Constraints and Automata (time-series constraints)

Nicolas Beldiceanu

nicolas.beldiceanu@mines-nantes.fr

ACP summer school, Cork, June 2016

Table of content

The story to remember

Generate a few hundred time-series constraints
in a **compositional way**,
but **need**
to **simplify** them,
to **generate necessary conditions** and **bounds**
to get some reasonable behaviour both in CP and LP

Since **too many constraints** this cannot be done
for each constraint individually in a reasonable time-frame

Has also to handle the combinatorial aspect
of these constraints in a compositional way

Finite state transducer

(*introduced by* M.P. Schützenberger)

- A FST is defined by:
  - A finite set of **states** Q
  - An **input alphabet** Σ (*finite set of input symbols*)
  - An **output alphabet** Γ (finite set of output symbols)
  - A **transition relation** δ: $Q \times (\Sigma \cup \{\varepsilon\}) \times (\Sigma \cup \{\varepsilon\}) \times Q$
  - An **initial state** $q_{\text{init}} \in Q$
  - A set of **accepting states** $F \subseteq Q$

Some time can have more than one output symbol
in the transition relation

Finite state transducer

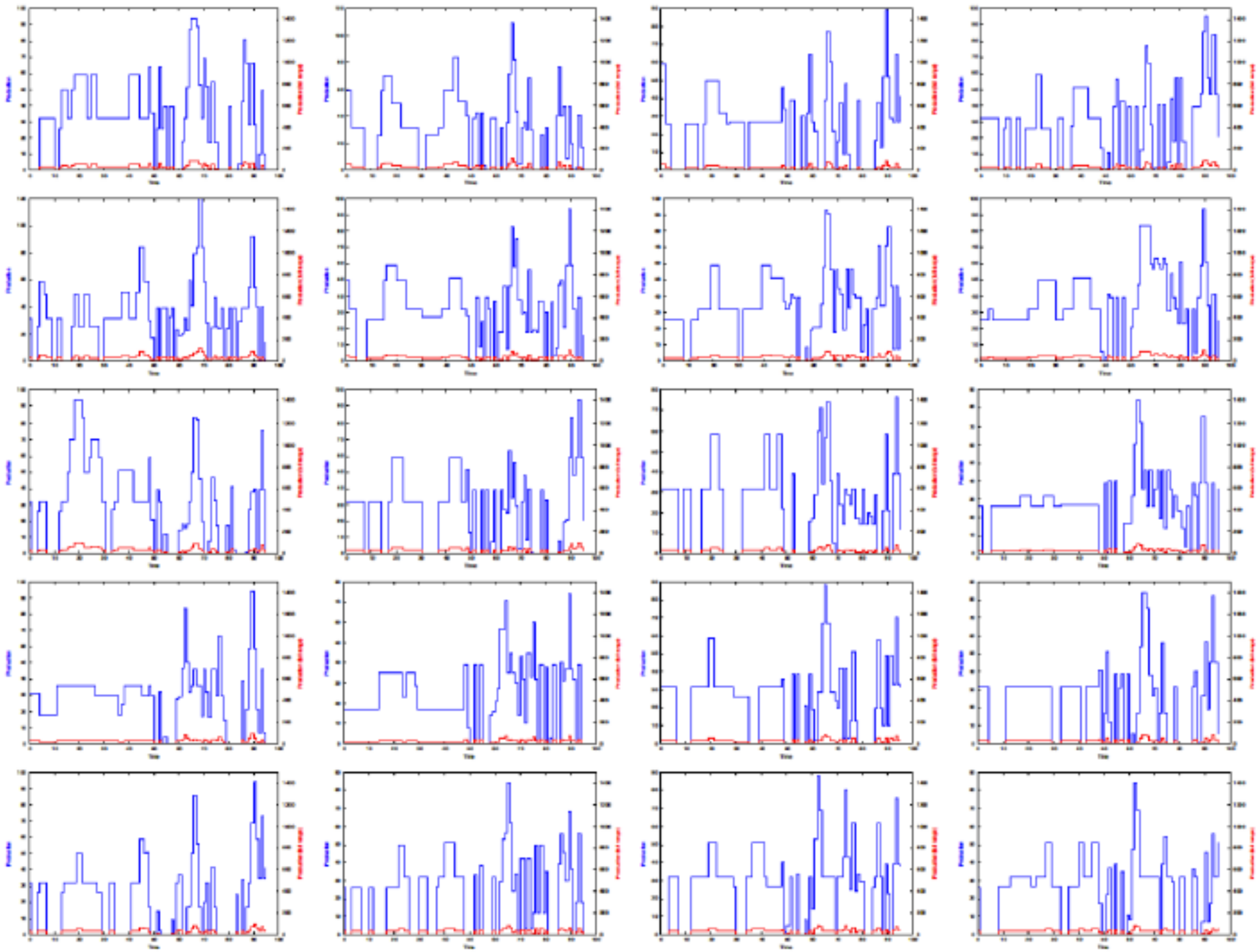
- Used mostly
  - Natural language (*text of speech*)
- But also
  - Computational biology
  - Fraud detection
- Popular (*like automata*) in industry
  - microsoft research
  - google (M. Mohri)
- Like automata you can learn them

