5.172 group

DESCRIPTION LINKS GRAPH AUTOMATON

Origin CHIP

Constraint

$$\texttt{group}\begin{pmatrix}\texttt{NGROUP},\\ \texttt{MIN\_SIZE},\\ \texttt{MAX\_SIZE},\\ \texttt{MIN\_DIST},\\ \texttt{MAX\_DIST},\\ \texttt{NVAL},\\ \texttt{VARIABLES},\\ \texttt{VALUES}\end{pmatrix}$$

Arguments

NGROUP	:	dvar
MIN_SIZE	:	dvar
MAX_SIZE	:	dvar
MIN_DIST	:	dvar
MAX_DIST	:	dvar
NVAL	:	dvar
VARIABLES	:	collection(var−dvar)
VALUES	:	collection(val−int)

Restrictions

$\texttt{NGROUP} \geq 0$
$\texttt{MIN\_SIZE} \geq 0$
$\texttt{MAX\_SIZE} \geq \texttt{MIN\_SIZE}$
$\texttt{MIN\_DIST} \geq 0$
$\texttt{MAX\_DIST} \geq \texttt{MIN\_DIST}$
$\texttt{MAX\_DIST} \leq |\texttt{VARIABLES}|$
$\texttt{NVAL} \geq \texttt{MAX\_SIZE}$
$\texttt{NVAL} \geq \texttt{NGROUP}$
$\texttt{NVAL} \leq |\texttt{VARIABLES}|$
required(VARIABLES, var)
required(VALUES, val)
distinct(VALUES, val)

Purpose

Let n be the number of variables of the collection `VARIABLES`. Let $X_i, X_{i+1}, \ldots, X_j$ ($1 \leq i \leq j \leq n$) be consecutive variables of the collection of variables `VARIABLES` such that all the following conditions simultaneously apply:

- All variables $X_i, \ldots, X_j$ take their value in the set of values `VALUES`,
- $i = 1$ or X_{i-1} does not take a value in `VALUES`,
- $j = n$ or X_{j+1} does not take a value in `VALUES`.

We call such a sequence of variables a *group*. Similarly an *anti-group* is a maximum sequence of variables that are not assigned any value from `VALUES`. The constraint `group` is true if all the following conditions hold:

- There are exactly `NGROUP` groups of variables,
- `MIN_SIZE` is the number of variables of the smallest group,
- `MAX_SIZE` is the number of variables of the largest group,
- `MIN_DIST` is the number of variables of the smallest anti-group,
- `MAX_DIST` is the number of variables of the largest anti-group,
- `NVAL` is the number of variables that take their value in the set of values `VALUES`.

Example

$$(2, 1, 2, 2, 4, 3, \langle 2, 8, 1, 7, 4, 5, 1, 1, 1\rangle, \langle 0, 2, 4, 6, 8\rangle)$$

Given the fact that groups are formed by even values in $\{0, 2, 4, 6, 8\}$ (i.e., values expressed by the `VALUES` collection), the `group` constraint holds since:

- Its first argument, `NGROUP`, is set to value 2 since the sequence $2\ 8\ 1\ 7\ 4\ 5\ 1\ 1\ 1$ contains two groups of even values (i.e., group $2\ 8$ and group 4).
- Its second argument, `MIN_SIZE`, is set to value 1 since the smallest group of even values involves only a single value (i.e., value 4).
- Its third argument, `MAX_SIZE`, is set to value 2 since the largest group of even values involves two values (i.e., group $2\ 8$).
- Its fourth argument, `MIN_DIST`, is set to value 2 since the smallest anti-group involves two values (i.e., anti-group $1\ 7$).
- Its fifth argument, `MAX_DIST`, is set to value 4 since the largest anti-group involves four values (i.e., anti-group $5\ 1\ 1\ 1$).
- Its sixth argument, `NVAL`, is set to value 3 since the total number of even values of the sequence $2\ 8\ 1\ 7\ 4\ 5\ 1\ 1\ 1$ is equal to 3 (i.e., values 2, 8 and 4).

