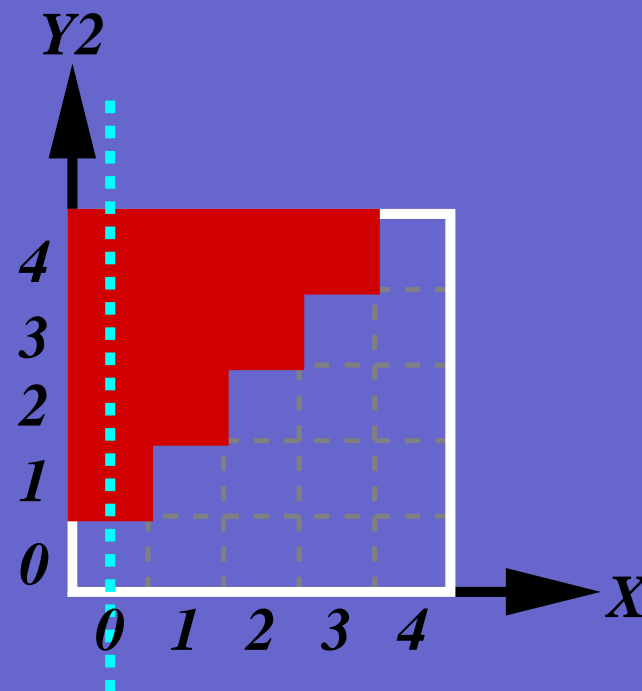
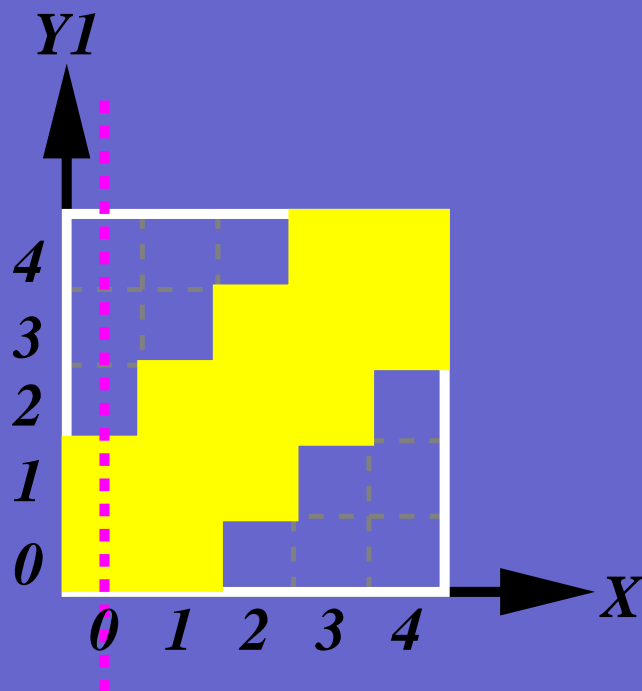


Why synchronize?

Example:

$$X, Y_1, Y_2 \in [0, 4], Z \in [1, 2]$$

$$|X - Y_1| > Z \wedge Y_2 \leq X \wedge |Y_1 - Y_2| < 2$$

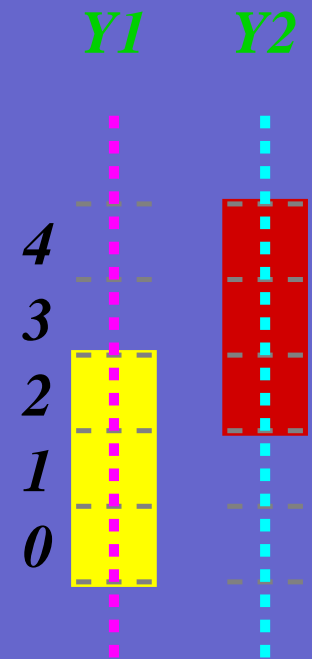
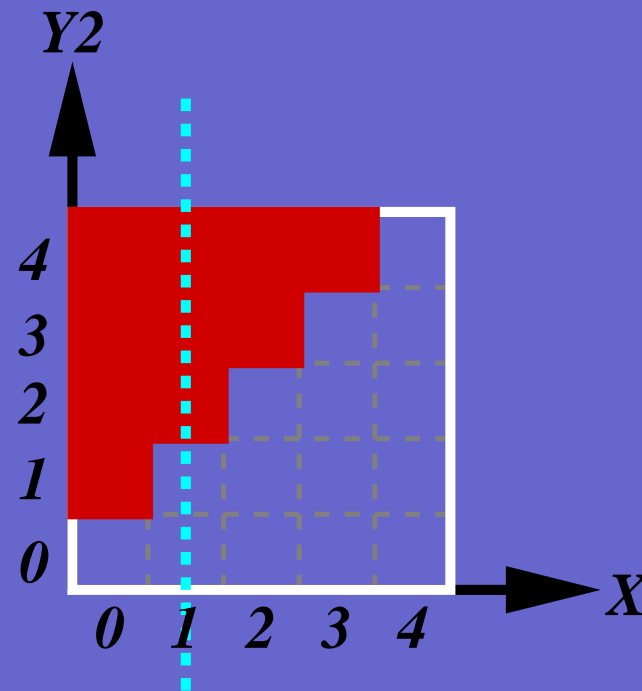
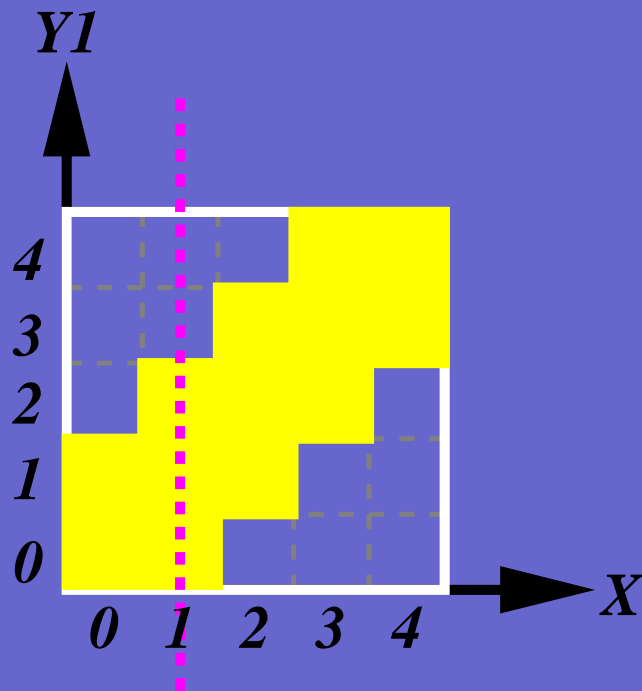


Why synchronize?

Example:

$$X, Y_1, Y_2 \in [0, 4], Z \in [1, 2]$$

$$|X - Y_1| > Z \wedge Y_2 \leq X \wedge |Y_1 - Y_2| < 2$$

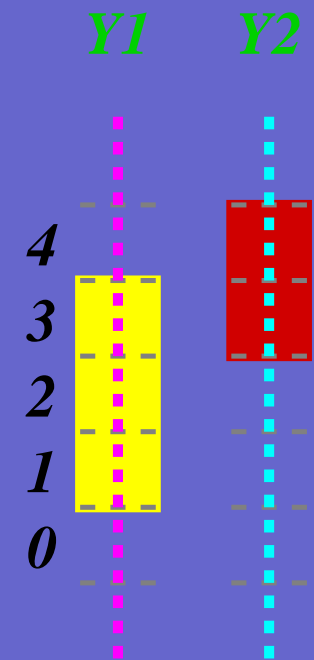
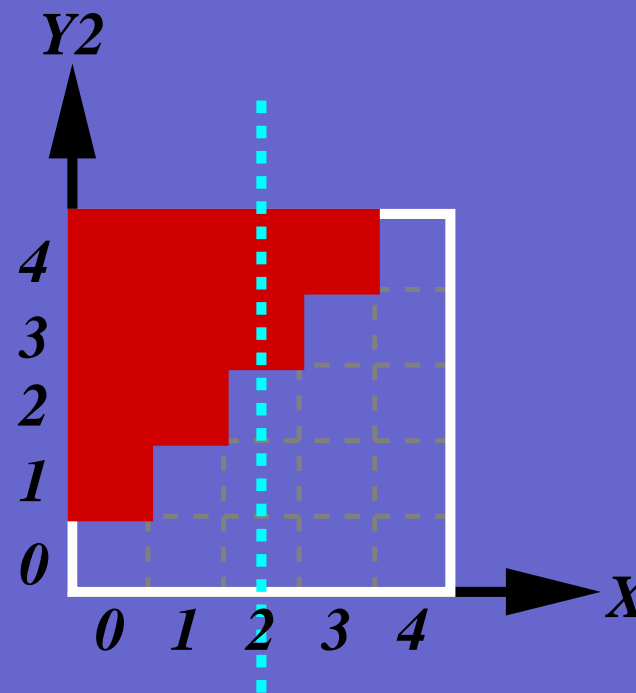
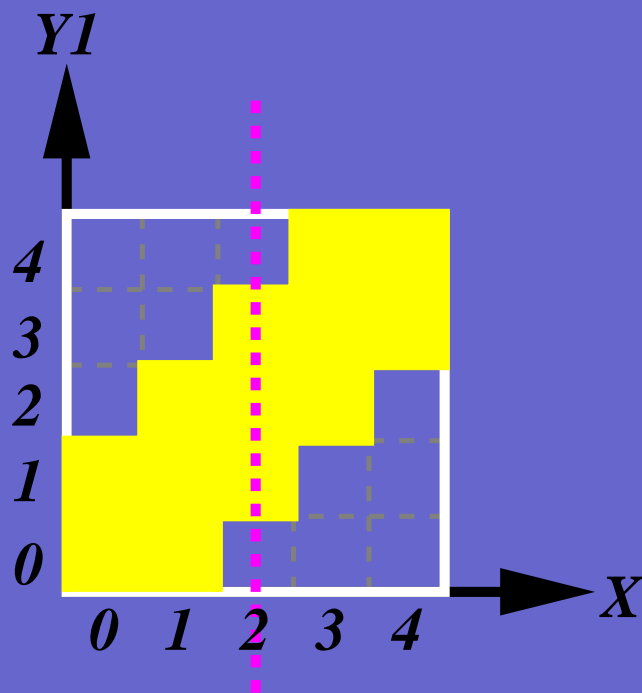