examples of transducers later on

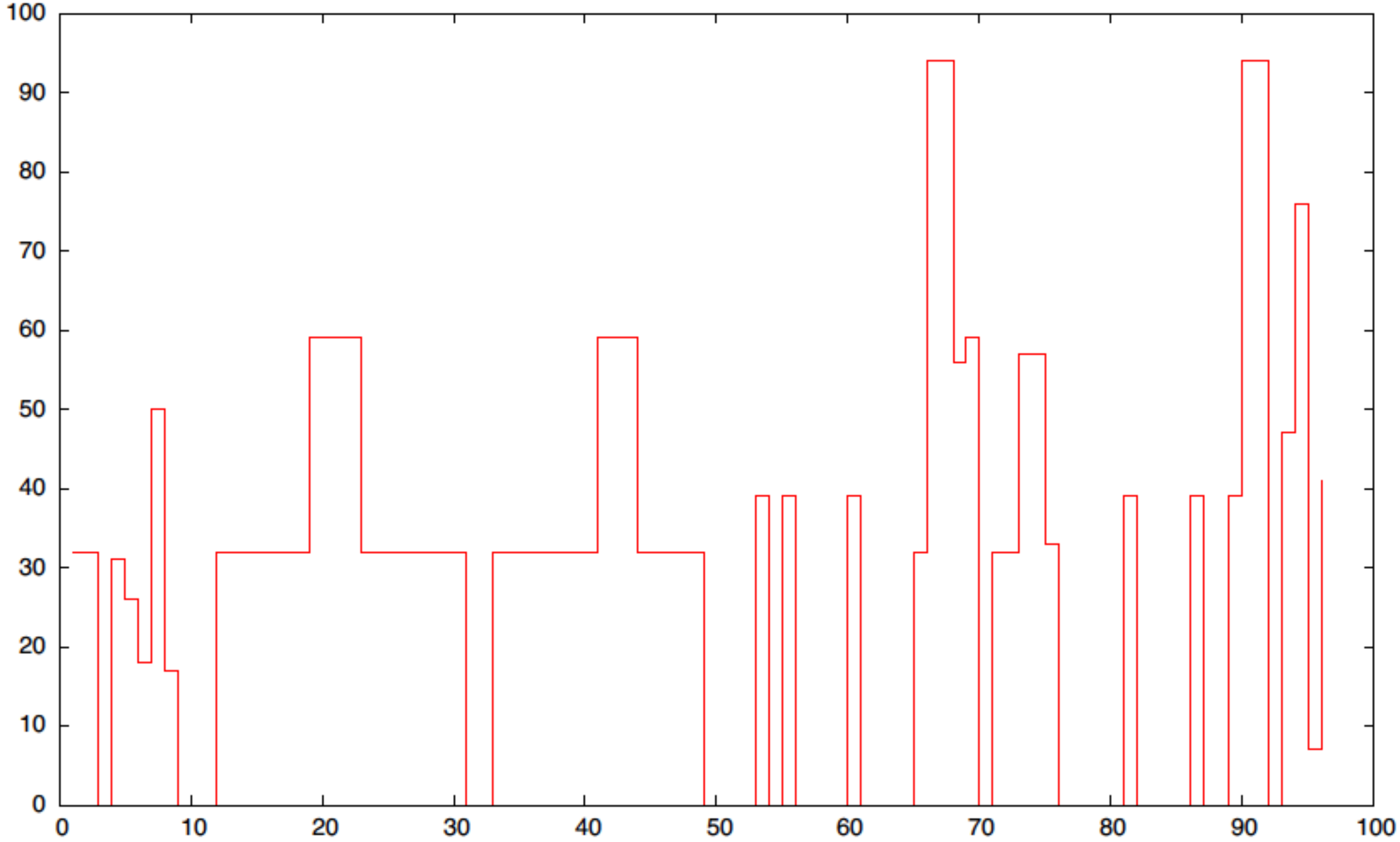
Background

- PGMO Project with EDF
- Analysis of power output curves for electricity generators
- Use ModelSeeker to describe/categorize/synthesize output from UCP model
- Published in CP 2013