All solutions

Figure 5.386 gives all solutions to the following non ground instance of the `group` constraint: `NGROUP` $\in [2, 3]$, `MIN_SIZE` $\in [3, 4]$, `MAX_SIZE` $\in [3, 5]$, `MIN_DIST` $\in [1, 2]$, `MAX_DIST` $\in [1, 2]$, `NVAL` $\in [5, 6]$, $\mathtt{V}_1 \in [0, 1]$, $\mathtt{V}_2 \in [0, 1]$, $\mathtt{V}_3 \in [0, 1]$, $\mathtt{V}_4 \in [0, 1]$, $\mathtt{V}_5 \in [0, 1]$, $\mathtt{V}_6 \in [0, 1]$, $\mathtt{V}_7 \in [0, 1]$, $\mathtt{V}_8 \in [0, 1]$, $\mathtt{V}_9 \in [0, 1]$, `group`(`NGROUP`, `MIN_SIZE`, `MAX_SIZE`, `MIN_DIST`, `MAX_DIST`, `NVAL`, $\langle \mathtt{V}_1, \mathtt{V}_2, \mathtt{V}_3, \mathtt{V}_4, \mathtt{V}_5, \mathtt{V}_6, \mathtt{V}_7, \mathtt{V}_8, \mathtt{V}_9\rangle, \langle 1\rangle$),

① $(2, 3, 3, 1, 2, 6, \langle 0, 0, 1, 1, 1, 0, 1, 1, 1 \rangle, \langle 1 \rangle)$
② $(2, 3, 3, 1, 2, 6, \langle 0, 1, 1, 1, 0, 0, 1, 1, 1 \rangle, \langle 1 \rangle)$
③ $(2, 3, 3, 1, 1, 6, \langle 0, 1, 1, 1, 0, 1, 1, 1, 0 \rangle, \langle 1 \rangle)$
④ $(2, 3, 3, 1, 2, 6, \langle 1, 1, 1, 0, 0, 1, 1, 1, 0 \rangle, \langle 1 \rangle)$
⑤ $(2, 3, 3, 1, 2, 6, \langle 1, 1, 1, 0, 1, 1, 1, 0, 0 \rangle, \langle 1 \rangle)$

Figure 5.386: All solutions corresponding to the non ground example of the `group` constraint of the **All solutions** slot

Typical

`NGROUP` > 0
`MIN_SIZE` > 0
`MAX_SIZE` $>$ `MIN_SIZE`
`MIN_DIST` > 0
`MAX_DIST` $>$ `MIN_DIST`
`MAX_DIST` $<$ |`VARIABLES`|
`NVAL` $>$ `MAX_SIZE`
`NVAL` $>$ `NGROUP`
`NVAL` $<$ |`VARIABLES`|
|`VARIABLES`| > 1
range(`VARIABLES.var`) > 1
|`VALUES`| > 0
|`VARIABLES`| $>$ |`VALUES`|

Symmetries

- Items of `VARIABLES` can be reversed.
- Items of `VALUES` are permutable.
- An occurrence of a value of `VARIABLES.var` that belongs to `VALUES.val` (resp. does not belong to `VALUES.val`) can be replaced by any other value in `VALUES.val` (resp. not in `VALUES.val`).

Arg. properties

- Functional dependency: `NGROUP` determined by `VARIABLES` and `VALUES`.
- Functional dependency: `MIN_SIZE` determined by `VARIABLES` and `VALUES`.
- Functional dependency: `MAX_SIZE` determined by `VARIABLES` and `VALUES`.
- Functional dependency: `MIN_DIST` determined by `VARIABLES` and `VALUES`.
- Functional dependency: `MAX_DIST` determined by `VARIABLES` and `VALUES`.
- Functional dependency: `NVAL` determined by `VARIABLES` and `VALUES`.

