

Revisiting the *cardinality* Operator and Introducing the *cardinality-path* Constraint Family

Nicolas Beldiceanu and Mats Carlsson

SICS

Lägerhyddsvägen 18

75237 Uppsala

{nicolas,matsc}@sics.se

Outline of the Presentation

The *cardinality* Operator

The *cardinality-path* Family

- Situation of the *cardinality-path* Family
- Instances of the *cardinality-path* Family

Filtering Algorithms for *cardinality-path*

- Required Basic Services
- Lower Bound of the Minimum Number of Constraints that Hold
- Pruning According to the Bounds
- Example of Pruning

Conclusion and Perspectives

➔ The *cardinality* Operator

The *cardinality-path* Family

- Situation of the *cardinality-path* Family
- Instances of the *cardinality-path* Family

Filtering Algorithms for *cardinality-path*

- Required Basic Services
- Lower Bound of the Minimum Number of Constraints that Hold
- Pruning According to the Bounds
- Example of Pruning

Conclusion and Perspectives

The *cardinality* Operator

The **cardinality operator**, Van Hentenryck, P., Deville, Y. (ICLP 1991):

- **Definition:** $C = \sum_{j=1}^n \#CTR_j(V_1, \dots, V_{m_j})$

- **Pruning** based on: **counting**



$$C \leq 5 - 1$$

entailment

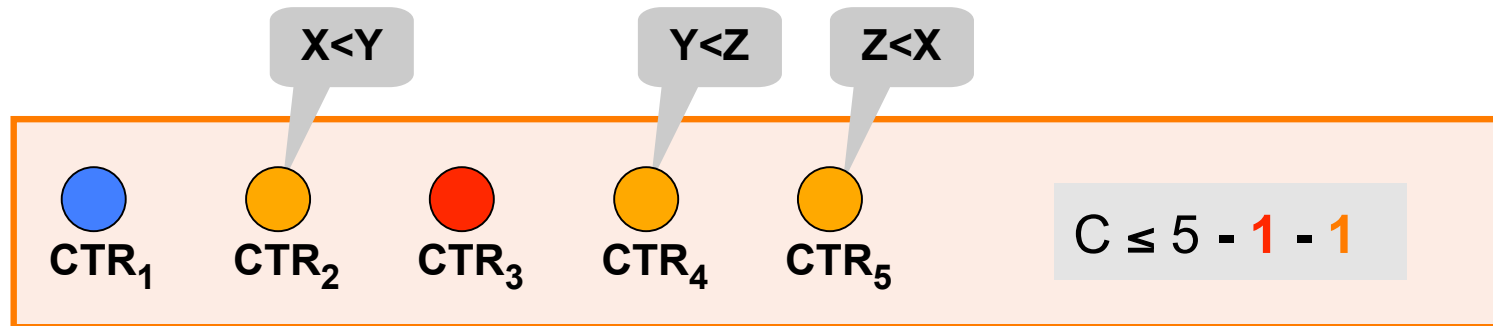


If $C \geq 1+3$ then

CTR₂, CTR₄, CTR₅ hold

The Pruning

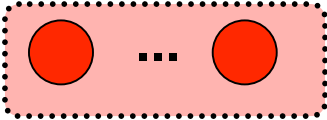
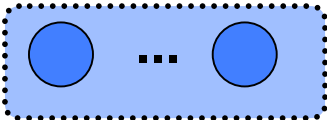
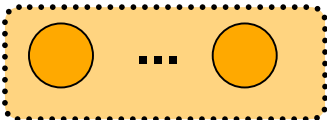
Perfect if the constraints are **independant**,
but
weak if the constraints **share** some variables:

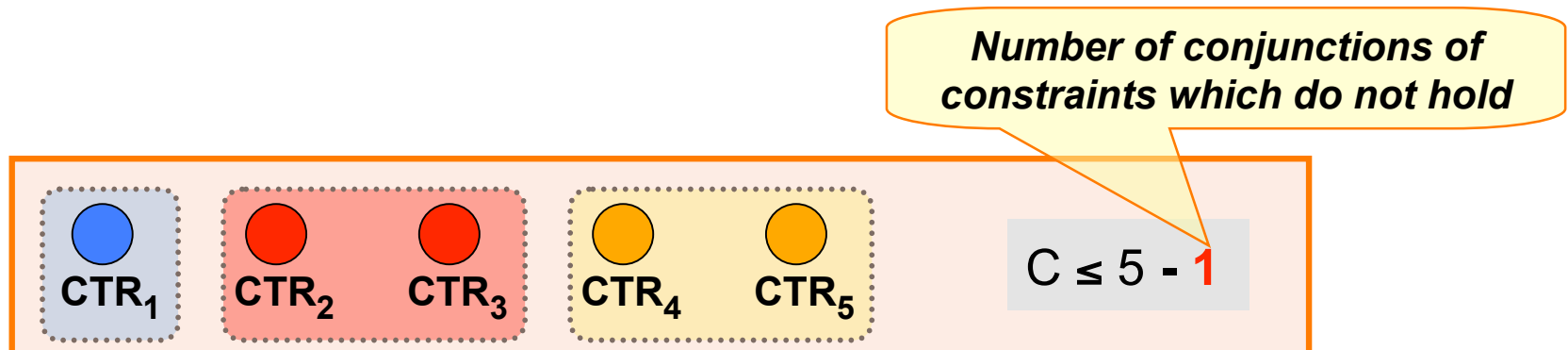


*Generalize existing deduction rules
of the cardinality operator*

Estimate the maximum number of constraints which hold

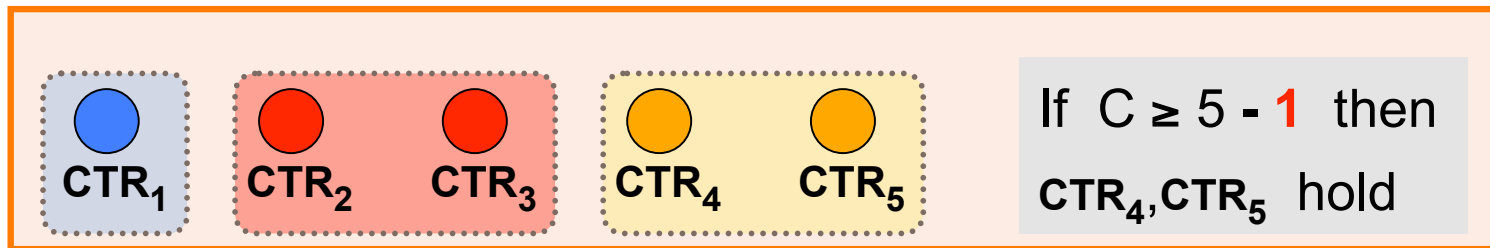
Partition the constraints in sets of the following types:

-  : the conjunction of constraints **does not hold**
-  : the conjunction of constraints **allway holds**
-  : the conjunction of constraints **may hold**



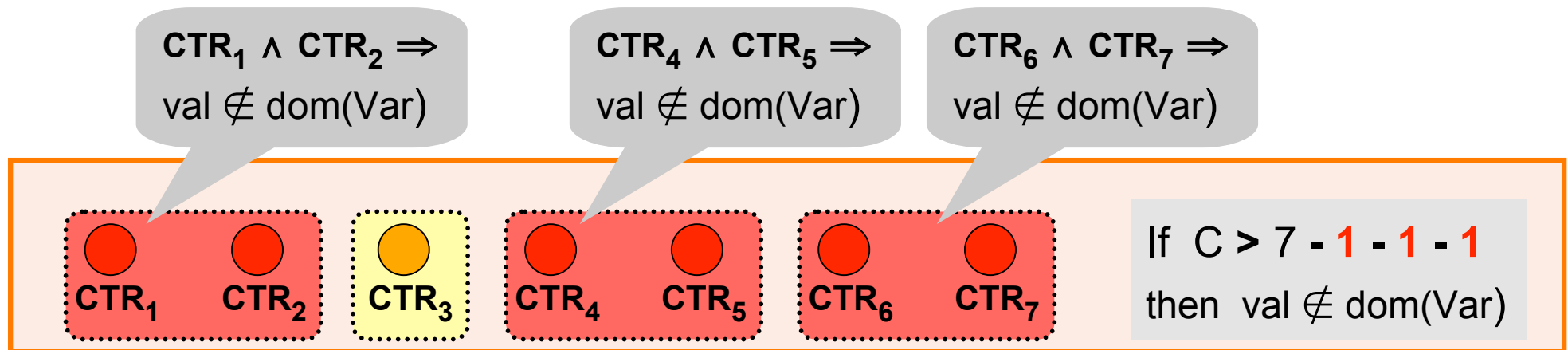
Enforcing conjunctions of constraints

Enforcing constraints according to $\min(C)$:



Pruning Values

Pruning a variable according to $\min(C)$:



The *cardinality* Operator

The *cardinality-path* Family

- ➔ • **Situation of the *cardinality-path* Family**
- Instances of the *cardinality-path* Family

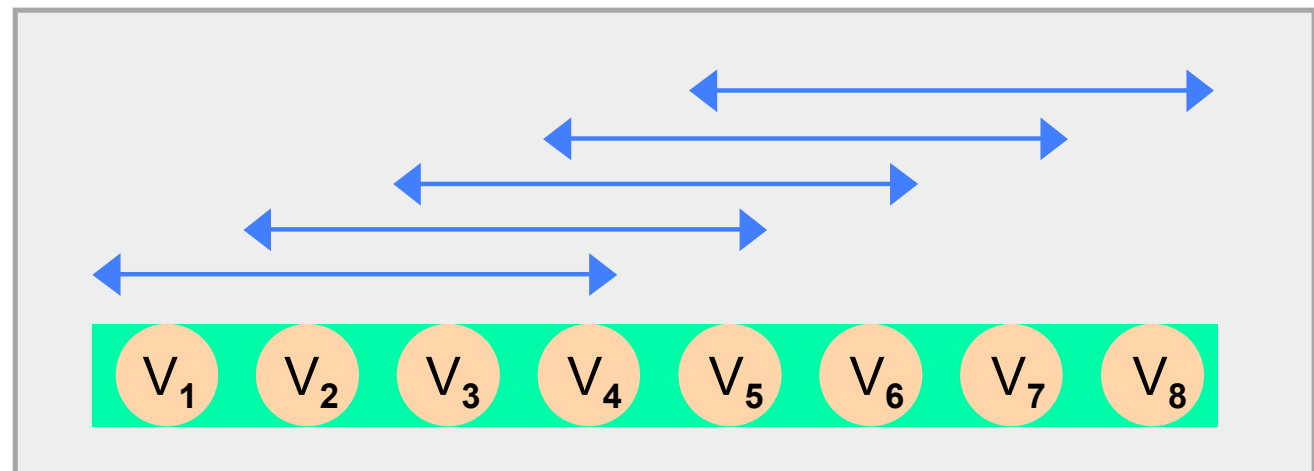
Filtering Algorithms for *cardinality-path*

- Required Basic Services
- Lower Bound of the Minimum Number of Constraints that Hold
- Pruning According to the Bounds
- Example of Pruning

Conclusion and Perspectives

The *cardinality-path* Family

- **Definition:** $C = \sum_{i=1}^{n-k+1} \#CTR(V_i, \dots, V_{i+k-1})$
- **Pruning** : { considers **interaction** between constraints
(shared variables)



Situation of the *cardinality-path* Family Inside the Classification

- Constraint parameters
 - +
 - Graph generator
 - +
 - Elementary constraint
 - +
 - Graph properties

*A **complete** description of the global constraint is **explicitly** provided !*

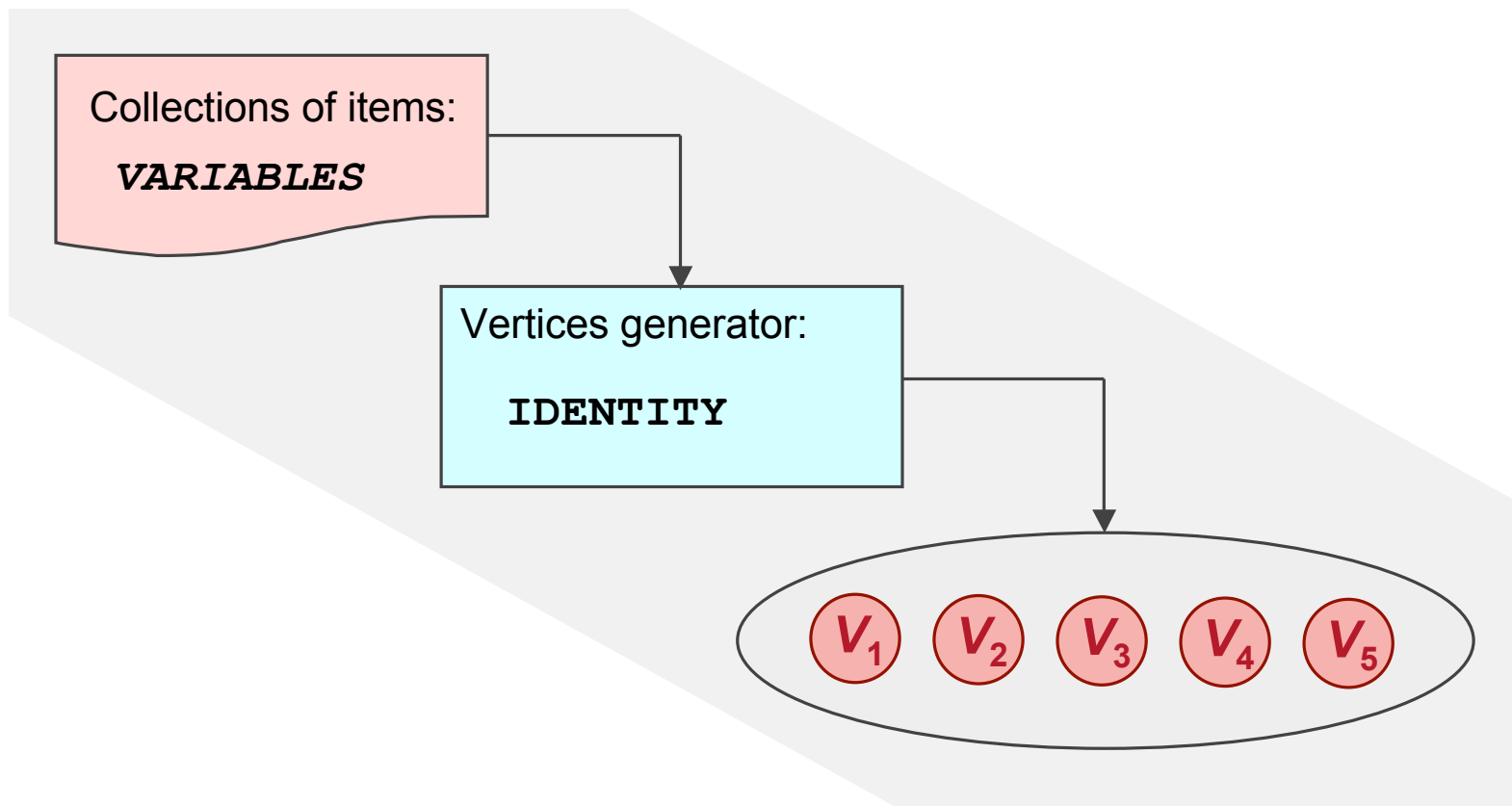
cyclic_change(*NCHANGE*, *L*, *VARIABLES*)

• ARGUMENT	:	<i>NCHANGE</i>	:	dvar
		<i>L</i>	:	int
		<i>VARIABLES</i> :	collection(<i>var-dvar</i>)	
• RESTRICTION(S)	:	<i>NCHANGE</i> ≥ 0		
		<i>NCHANGE</i> ≤ <i>VARIABLES</i>		
		<i>L</i> > 0		
		required(<i>VARIABLES.var</i>)		
• VERTEX INPUT	:	<i>VARIABLES</i>		
• VERTEX GENERATOR	:	IDENTITY		
• EDGE INPUT	:	<i>VARIABLES</i>		
• EDGE GENERATOR	:	PATH		
• EDGE ARITY	:	2		
• EDGE CONSTRAINT	:	(<i>VARIABLES.var</i> [1]+1) mod <i>L</i> ≠ <i>VARIABLES.var</i> [2]		
• GRAPH PROPERTY	:	<i>NEDGE</i> = <i>NCHANGE</i>		

cyclic_change(2,4,{var-3,var-0, var-2,var-3, var-1})