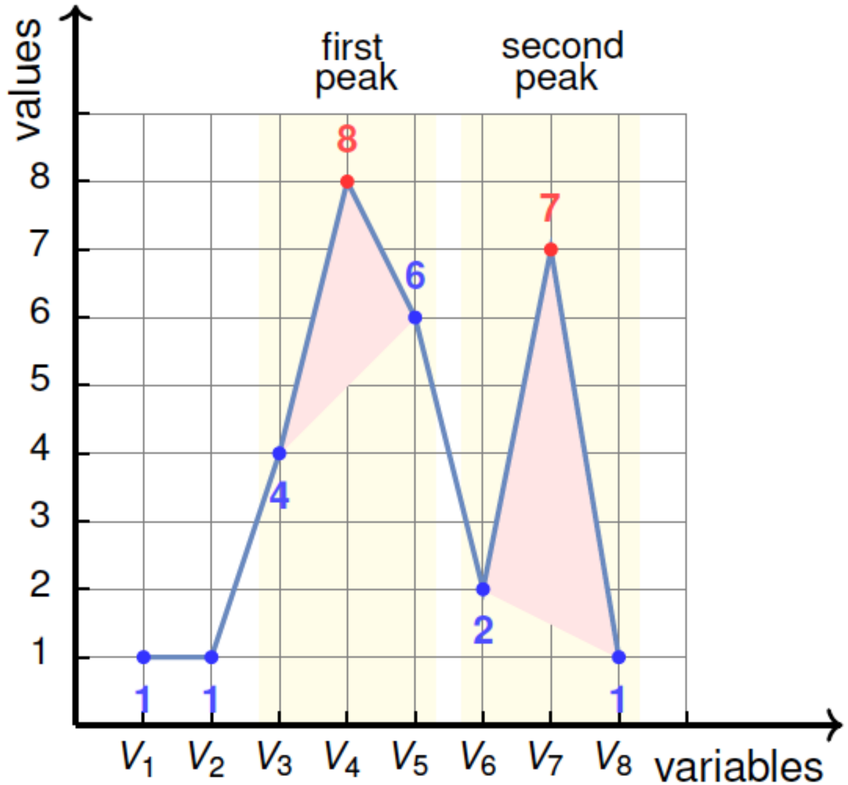
Example: From this …



… to this (generated profile)



Example: the peak constraint



peak $(2, \langle 1, 1, 4, 8, 6, 2, 7, 1 \rangle)$

Automaton with counters: peak constraint in Global Constraint Catalog

STATE SEMANTICS

s	: stationary/decreasing mode	$(\{> \mid =\}^*)$
u	: increasing mode	$(< \{< \mid =\}^*)$