Usage

A typical use of the `group` constraint in the context of timetabling is as follow: The value of the i^{th} variable of the `VARIABLES` collection corresponds to the type of shift (i.e., night, morning, afternoon, rest) performed by a specific person on day i. A complete period of work is represented by the variables of the `VARIABLES` collection. In this context the `group` constraint expresses for a person:

- The number of periods of consecutive night-shift during a complete period of work.
- The total number of night-shift during a complete period of work.
- The maximum number of allowed consecutive night-shift.
- The minimum number of days, which do not correspond to a night-shift, between two consecutive sequences of night-shift.

Remark For this constraint we use the possibility to express directly more than one constraint on the parameters of the final graph we want to obtain. For more propagation, it is crucial to keep this in a single constraint, since strong relations relate the different parameters of a graph. This constraint is very similar to the `group` constraint introduced in **CHIP**, except that here, the `MIN_DIST` and `MAX_DIST` constraints apply also for the two borders: we cannot start or end with a group of k consecutive variables that take their values outside `VALUES` and such that k is less than `MIN_DIST` or k is greater than `MAX_DIST`.

See also **common keyword:** `change_continuity`, `full_group` *(timetabling constraint,sequence)*, `global_contiguity` *(sequence)*, `group_skip_isolated_item` *(timetabling constraint,sequence)*, `multi_global_contiguity` *(sequence)*, `pattern`, `stretch_circuit` *(timetabling constraint)*, `stretch_path` *(timetabling constraint,sequence)*.

shift of concept: `consecutive_groups_of_ones`.

used in graph description: `in`, `not_in`.

Keywords **characteristic of a constraint:** automaton, automaton with counters, automaton with same input symbol.

combinatorial object: sequence.

constraint arguments: reverse of a constraint, pure functional dependency.

constraint network structure: alpha-acyclic constraint network(2), alpha-acyclic constraint network(3).

constraint type: timetabling constraint.

filtering: glue matrix.

final graph structure: connected component, vpartition, consecutive loops are connected.

modelling: functional dependency.

Arc input(s)	`VARIABLES`
Arc generator	$PATH \mapsto$ `collection(variables1, variables2)` $LOOP \mapsto$ `collection(variables1, variables2)`
Arc arity	2
Arc constraint(s)	• `in(variables1.var, VALUES)` • `in(variables2.var, VALUES)`
Graph property(ies)	• **NCC**= `NGROUP` • **MIN_NCC**= `MIN_SIZE` • **MAX_NCC**= `MAX_SIZE` • **NVERTEX**= `NVAL`

Arc input(s)	`VARIABLES`
Arc generator	$PATH \mapsto$ `collection(variables1, variables2)` $LOOP \mapsto$ `collection(variables1, variables2)`
Arc arity	2
Arc constraint(s)	• `not_in(variables1.var, VALUES)` • `not_in(variables2.var, VALUES)`
Graph property(ies)	• **MIN_NCC**= `MIN_DIST` • **MAX_NCC**= `MAX_DIST`

Graph model

We use two graph constraints for modelling the `group` constraint: a first one for specifying the constraints on `NGROUP`, `MIN_SIZE`, `MAX_SIZE` and `NVAL`, and a second one for stating the constraints on `MIN_DIST` and `MAX_DIST`. In order to generate the initial graph related to the first graph constraint we use:

- The arc generators $PATH$ and $LOOP$,
- The binary constraint `variables1.var` $\in$ `VALUES` $\wedge$ `variables2.var` $\in$ `VALUES`.

On the first graph constraint of the **Example** slot this produces an initial graph depicted in part (A) of Figure 5.387. We use $PATH$ $LOOP$ and the binary constraint `variables1.var` $\in$ `VALUES` $\wedge$ `variables2.var` $\in$ `VALUES` in order to catch the two following situations:

- A binary constraint has to be used in order to get the notion of group: *Consecutive* variables that take their value in `VALUES`.
- If we only use $PATH$ then we would lose the groups that are composed from a single variable since the predecessor and the successor arc would be destroyed; this is why we use also the $LOOP$ arc generator.