Why synchronize?

Example:

$$X, Y_1, Y_2 \in [0, 4], Z \in [1, 2]$$

$$|X - Y_1| > Z \wedge Y_2 \leq X \wedge |Y_1 - Y_2| < 2$$



Problem Definition

Considered constraints:

- domain constraints
e.g. $X, Y_1, Y_2 \in [0, 4], Z \in [1, 2]$
- sweep constraints: $\mathcal{C}_{X, Y_k} = \bigwedge C(X, Y_k, \dots)$, $k = 1, \dots, n$
e.g. $\mathcal{C}_{X, Y_1} = |X - Y_1| > Z$ and $\mathcal{C}_{X, Y_2} = Y_2 \leq X$
assumption: forbidden regions can be computed efficiently
- synchro constraints: $\mathcal{S} = \bigwedge C(Y_1, \dots, Y_n, \dots)$
e.g. $\mathcal{S} = |Y_1 - Y_2| < 2$
assumption: sound infeasibility check exists

Synchronization primitives

- $\text{InitSynchronization}(Y_1, \dots, Y_n, \mathcal{S})$
- $\text{TellEventToSynchronization}(E)$
where $E = (xpos, [y_s, y_e], start/end, k)$
- $\text{IsSynchronized}(y_1, \dots, y_n)$
returns *No* **or** *Maybe* and a tentative assignment (y_1, \dots, y_n)

Synchronized sweep algorithm

Function: $\text{SynchroSweep}(X, Y_1, \dots, Y_n; \mathcal{C}_1, \dots, \mathcal{C}_n, \mathcal{S})$

- 1: $Q \leftarrow \emptyset$ // event queue
- 2: **for all** k and **all** forbidden regions R wrt. \mathcal{C}_k **do**
- 3: insert $E_s = (x_s, [y_s, y_e], start, k)$ and $E_e = (x_e + 1, [y_s, y_e], end, k)$ into Q
- 4: **InitSynchronization** $(Y_1, \dots, Y_n, \mathcal{S})$

- 5: $x \leftarrow \min \text{Dom}(X)$; $CurD(Y_1) \leftarrow \text{Dom}(Y_1), \dots, CurD(Y_n) \leftarrow \text{Dom}(Y_n)$
- 6: **while** $x \leq \max \text{Dom}(X)$ **do**
- 7: **for all** $E \in Q$ at position x **do**
- 8: extract E from Q and update $CurD$ corresp. to E
- 9: **TellEventToSynchronization** (E)
- 10: **if** no $CurD$ is empty **and** **IsSynchronized** $(y_1, \dots, y_n) = Maybe$ **then**
- 11: return (x, y_1, \dots, y_n)
- 12: **if** $Q \neq \emptyset$ **then** $x \leftarrow$ position of next event in Q **else** $x \leftarrow \infty$
- 13: return **failure**

A timetabling application

Manager wants meeting with two developers and one marketing expert.
Problem: He and other persons are involved in other tasks.