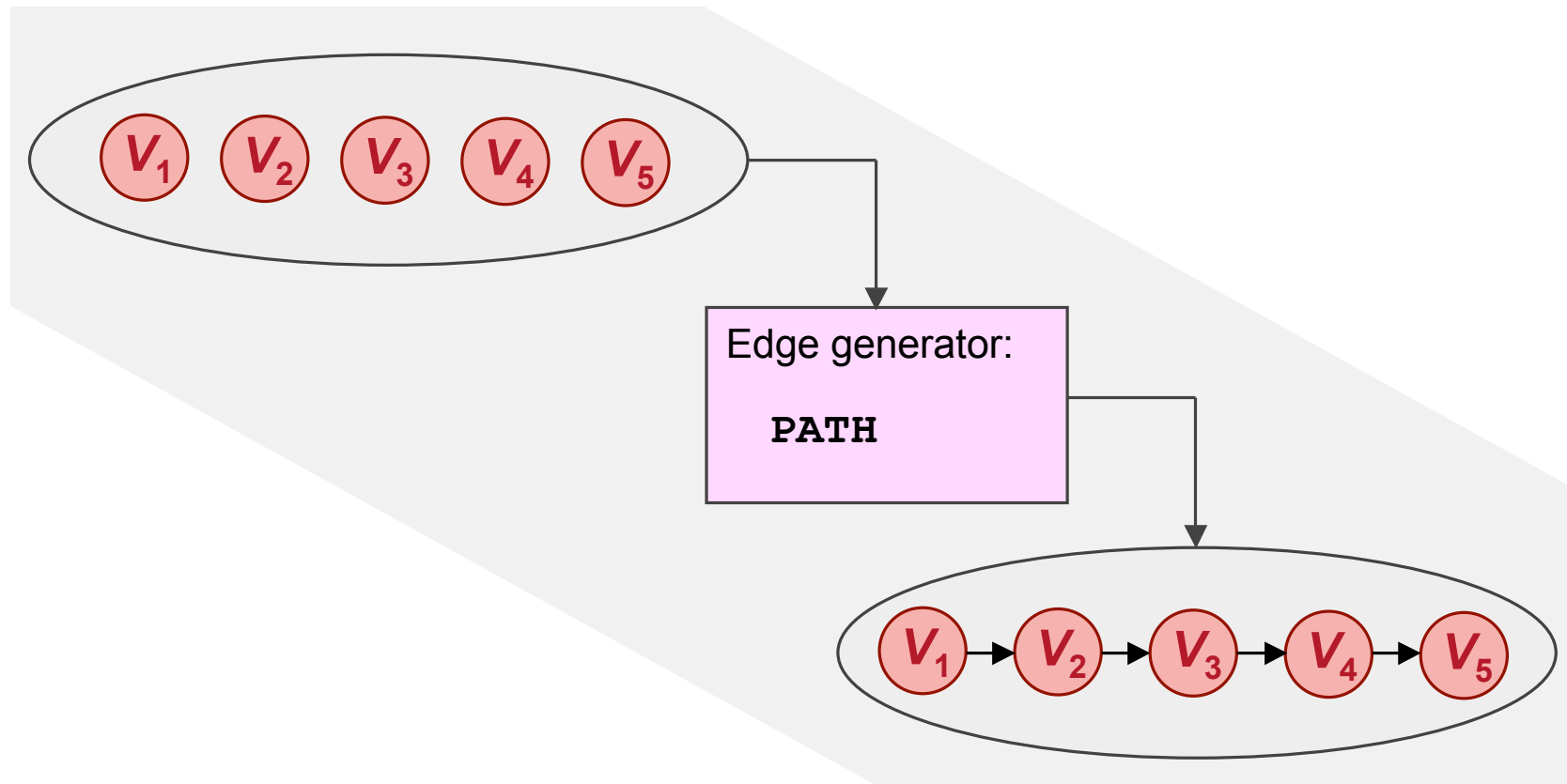
Graph Generation for `cyclic_change(NCHANGE,L,VARIABLES)`

- VERTEX INPUT : *VARIABLES*
- VERTEX GENERATOR : IDENTITY



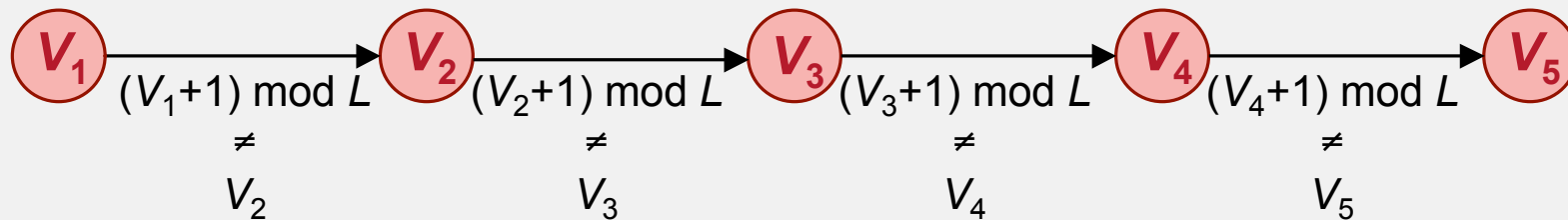
Graph Generation for `cyclic_change(NCHANGE,L,VARIABLES)`

- **EDGE INPUT** : *VARIABLES*
- **EDGE GENERATOR** : **PATH**
- **EDGE ARITY** : 2



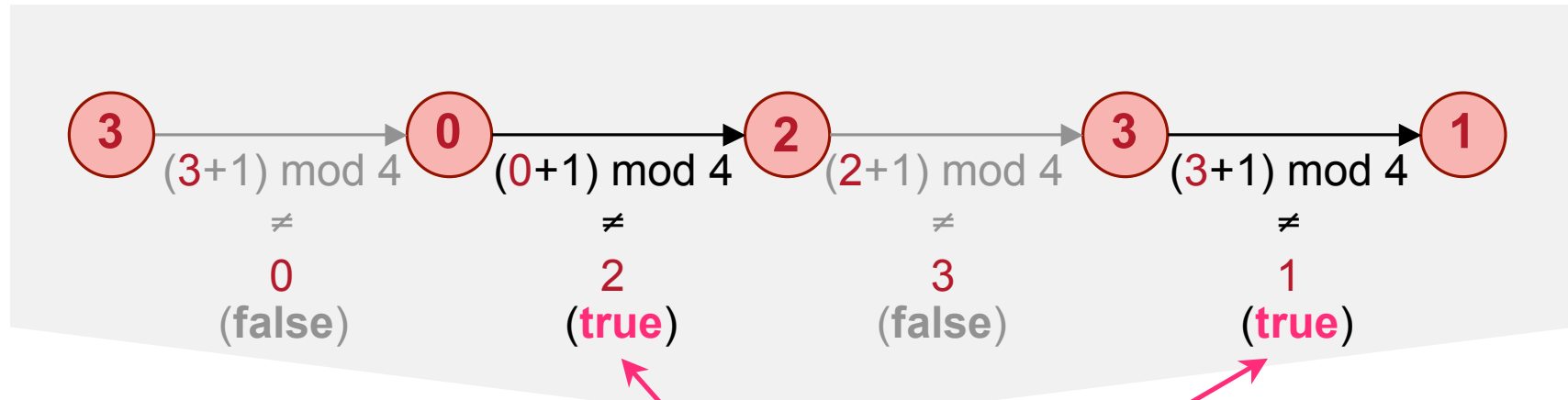
Elementary Constraint for `cyclic_change(NCHANGE,L,VARIABLES)`

- EDGE CONSTRAINT: $(VARIABLES.var[1]+1) \bmod L \neq VARIABLE.var[2]$



Graph Property for `cyclic_change(NCHANGE,L,VARIABLES)`

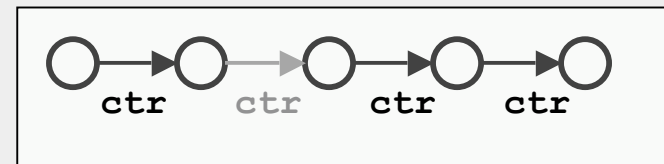
- GRAPH PROPERTY : `NEDGE = NCHANGE`



`cyclic_change(2,4,{var-3,var-0 | var-2,var-3 | var-1})`

"Signature" of the *cardinality-path* family

• VERTEX	GENERATOR :	IDENTITY
• EDGE	GENERATOR :	PATH
• EDGE	ARITY :	$k > 1$
• CONSTRAINT	PROPERTY :	-
• GRAPH	PROPERTY :	NEDGE = <i>var</i>



Constraint pattern

The *cardinality* Operator

The *cardinality-path* Family

- Situation of the *cardinality-path* Family
- ➔ • **Instances of the *cardinality-path* Family**

Filtering Algorithms for *cardinality-path*

- Required Basic Services
- Lower Bound of the Minimum Number of Constraints that Hold
- Pruning According to the Bounds
- Example of Pruning