Part (B) of Figure 5.387 shows the final graph associated with the first graph constraint of the **Example** slot. Since we use the **NVERTEX** graph property, the vertices of the final graph are stressed in bold. In addition, since we use the **MIN_NCC** and the **MAX_NCC** graph properties, we also show the smallest and largest connected components of the final graph.

The `group` constraint of the **Example** slot holds since:

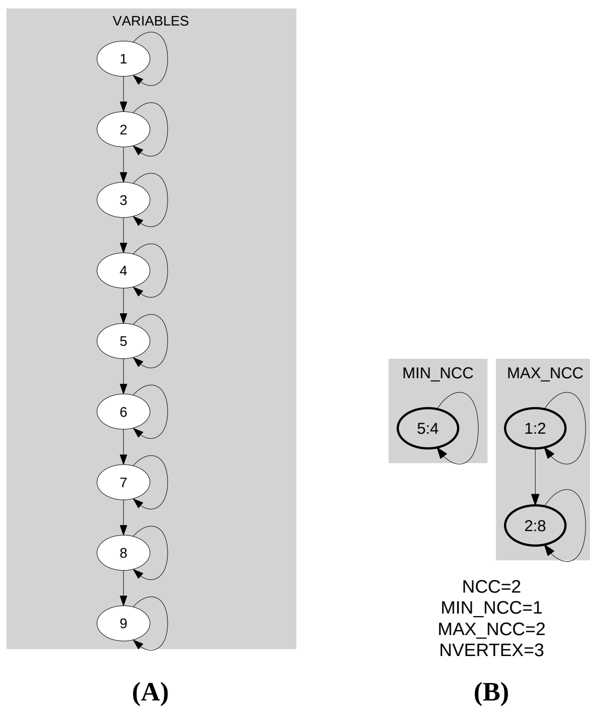


Figure 5.387: Initial and final graph of the `group` constraint

- The final graph of the first graph constraint has two connected components. Therefore the number of groups NGROUP is equal to two.
- The number of vertices of the smallest connected component of the final graph of the first graph constraint is equal to 1. Therefore MIN_SIZE is equal to 1.
- The number of vertices of the largest connected component of the final graph of the first graph constraint is equal to 2. Therefore MAX_SIZE is equal to 2.
- The number of vertices of the smallest connected component of the final graph of the second graph constraint is equal to 2. Therefore MIN_DIST is equal to 2.
- The number of vertices of the largest connected component of the final graph of the second graph constraint is equal to 4. Therefore MAX_DIST is equal to 4.
- The number of vertices of the final graph of the first graph constraint is equal to three. Therefore NVAL is equal to 3.

Automaton Figures 5.388, 5.390, 5.393, 5.395, 5.397 and 5.399 depict the different automata associated with the `group` constraint. For the automata that respectively compute `NGROUP`, `MIN_SIZE`, `MAX_SIZE`, `MIN_DIST`, `MAX_DIST` and `NVAL` we have a 0-1 signature variable $\mathtt{S}_i$ for each variable $\mathtt{VAR}_i$ of the collection `VARIABLES`. The following signature constraint links $\mathtt{VAR}_i$ and $\mathtt{S}_i$: $\mathtt{VAR}_i \in \mathtt{VALUES} \Leftrightarrow \mathtt{S}_i$.

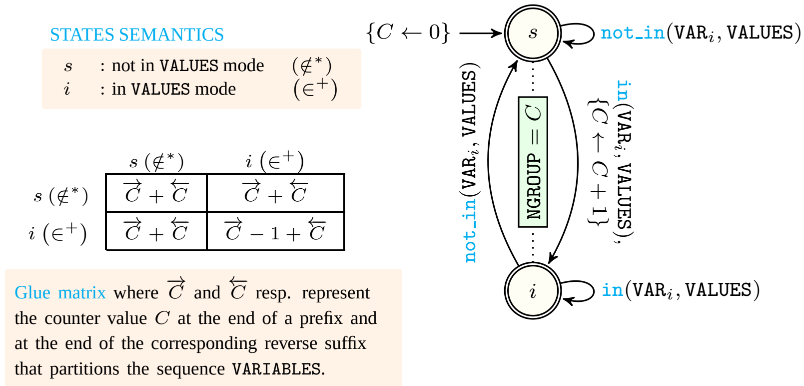


Figure 5.388: Automaton for the `NGROUP` argument of the `group` constraint and its glue matrix