Model:

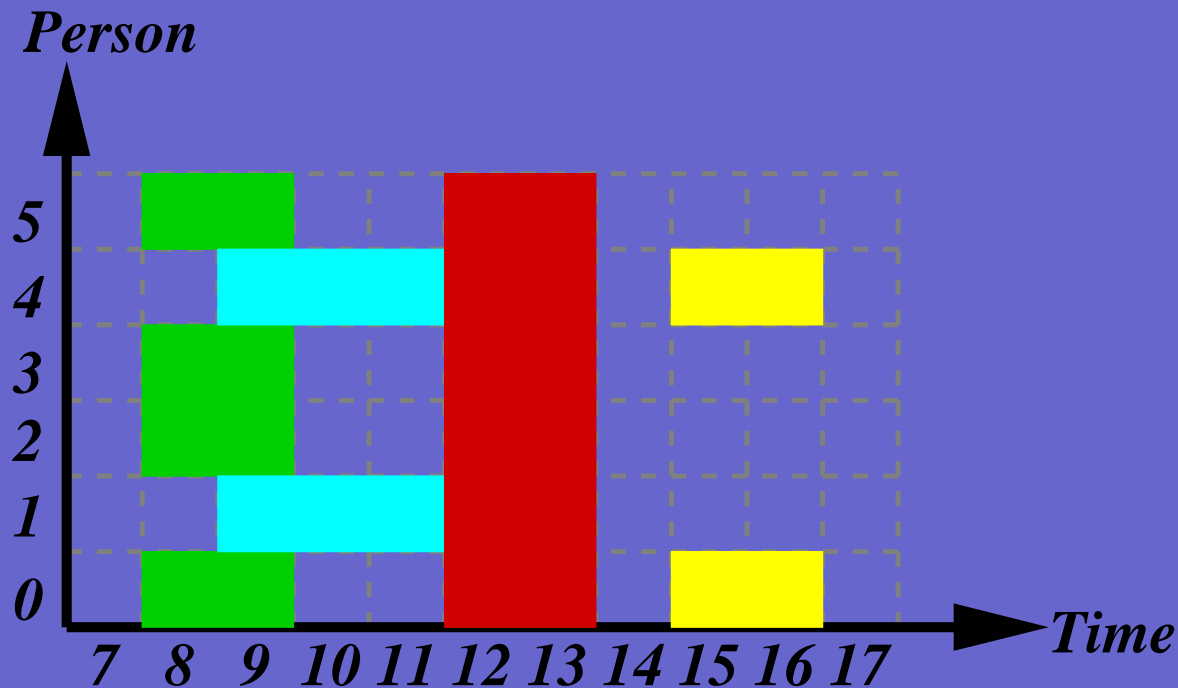
- meeting:
start time $S^{(m)}$, duration $d^{(m)}$ and involved persons $P_1^{(m)}, \dots, P_4^{(m)}$ with
 $Dom(P_1^{(m)}) = \{\text{mgr}\}$
 $Dom(P_2^{(m)}) = Dom(P_3^{(m)}) = \{\text{dev}_1, \text{dev}_2, \dots\}$
 $Dom(P_4^{(m)}) = \{\text{mark}_1, \text{mark}_2, \dots\}$
- other tasks: analogous

A timetabling application (cont'd)

Rectangle $Rec_{t,i}$ for every person $P_i^{(t)}$ assigned to task t :

origin = $(S^{(t)}, P_i^{(t)})$, width = $d^{(t)}$, height = 1

Constraint: No rectangles overlap.