Conclusion and Perspectives

Arity	Bounds for C and constraint <i>CTR</i>	Member and example of solution
2	$C : 0..n-1$ $V_i \neq V_{i+1}$	change($C, \{V_1, \dots, V_n\}, \neq$) change(3 ,{ 4 , 4 , 3 , 4 , 1 }, \neq)
2	$C : 0..n-1$ $(V_i + 1) \bmod L \neq V_{i+1}$	cyclic_change($C, L, \{V_1, \dots, V_n\}$) cyclic_change(2 , 5 ,{ 2 , 3 , 4 , 0 , 2 , 3 , 1 })
2	$C : 0..n-1$ $ V_i - V_{i+1} > T$	smooth($C, T, \{V_1, \dots, V_n\}$) smooth(1 , 2 ,{ 1 , 3 , 4 , 5 , 2 })
3	$C : 0..n-2$ $V_i = 0 \wedge V_{i+1} = 0 \wedge V_{i+2} \neq 0$	number_of_rest($C, \{V_1, \dots, V_n\}$) number_of_rest(2 ,{ 2 , 0 , 0 , 1 , 1 , 0 , 2 , 0 , 0 , 1 , 2 })
k	$C : a..b$ $low \leq \sum_{j=i}^{i+k-1} V_j \leq up$	relaxed_sliding_sum($a, b, low, up, k, \{V_1, \dots, V_n\}$) relaxed_sliding_sum(3 , 4 , 3 , 7 , 4 ,{ 2 , 4 , 2 , 0 , 0 , 3 , 4 })

The *cardinality* Operator

The *cardinality-path* Family

- Situation of the *cardinality-path* Family
- Instances of the *cardinality-path* Family

Filtering Algorithms for *cardinality-path*



- **Required Basic Services**

- Lower Bound of the Minimum Number of Constraints that Hold
- Pruning According to the Bounds
- Example of Pruning

Conclusion and Perspectives

Required Services

MAIN PRUNING ALGORITHM

- $\text{enforce_CTR}(V_i, \dots, V_{i+k-1})$
- $\text{enforce_NOT_CTR}(V_i, \dots, V_{i+k-1})$
- $\text{create_choice_point}$
- backtrack

ASSUMPTION :

*For constraint CTR or its negation, failure detection should be **independent from the order** in which the constraints are posted*

Organisation of the Algorithms

cardinality_path(**C**, $\{V_1, \dots, V_n\}$, CTR)

FILTERING ALGORITHMS

find **BOUNDS**
for **C**

PROPAGATE
from **C** to $\{V_1, \dots, V_n\}$

The *cardinality* Operator

The *cardinality-path* Family

- Situation of the *cardinality-path* Family
- Instances of the *cardinality-path* Family

Filtering Algorithms for *cardinality-path*

- Required Basic Services
- ➔ • **Lower Bound of the Minimum Number of Constraints that Hold**
- Pruning According to the Bounds
- Example of Pruning

Conclusion and Perspectives

Lower Bound of the Number of Constraints that Hold

CTR: $(V_i+1) \bmod 5 \neq V_{i+1}$

V1:	0,3	↓
V2:	2,3,4	
V3:	0,4	
V4:	0,1,2,3,4	
V5:	0,1,2,3	
V6:	0,2,4	↓
V7:	0,1,2	
V8:	0,1,2	
V9:	0,4	↓

cyclic_change(**C**, 5, {V1,V2,V3,V4,V5,V6,V7,V8,V9})

	V1: 0,3
$(V1+1) \bmod 5 = V2$	V1: 3 V2: 4
$(V2+1) \bmod 5 = V3$	V1: 3 V2: 4 V3: 0
$(V3+1) \bmod 5 = V4$	V1: 3 V2: 4 V3: 0 V4: 1
$(V4+1) \bmod 5 = V5$	V1: 3 V2: 4 V3: 0 V4: 1 V5: 2
$(V5+1) \bmod 5 = V6$	contradiction

	V6: 0,2,4
$(V6+1) \bmod 5 = V7$	V6: 0,4 V7: 0,1
$(V7+1) \bmod 5 = V8$	V6: 0,4 V7: 0,1 V8: 1,2
$(V8+1) \bmod 5 = V9$	contradiction

	V9: 0,4
--	---------

min(C) ≥ 2

The *cardinality* Operator

The *cardinality-path* Family

- Situation of the *cardinality-path* Family
- Instances of the *cardinality-path* Family

Filtering Algorithms for *cardinality-path*

- Required Basic Services
- Lower Bound of the Minimum Number of Constraints that Hold
- ➔ • **Pruning According to the Bounds**
- Example of Pruning

Conclusion and Perspectives

Propagation from C to $\{V_1, \dots, V_n\}$

```
Set min_break to 0.  
Set choice to 1. Create_choice_point.  
for  $i=1$  to  $n-k+1$  do  
  Set before[i] to min_break.  
  if choice=0 then Set choice to 1.  
    Create_choice_point.  
  
  if enforce  $\neg\text{CTR}(V_i, \dots, V_{i+k-1})$  fails then  
    Backtrack. Set choice to 0.  
    Set min_break to min_break + 1.  
  
  Set vbefore[i+k-1] to  $\text{dom}(V_{i+k-1})$ .
```

before[i] ($1 \leq i \leq n-k+1$):

min.number of constraints that hold in
 $\{\text{CTR}(V_1, \dots, V_k), \dots, \text{CTR}(V_{i-1}, \dots, V_{i+k-2})\}$

vbefore[i] ($k \leq i \leq n$):

state of $\text{dom}(V_i)$ after stating constraint
 $\text{CTR}(V_{i-k+1}, \dots, V_i)$

Propagation from C to $\{V_1, \dots, V_n\}$

```
Set min_break to 0.  
Set choice to 1. Create_choice_point.  
for  $i = n - k + 1$  to 1 (step -1) do  
  Set after[ $i$ ] to min_break.  
  if choice=0 then Set choice to 1.  
    Create_choice_point.  
  
  if enforce  $\neg$ CTR( $V_i, \dots, V_{i+k-1}$ ) fails then  
    Backtrack. Set choice to 0.  
    Set min_break to min_break + 1.  
  
  Set vafter[ $i$ ] to dom( $V_i$ ).
```

after[i] ($n-k+1 \geq i \geq 1$):

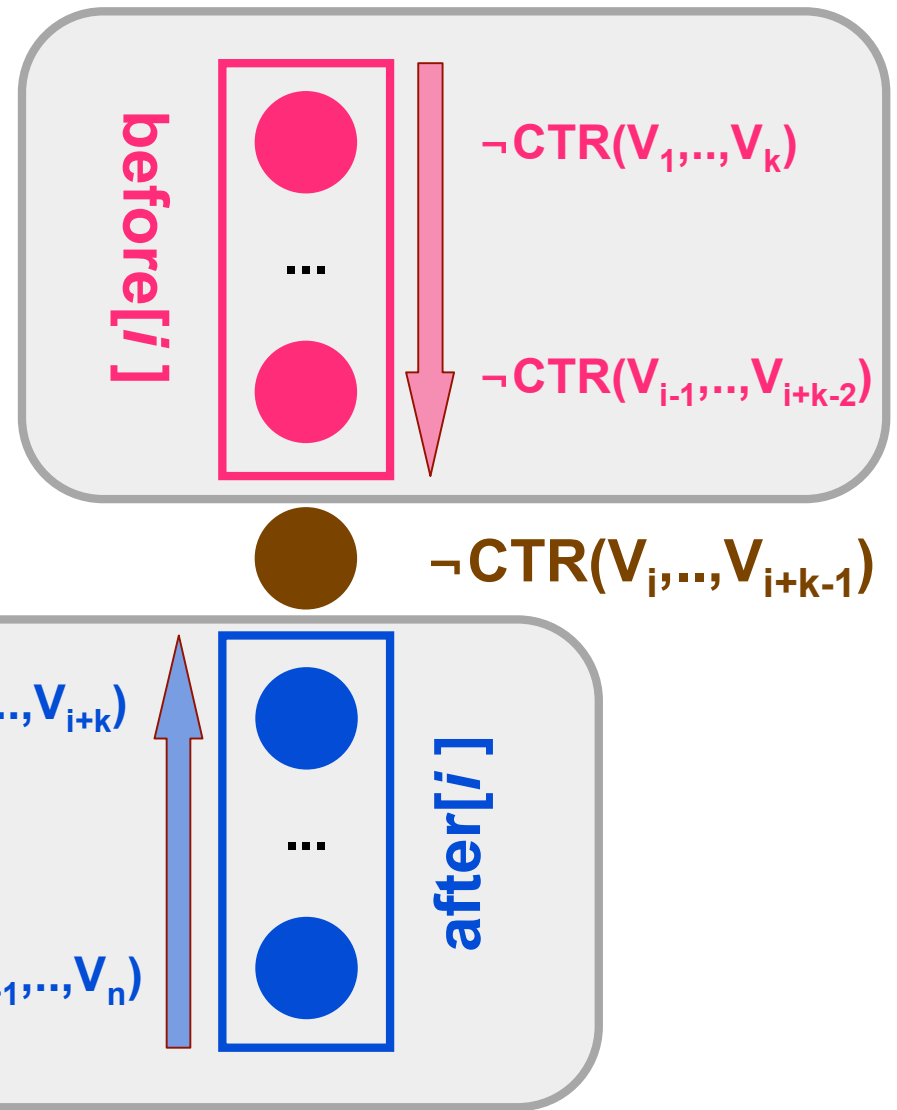
min.number of constraints that hold in
 $\{\text{CTR}(V_{i+1}, \dots, V_{i+k}), \dots, \text{CTR}(V_{n-k+1}, \dots, V_n)\}$

vafter[i] ($n-k+1 \geq i \geq 1$):

state of $\text{dom}(V_i)$ after stating constraint
CTR(V_i, \dots, V_{i+k-1})