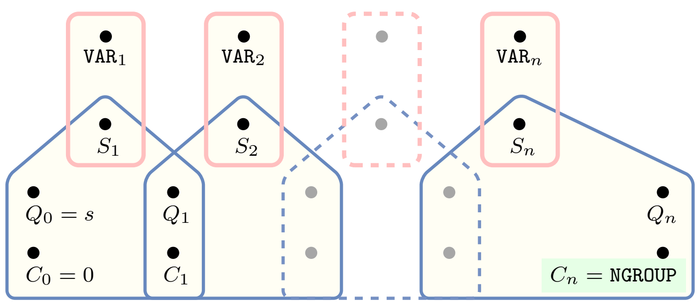
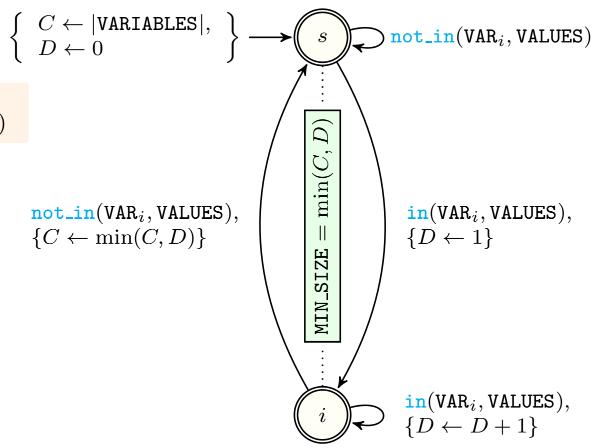


Figure 5.389: Hypergraph of the reformulation corresponding to the automaton (with one counter) of the `NGROUP` argument of the `group` constraint (since all states of the automaton are accepting there is no restriction on the last variable Q_n)

STATES SEMANTICS

s	: not in `VALUES` mode	$(\notin^*)$
i	: in `VALUES` mode	$(\in^+)$



Glue matrix where $\overrightarrow{C}$, $\overrightarrow{D}$ and $\overleftarrow{C}$, $\overleftarrow{D}$ resp. represent the counters values C, D at the end of a prefix and at the end of the corresponding reverse suffix that partitions the sequence `VARIABLES`.

	$s\ (\notin^*)$	$i\ (\in^+)$
$s\ (\notin^*)$	$\min(\overrightarrow{C}, \overrightarrow{D}+\overleftarrow{D}, \overleftarrow{C})$	$\min(\overrightarrow{C}, \overleftarrow{D}, \overleftarrow{C})$
$i\ (\in^+)$	$\min(\overrightarrow{C}, \overrightarrow{D}, \overleftarrow{C})$	$\min(\overrightarrow{C}, \overrightarrow{D}+\overleftarrow{D}, \overleftarrow{C})$

Figure 5.390: Automaton for the `MIN_SIZE` argument of the `group` constraint and its glue matrix

`group`(`MIN_SIZE` $= 2, \langle 0,1,1,1,0,0,1,1,0,1,1,1,1\rangle, \mathcal{V} = \{1\}$)

		0	1	1	1	0	0	1
		$\notin$	$\in$	$\in$	$\in$	$\notin$	$\notin$	$\in$
i	0	1	2	3	4	5	6	7
$\overrightarrow{Q}_i$	s	s	i	i	i	s	s	i
$\overrightarrow{C}_i$	13	13	13	13	13	3	3	3
$\overrightarrow{D}_i$	0	0	1	2	3	3	3	1
$\overrightarrow{\texttt{MIN\_SIZE}}_i$	0	0	0	0	3	3	3	1