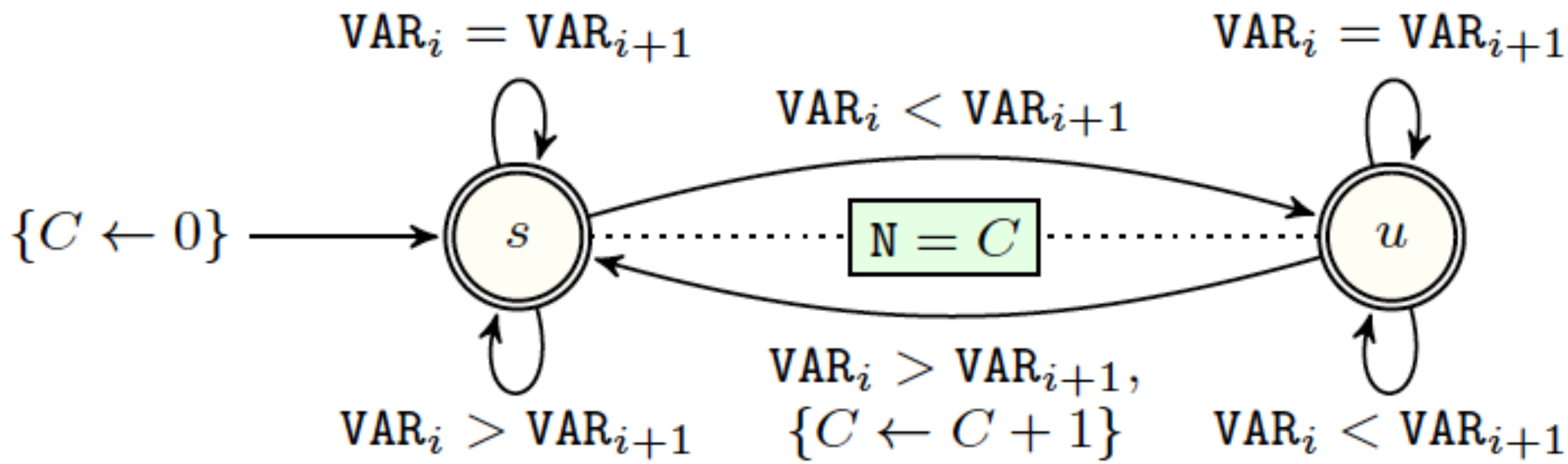


Figure 5.689: Automaton of the PEAK constraint

But always some missing constraints
(*when meeting people from industry*)

and dont want to introduce the missing
constraints one by one in the
global constraint catalog

which leads to a synthesized
time-series catalog

Table of content

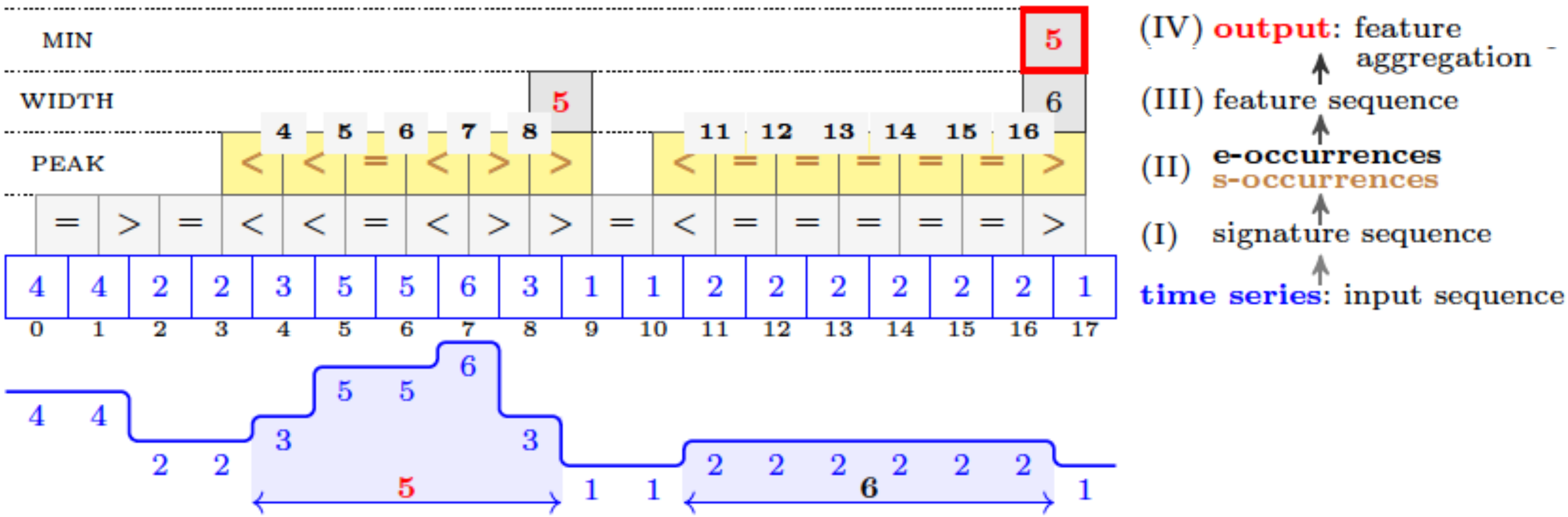
Decomposing the definition of a constraint

- Constraints are **pure functional dependencies** or **predicates** (*see predicates later on*)