First Rule

for $i=1$ to $n-k+1$ do
if $\text{before}[i] + \text{after}[i] + 1 > \max(C)$ then
enforce $\neg \text{CTR}(V_i, \dots, V_{i+k-1})$.



Second Rule

$k \leq i \leq n - k + 1$:

if $\text{before}[i - k + 2] + \text{after}[i - 1] + 2 > \max(C)$

then $V_i \subseteq v\text{before}[i] \cup \text{after}[i]$.

We prove that :

If $V_i \notin v\text{before}[i] \cup \text{after}[i]$

then at least $\text{before}[i - k + 2] + \text{after}[i - 1] + 2$ constraints CTR hold

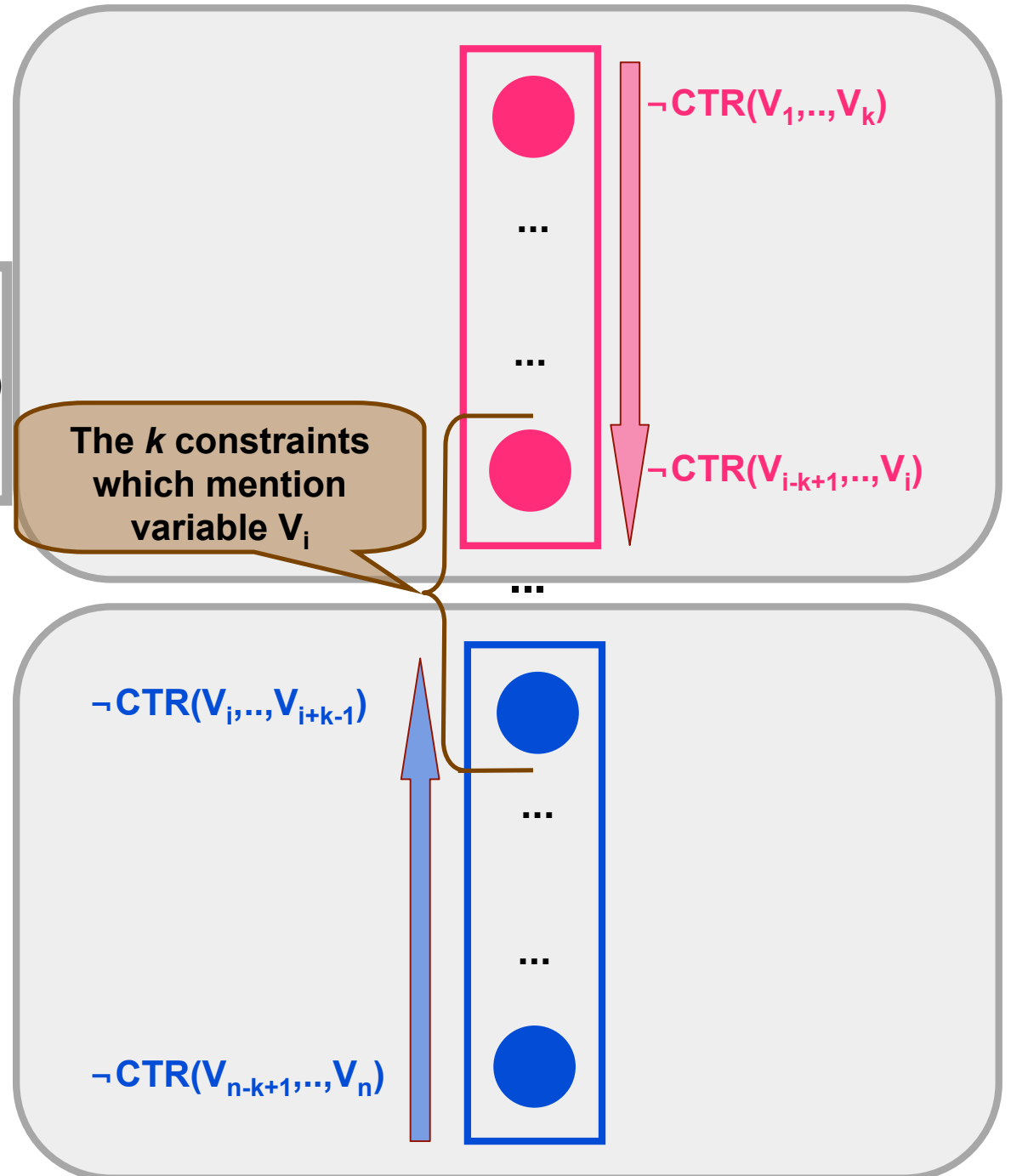
Second Rule

$k \leq i \leq n - k + 1$:

if $\text{before}[i - k + 2] + \text{after}[i - 1] + 2 > \max(C)$

then $V_i \subseteq \text{vbefore}[i] \cup \text{vafter}[i]$.

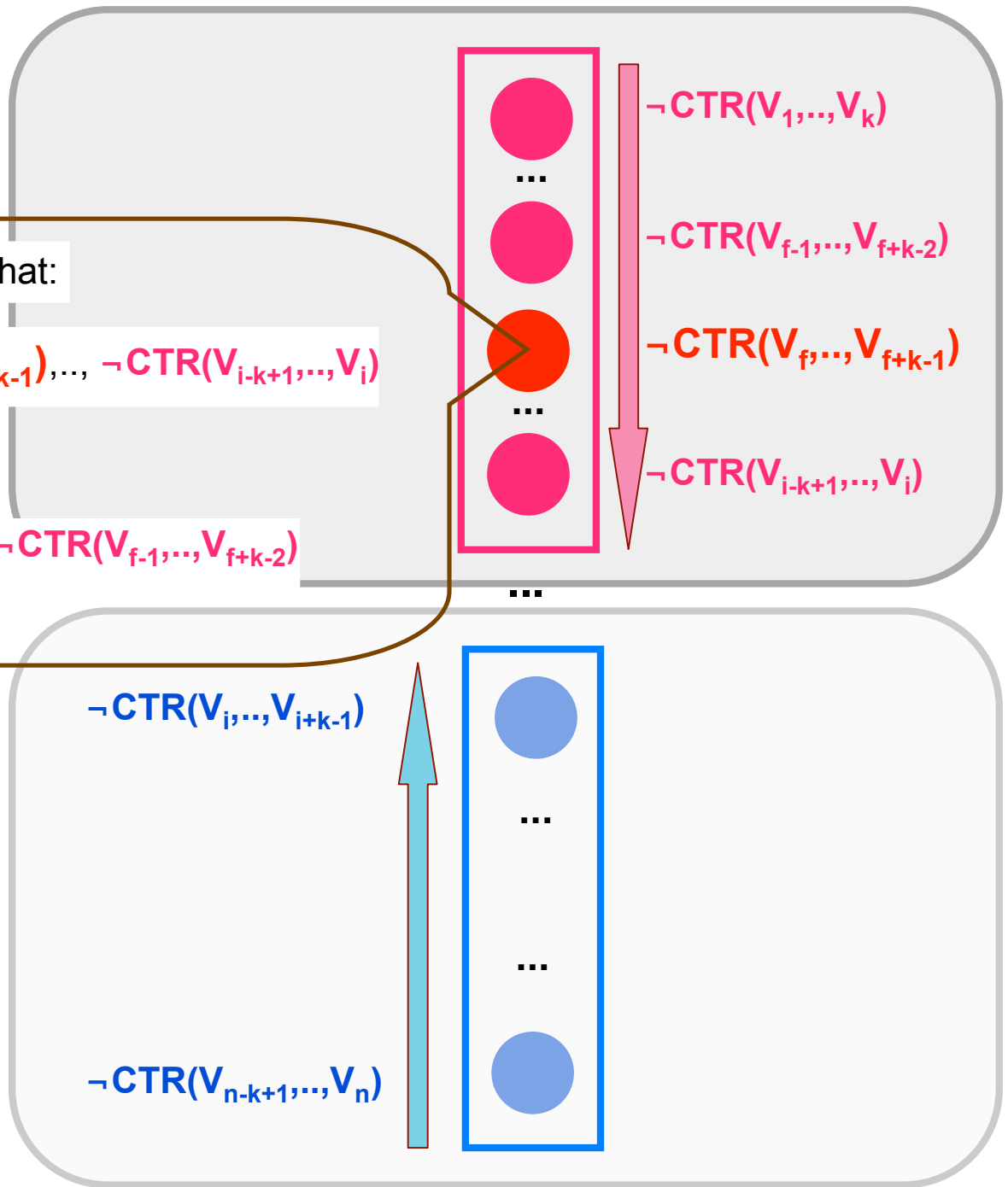
The k constraints which mention variable V_i