1	0	1	1	1	1		
$\in$	$\notin$	$\in$	$\in$	$\in$	$\in$		
6	5	4	3	2	1	0	i
i	s	i	i	i	i	s	$\overleftarrow{Q}_i$
4	4	13	13	13	13	13	$\overleftarrow{C}_i$
1	4	4	3	2	1	0	$\overleftarrow{D}_i$
1	4	0	0	0	0	0	$\overleftarrow{\texttt{MIN\_SIZE}}_i$

`group` $\left(\overrightarrow{\texttt{MIN\_SIZE}}_7 = 1, \langle 0,1,1,1,0,0,1\rangle\right)$ `group` $\left(\overleftarrow{\texttt{MIN\_SIZE}}_6 = 1, \langle 1,1,1,1,0,1\rangle\right)$

glue matrix entry associated with the state pair (i, i):

`MIN_SIZE` $= \min\left(\overrightarrow{C}_7, \overrightarrow{D}_7 + \overleftarrow{D}_6, \overleftarrow{C}_6\right) = \min(3, 1+1, 4) = 2$

Figure 5.391: Illustrating the use of the state pair (i, i) of the glue matrix for linking `MIN_SIZE` with the counters variables obtained after reading the prefix $0,1,1,1,0,0,1$ and corresponding suffix $1,0,1,1,1,1$ of the sequence $0,1,1,1,0,0,1,1,0,1,1,1,1$; note that the suffix $1,0,1,1,1,1$ (in pink) is proceed in reverse order; the left (resp. right) table shows the initialisation (for $i = 0$) and the evolution (for $i > 0$) of the state of the automaton and its counters C and D upon reading the prefix $0,1,1,1,0,0,1$ (resp. the reverse suffix $1,1,1,1,0,1$).

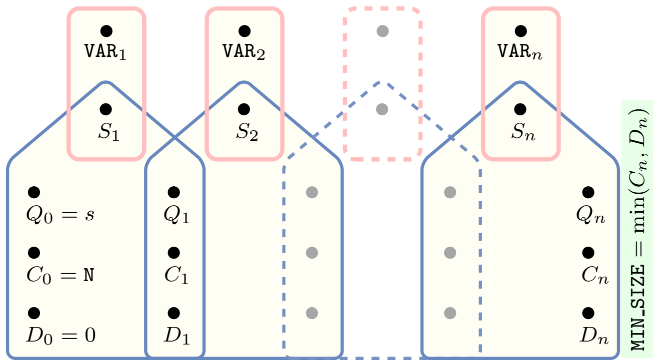


Figure 5.392: Hypergraph of the reformulation corresponding to the automaton (with two counters) of the MIN_SIZE argument of the group constraint where N stands for |VARIABLES| (since all states of the automaton are accepting there is no restriction on the last variable Q_n)

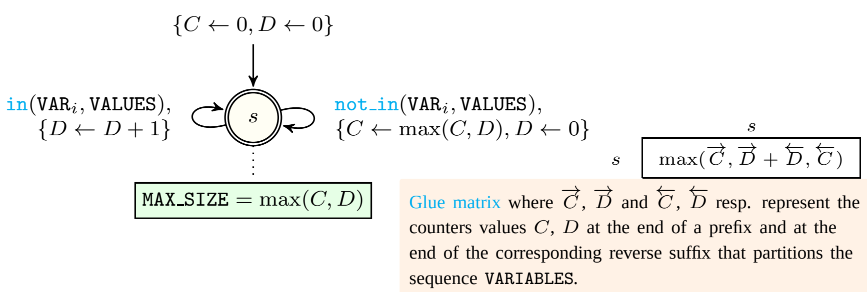


Figure 5.393: Automaton for the MAX_SIZE argument of the group constraint and its glue matrix