- Implemented as **automata with counters**

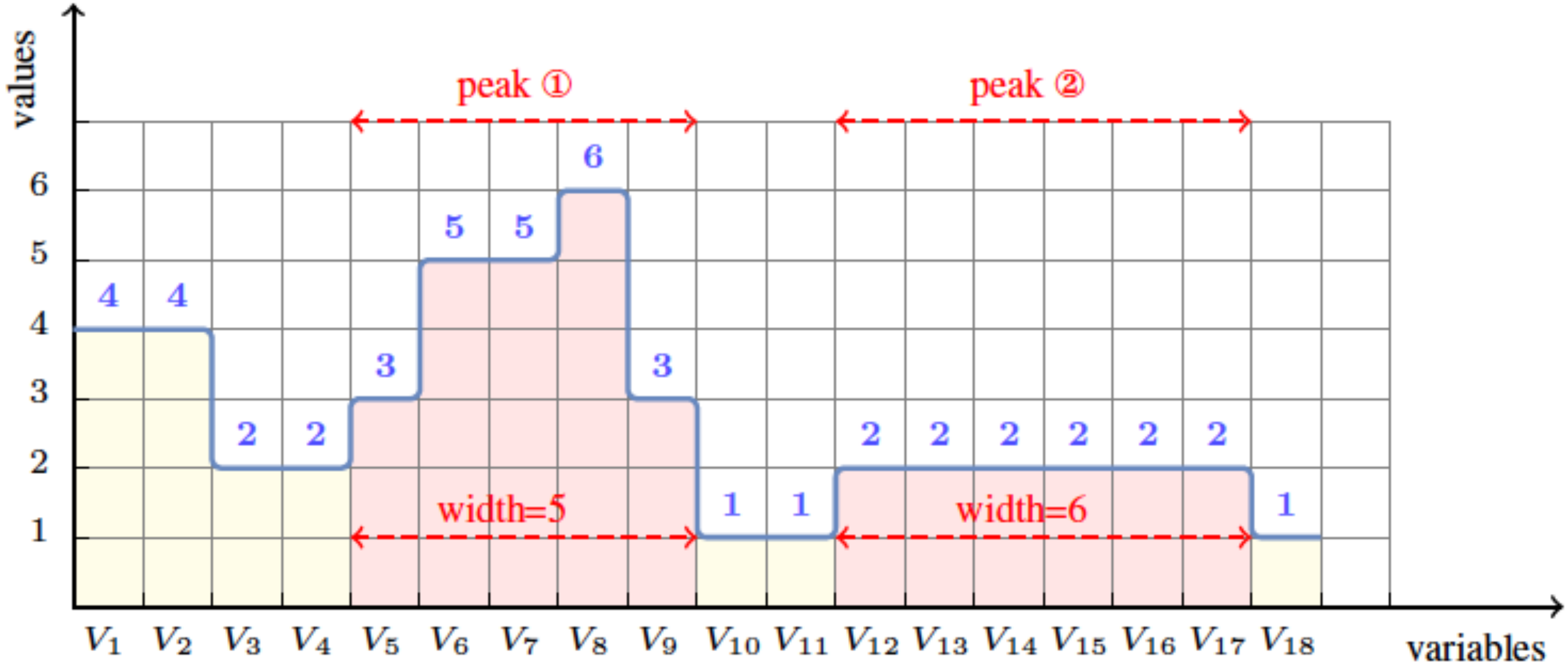
- **Four steps** (layers) in definition
  - Building **signature**
  - Recognize **pattern occurrences** in sequence
  - Extract **feature** per pattern
  - **Aggregate** features

Example: min_width_peak



$$\text{MIN\_WIDTH\_PEAK}(5, \langle 4, 4, 2, 2, 3, 5, 5, 6, 3, 1, 1, 2, 2, 2, 2, 2, 2, 1\rangle)$$

Example: min_width_peak (continued)



$$\text{MIN\_WIDTH\_PEAK}(5, \langle 4, 4, 2, 2, 3, 5, 5, 6, 3, 1, 1, 2, 2, 2, 2, 2, 2, 1 \rangle)$$

Signature

- Convert (integer) value to finite alphabet
- Signature links two consecutive entries in time-series
- We use <,=,> with their natural semantics
- Other signatures possible