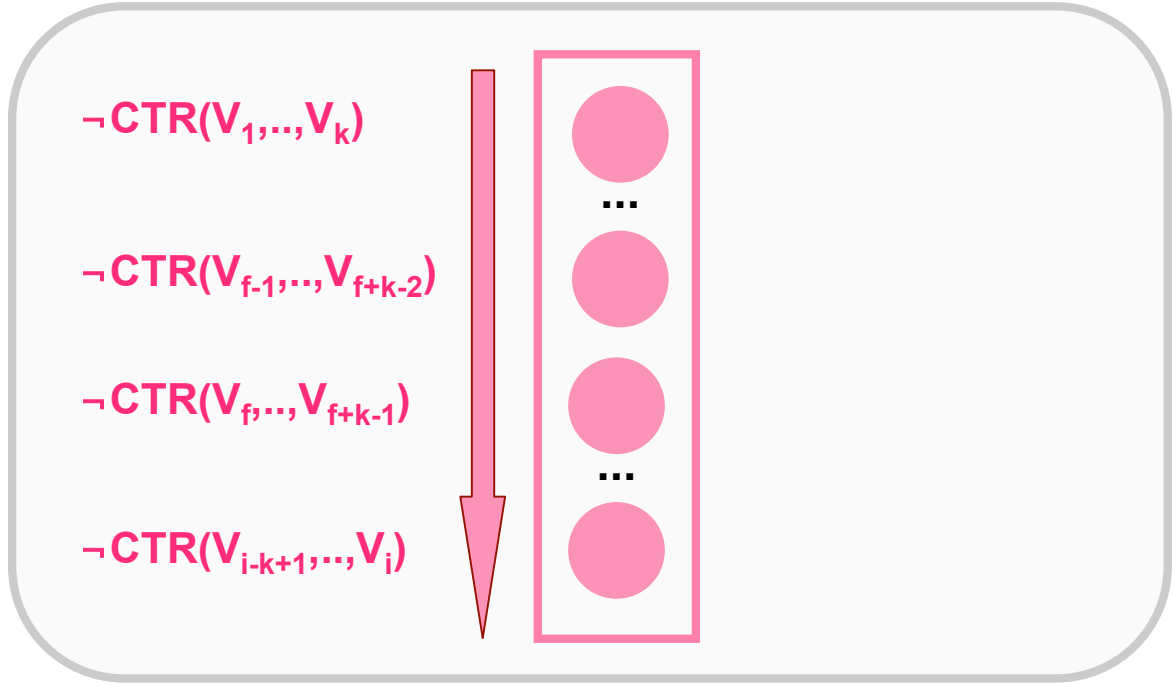
Second Rule

f: smallest value $\leq i-k+1$ such that:

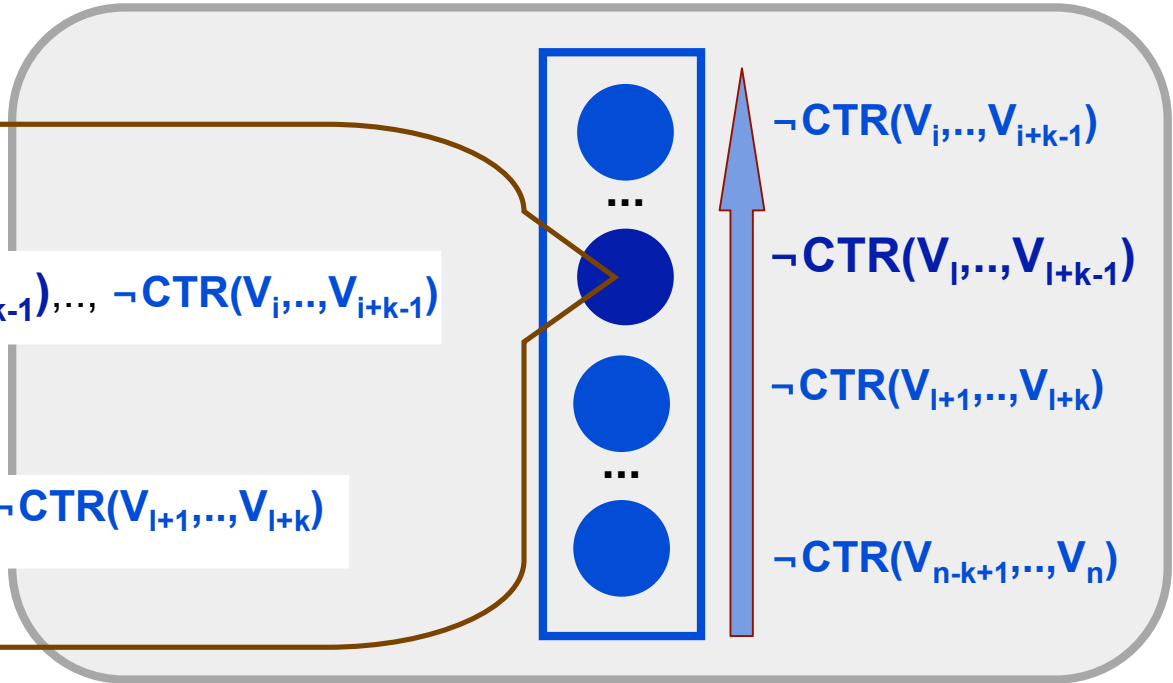
- no failure on $\neg\text{CTR}(V_f, \dots, V_{f+k-1}), \dots, \neg\text{CTR}(V_{i-k+1}, \dots, V_i)$ during first iteration
- **f** = 1 or a failure occurs after stating $\neg\text{CTR}(V_{f-1}, \dots, V_{f+k-2})$



Second Rule



...



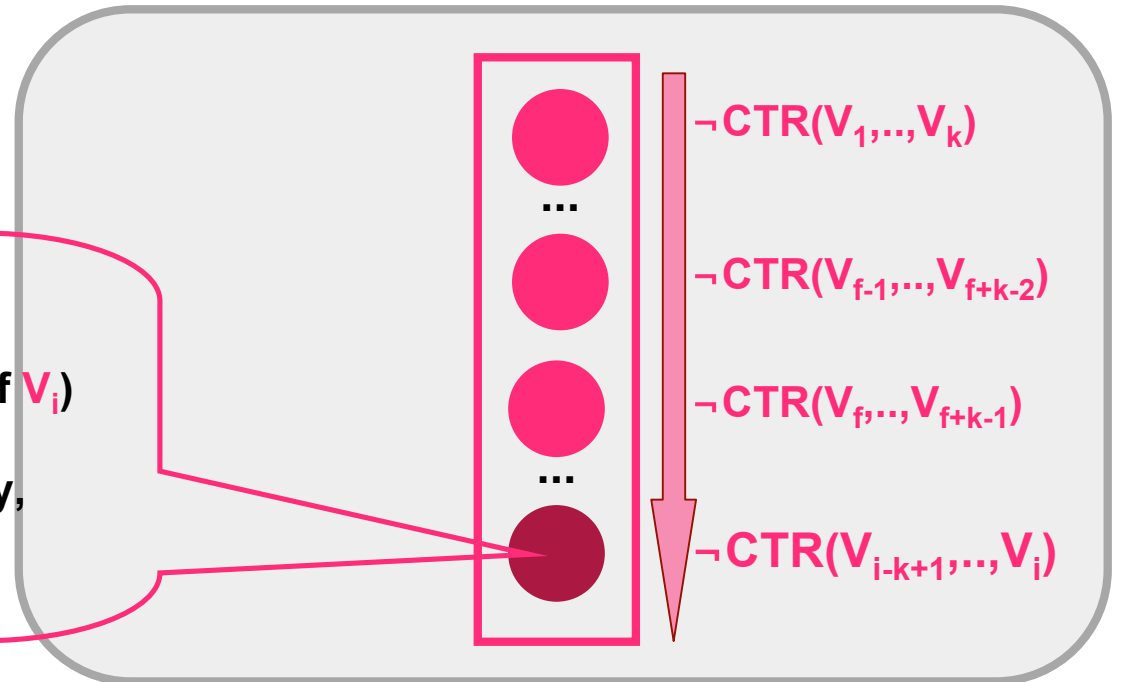
l : largest value $\geq i$ such that:

- no failure on $\neg \text{CTR}(V_l, \dots, V_{l+k-1}), \dots, \neg \text{CTR}(V_i, \dots, V_{i+k-1})$ during first iteration
- $l = n-k+1$ or a failure occurs after stating $\neg \text{CTR}(V_{l+1}, \dots, V_{l+k})$

Second Rule

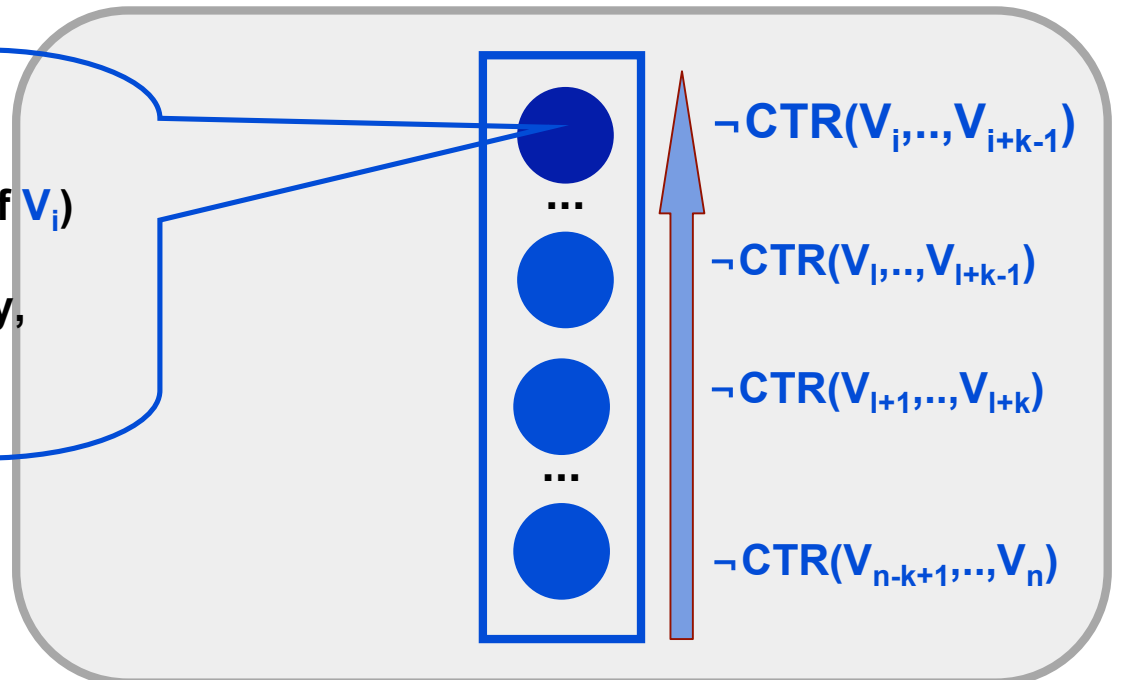
If fail after posting $\neg\text{CTR}(V_{i-k+1}, \dots, V_i)$ then:
 $\text{vbefore}[i] = \text{dom}(V_i)$
 (since we backtrack and restore domain of V_i)

So $\text{dom}(V_i) - \text{vbefore}[i] \cup \text{vafter}[i]$ is empty,
 and we prune nothing.



If fail after posting $\neg\text{CTR}(V_i, \dots, V_{i+k-1})$ then:
 $\text{vafter}[i] = \text{dom}(V_i)$
 (since we backtrack and restore domain of V_i)

So $\text{dom}(V_i) - \text{vbefore}[i] \cup \text{vafter}[i]$ is empty,
 and we prune nothing.



Second Rule

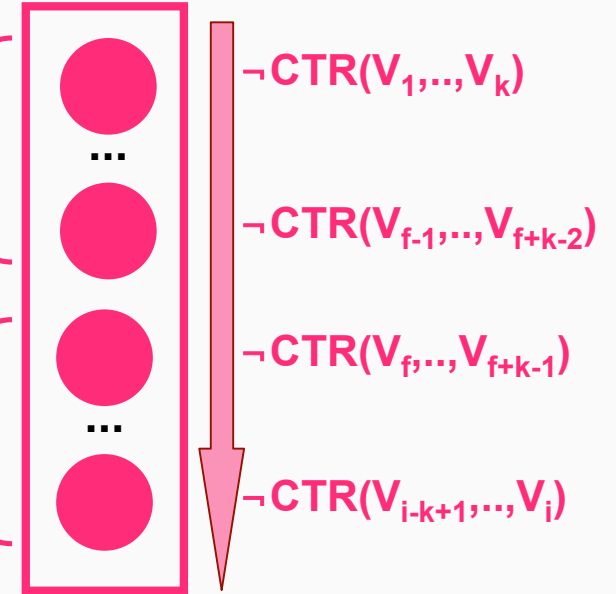
Now let's assume that:

- no fail after posting $\neg\text{CTR}(V_{i-k+1}, \dots, V_i)$,
- $V_i \notin \text{vbefore}[i]$,
- no fail after posting $\neg\text{CTR}(V_i, \dots, V_{i+k-1})$,
- $V_i \notin \text{vafter}[i]$.

And evaluate the **minimum** number of constraints CTR that hold:

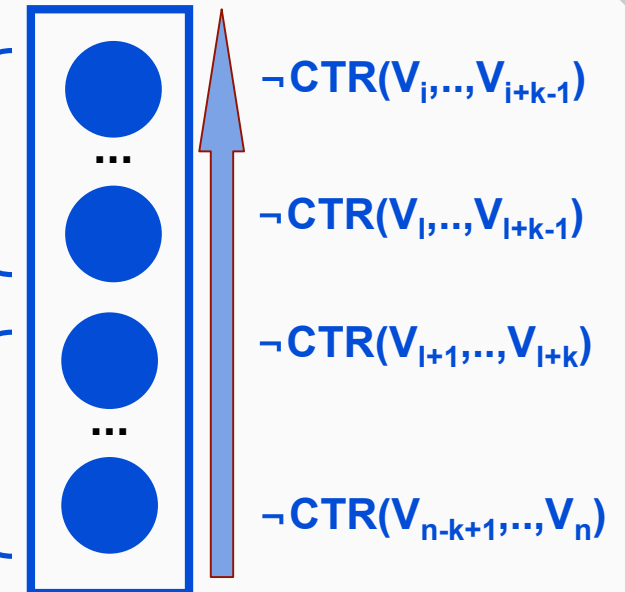
$\geq \text{before}[f] = \text{before}[i-k+2]$
(from definition of f)

≥ 1
(since $V_i \notin \text{vbefore}[i]$)



≥ 1
(since $V_i \notin \text{vafter}[i]$)

$\geq \text{after}[l] = \text{after}[i-1]$
(from definition of l)



CONCLUSION: the minimum is
 $\text{before}[i-k+2] + \text{after}[i-1] + 2$

The *cardinality* Operator

The *cardinality-path* Family

- Situation of the *cardinality-path* Family
- Instances of the *cardinality-path* Family

Filtering Algorithms for *cardinality-path*

- Required Basic Services
- Lower Bound of the Minimum Number of Constraints that Hold
- Pruning According to the Bounds
- ➔ • **Example of Pruning**

Conclusion and Perspectives

Pruning According to the Maximum Number of Constraints that Hold

$$(V1+1) \bmod 5 = V2$$

3	4	4	4	3	4	2	2	4
0	3	0	3	2	2	1	1	0
	2		2	1	0	0	0	
			1	0				
			0					
V1	V2	V3	V4	V5	V6	V7	V8	V9

vbefore [2..9]		4 3 2						
before [1..8]	0							
after [1..8]								
vafter [1..8]								