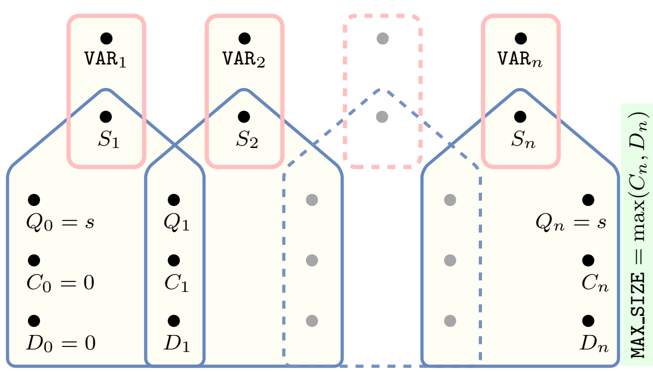
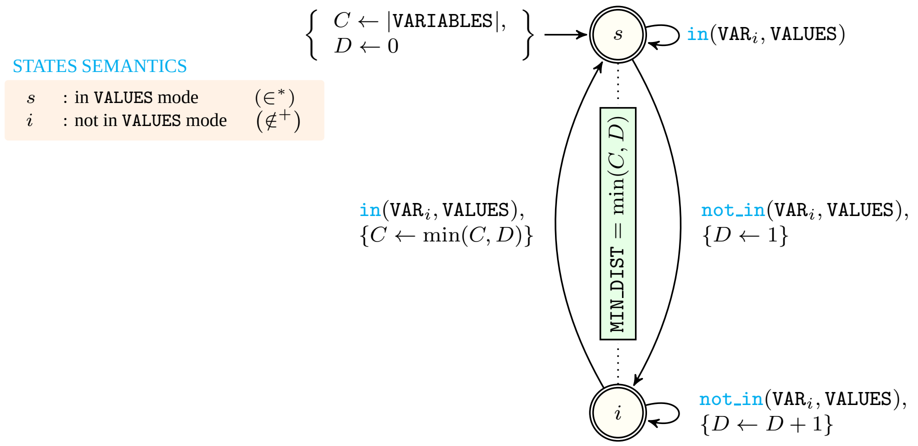


Figure 5.394: Hypergraph of the reformulation corresponding to the automaton (with two counters) of the MAX_SIZE argument of the group constraint



Glue matrix where $\overrightarrow{C}$, $\overrightarrow{D}$ and $\overleftarrow{C}$, $\overleftarrow{D}$ resp. represent the counters values C, D at the end of a prefix and at the end of the corresponding reverse suffix that partitions the sequence VARIABLES.

	$s\ (\in^*)$	$i\ (\notin^+)$
$s\ (\in^*)$	$\min(\overrightarrow{C}, \overrightarrow{D} + \overleftarrow{D}, \overleftarrow{C})$	$\min(\overrightarrow{C}, \overrightarrow{D}, \overleftarrow{C})$
$i\ (\notin^+)$	$\min(\overrightarrow{C}, \overrightarrow{D}, \overleftarrow{C})$	$\min(\overrightarrow{C}, \overrightarrow{D} + \overleftarrow{D}, \overleftarrow{C})$

Figure 5.395: Automaton for the MIN_DIST argument of the group constraint and its glue matrix

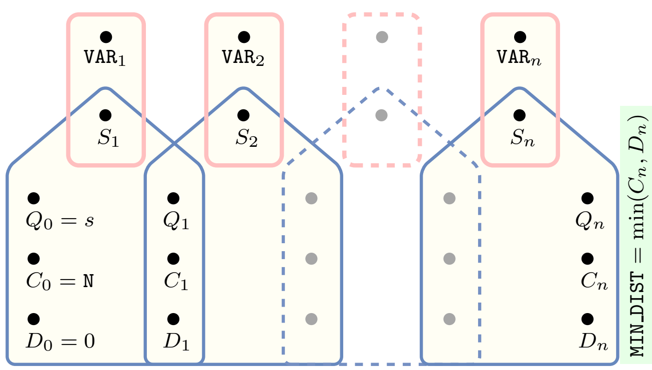


Figure 5.396: Hypergraph of the reformulation corresponding to the automaton (with two counters) of the MIN_DIST argument of the group constraint where N stands for $|\texttt{VARIABLES}|$ (since all states of the automaton are accepting there is no restriction on the last variable Q_n)