Applying sweep synchronization

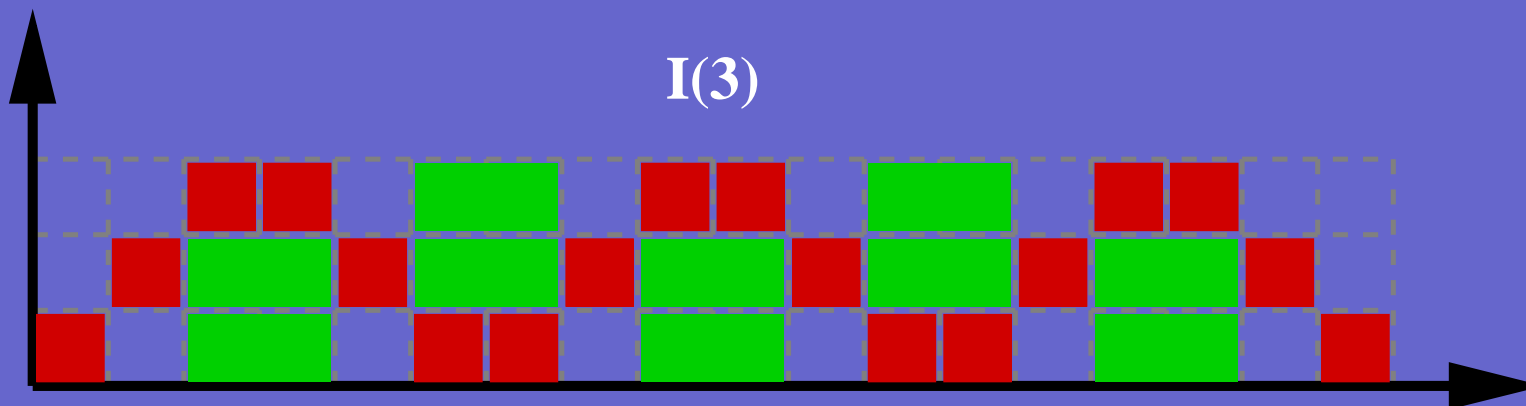
Goal: Find minimum value for $\mathcal{S}^{(m)}$ suitable for all 4 persons in the meeting

Sweep synchronization:

- sweep constraints: $\mathcal{C}_k = \bigwedge C(\mathcal{S}^{(m)}, P_k^{(m)}, \dots)$, $k = 1, \dots, 4$
 $Rec_{m,k}$ and $Rec_{t,i}$ do not overlap, for all $t \neq m$ and all i
- synchro constraint: $\mathcal{S} = \text{Alldiff}(P_1^{(m)}, \dots, P_4^{(m)})$

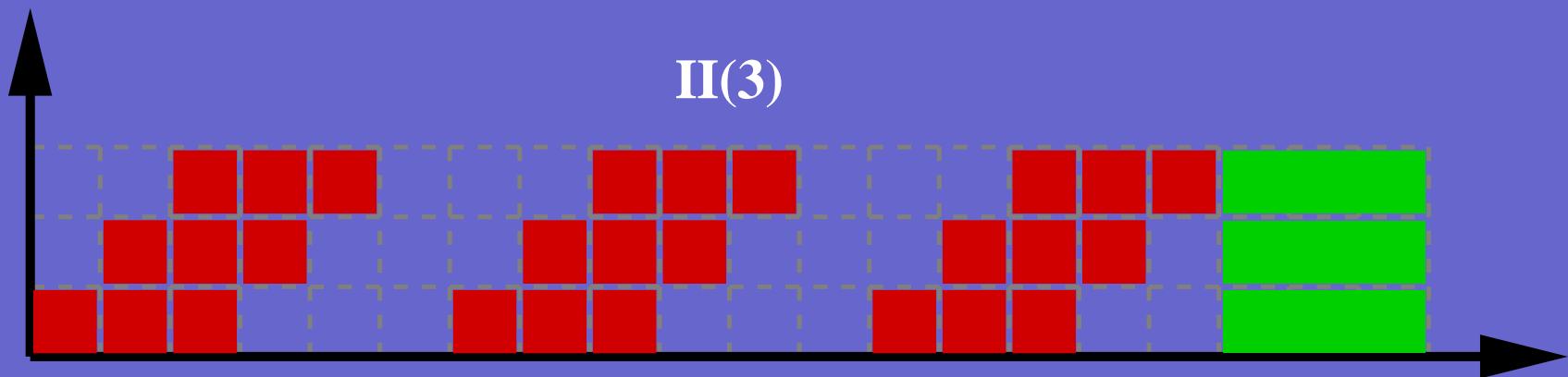
Benchmarks I

Instance	# rect.	Non-overlapping rect.		Sweep synchronization	
		#backtracks	time/msec	#backtracks	time/msec
I(4)	50	66	80	0	40
I(5)	82	528	671	0	80
I(6)	122	4440	7761	0	151
I(7)	170	40320	99313	0	280



Benchmarks II

Instance	# rect.	Non-overlapping rect.		Sweep synchronization	
		#backtracks	time/msec	#backtracks	time/msec
II(10)	110	n.a.	>1h	0	40
II(25)	650	n.a.	>1h	0	131
II(50)	2550	n.a.	>1h	0	861
II(100)	10100	n.a.	>1h	0	9886



Conclusion

Sweep synchronization

- can be applied if you have
 - several independent sweeps (pruning X acc. to Y_1, Y_2, \dots)
and synchronizing constraints (linking Y_1, Y_2, \dots)
- is a generic framework
 - sweeps + synchro feasibility check \rightarrow single propagator
- may yield considerable speed-up in practice