Pruning According to the Maximum Number of Constraints that Hold

$$(V2+1) \bmod 5 = V3$$

3	4	4	4	3	4	2	2	4
0	3	0	3	2	2	1	1	0
	2		2	1	0	0	0	
			1	0				
			0					
V1	V2	V3	V4	V5	V6	V7	V8	V9

vbefore [2..9]		4 3 2	4 0					
before [1..8]	0	0						
after [1..8]								
vafter [1..8]								

Pruning According to the Maximum Number of Constraints that Hold

$$(V3+1) \bmod 5 = V4$$

3	4	4	4	3	4	2	2	4
0	3	0	3	2	2	1	1	0
	2		2	1	0	0	0	
			1	0				
			0					
V1	V2	V3	V4	V5	V6	V7	V8	V9

vbefore [2..9]		4 3 2	4 0	4 3 2 1 0				
before [1..8]	0	0	0					
after [1..8]								
vafter [1..8]								

Pruning According to the Maximum Number of Constraints that Hold

fails

$(V8+1) \bmod 5 = V9$

	V1	V2	V3	V4	V5	V6	V7	V8	V9
	3	4	4	4	3	4	2	2	4
	0	3	0	3	2	2	1	1	0
		2		2	1	0	0	0	
				1	0				
				0					
	V1	V2	V3	V4	V5	V6	V7	V8	V9
vbefore [2..9]		4 3 2	4 0	4 3 2 1 0	3 2 1 0	4 3 0	2 1 0	2 1 0	4 0
before [1..8]	0	0	0	0	0	1	1	1	
after [1..8]								0	
vafter [1..8]								2 1 0	

Pruning According to the Maximum Number of Constraints that Hold

$$(V7+1) \bmod 5 = V8$$

		3	4	4	4	3	4	2	2	4
		0	3	0	3	2	2	1	1	0
			2		2	1	0	0	0	
					1	0				
					0					
		V1	V2	V3	V4	V5	V6	V7	V8	V9
vbefore [2..9]			4	4	4	3	4	2	2	4
			3	0	3	2	2	1	1	0
before [1..8]					2	1	0	0	0	
					1	0				
before [1..8]		0	0	0	0	0	1	1	1	
after [1..8]								1	0	
vafter [1..8]								2	2	
								1	1	
								0	0	

Pruning According to the Maximum Number of Constraints that Hold

$$(V6+1) \bmod 5 = V7$$

		3	4	4	4	3	4	2	2	4
		0	3	0	3	2	2	1	1	0
			2		2	1	0	0	0	
					1	0				
					0					
		V1	V2	V3	V4	V5	V6	V7	V8	V9
vbefore [2..9]			4	4	4	3	4	2	2	4
			3	0	3	2	2	1	1	0
			2		2	1	0	0	0	
					1	0				
before [1..8]		0	0	0	0	0	1	1	1	
after [1..8]							1	1	0	
vafter [1..8]							4	2	2	
							2	1	1	
							0	0	0	

Pruning According to the Maximum Number of Constraints that Hold

$$(V5+1) \bmod 5 = V6$$

		3	4	4	4	3	4	2	2	4
		0	3	0	3	2	2	1	1	0
			2		2	1	0	0	0	
					1	0				
					0					
		V1	V2	V3	V4	V5	V6	V7	V8	V9
vbefore [2..9]			4	4	4	3	4	2	2	4
			3	0	3	2	2	1	1	0
before [1..8]			2		2	1	0	0	0	
					1	0				
after [1..8]		0	0	0	0	0	1	1	1	
vafter [1..8]						1	1	1	0	
						3	4	2	2	
					2	2	2	1	1	
					1	0	0	0	0	
					0					

Pruning According to the Maximum Number of Constraints that Hold

$$(V4+1) \bmod 5 = V5$$

		3	4	4	4	3	4	2	2	4
		0	3	0	3	2	2	1	1	0
			2		2	1	0	0	0	
					1	0				
					0					
		V1	V2	V3	V4	V5	V6	V7	V8	V9
vbefore	[2..9]		4 3 2	4 0	4 3 2 1 0	3 2 1 0	4 3 0	2 1 0	2 1 0	4 0
before	[1..8]	0	0	0	0	0	1	1	1	
after	[1..8]				1	1	1	1	0	
vafter	[1..8]				4 3 2 1 0	3 2 1 0	4 2 0	2 1 0	2 1 0	

Pruning According to the Maximum Number of Constraints that Hold

fails

$$(V3+1) \bmod 5 = V4$$

	V1	V2	V3	V4	V5	V6	V7	V8	V9
	3	4	4	4	3	4	2	2	4
	0	3	0	3	2	2	1	1	0
		2		2	1	0	0	0	
				1	0				
				0					
	V1	V2	V3	V4	V5	V6	V7	V8	V9
vbefore [2..9]		4 3 2	4 0	4 3 2 1 0	3 2 1 0	4 3 0	2 1 0	2 1 0	4 0
before [1..8]	0	0	0	0	0	1	1	1	
after [1..8]			1	1	1	1	1	0	
vafter [1..8]			4 0	4 3 2 1 0	3 2 1 0	4 3 0	2 1 0	2 1 0	

Pruning According to the Maximum Number of Constraints that Hold

$$(V2+1) \bmod 5 = V3$$

		3	4	4	4	3	4	2	2	4
		0	3	0	3	2	2	1	1	0
			2		2	1	0	0	0	
					1	0				
					0					
		V1	V2	V3	V4	V5	V6	V7	V8	V9
vbefore [2..9]			4	4	4	3	4	2	2	4
			3	0	3	2	2	1	1	0
			2		2	1	0	0	0	
					1	0				
before [1..8]		0	0	0	0	0	1	1	1	
after [1..8]			2	1	1	1	1	1	0	
vafter [1..8]			4	4	4	3	4	2	2	
			3	0	3	2	2	1	1	
			2		2	1	0	0	0	
					1	0				

Pruning According to the Maximum Number of Constraints that Hold

$$(V1+1) \bmod 5 = V2$$

		3	4	4	4	3	4	2	2	4
		0	3	0	3	2	2	1	1	0
			2		2	1	0	0	0	
					1	0				
					0					
		V1	V2	V3	V4	V5	V6	V7	V8	V9
vbefore [2..9]			4	4	4	3	4	2	2	4
			3	0	3	2	2	1	1	0
			2		2	1	0	0	0	
					1	0				
before [1..8]		0	0	0	0	0	1	1	1	
after [1..8]		2	2	1	1	1	1	1	0	
vafter [1..8]		3	4	4	4	3	4	2	2	
		0	3	0	3	2	2	1	1	
			2		2	1	0	0	0	
					1	0				
					0					

Pruning According to the Maximum Number of Constraints that Hold

Rule 1 IF $\text{before}[i] + \text{after}[i] + 1 > \max(C)$ THEN enforces not $\text{ctr}(V_i, \dots, V_{i+k-1})$

	V1	V2
vbefore[2..9]		4 3 2
before [1..8]	0	0
after [1..8]	2	2
vafter [1..8]	3 0	4 3 2

Let's assume $\max(C)=2$

Since $\text{before}[1] + \text{after}[1] + 1 > \max(C)$
enforces $(V_1 + 1) \bmod 5 = V_2$:

- $V_1 = 3$
- $V_2 = 4$

Pruning According to the Maximum Number of Constraints that Hold

Rule 2 IF $\text{before}[i-k+2] + \text{after}[i-1] + 2 > \max(C)$
 THEN V_i in $v\text{before}[i] \cup v\text{after}[i]$

	3	4	4
	0	3	0
		2	
	V1	V2	V3
$v\text{before}[2..9]$		4	4
		3	0
		2	
$\text{before}[1..8]$	0	0	0
$\text{after}[1..8]$	2	2	1
$v\text{after}[1..8]$	3	4	4
	0	3	0
		2	

Let's assume $\max(C)=3$

Since $\text{before}[2] + \text{after}[1] + 2 > \max(C)$:

- removes 2 from V_2

Pruning According to the Maximum Number of Constraints that Hold

Two more constraints will fail if $V_2=2$

	V1	V2	V3
	3 0	2	4 0
vbefore [2..9]		4 3 2	4 0
before [1..8]	0	0	0
after [1..8]	2	2	1
vafter [1..8]	3 0	4 3 2	4 0

Let's assume $\max(C)=3$

Since $\text{before}[2] + \text{after}[1] + 2 > \max(C)$:

- removes 2 from V_2

Conclusion and Perspectives

What was done :

- A **generic** propagation algorithm for another **family** of constraints
- A method which is also **useful** for **other** global constraints patterns

What remains to do :

- Take into account the fact that we have the **same** constraint
- Consider **properties** of that constraint