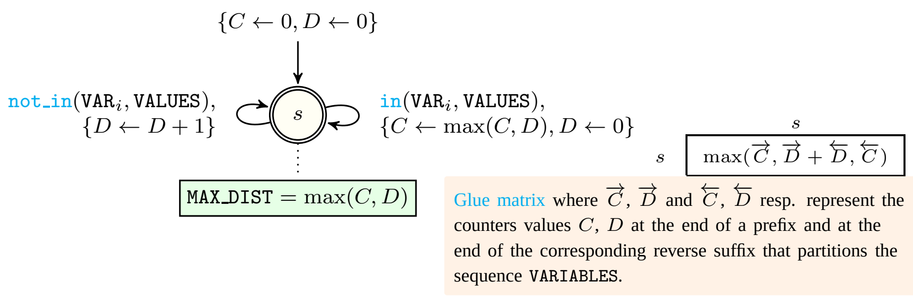


Figure 5.397: Automaton for the MAX_DIST argument of the group constraint and its glue matrix

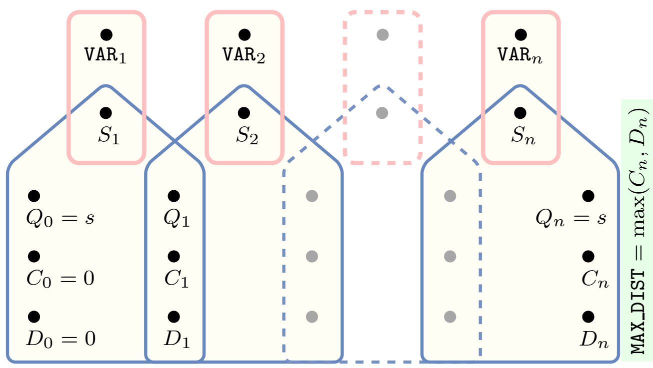


Figure 5.398: Hypergraph of the reformulation corresponding to the automaton (with two counters) of the MAX_DIST argument of the group constraint

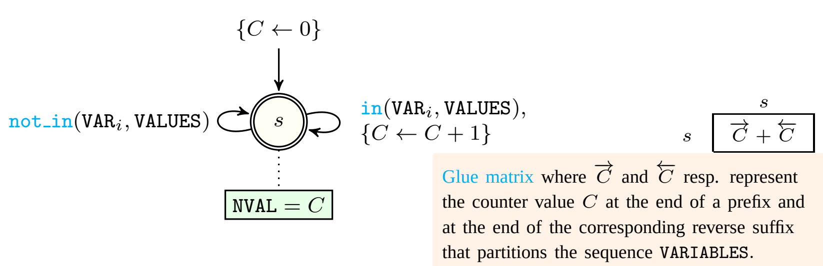


Figure 5.399: Automaton for the NVAL argument of the group constraint and its glue matrix

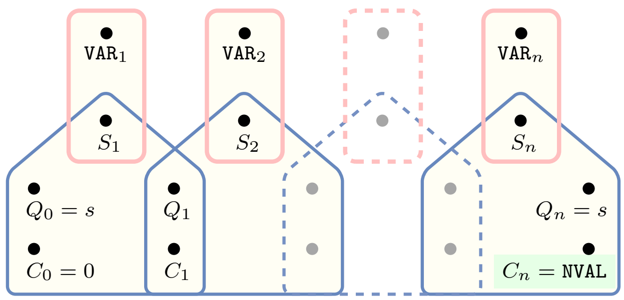


Figure 5.400: Hypergraph of the reformulation corresponding to the automaton (with one counter) of the `NVAL` argument of the `group` constraint