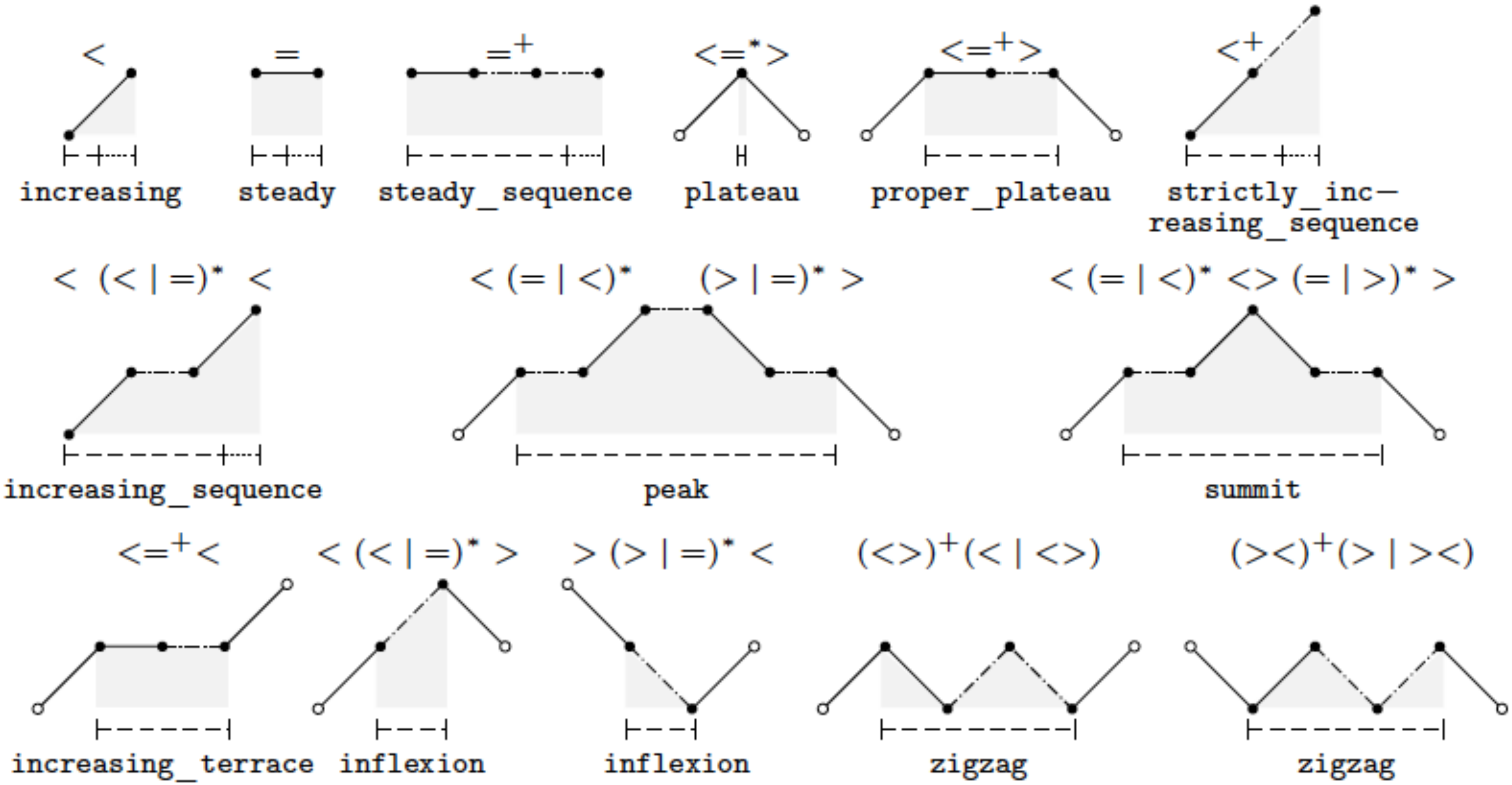
Patterns

- **Identify a pattern** we are looking for r
- **Extract subpart** for which computes feature a, b

pattern	regular expression r	before b	after a
increasing	$<$	0	0
increasing_sequence	$<(< \mid =)^{*} < \mid <$	0	0
increasing_terrace	$<=^{+}<$	1	1
summit	$(< \mid (<(= \mid <)^{*} <))(> \mid (>(= \mid >)^{*} >))$	1	1
plateau	$<=^{*}>$	1	1
proper_plateau	$<=^{+}>$	1	1
strictly_increasing_sequence	$<^{+}$	0	0
peak	$<(= \mid <)^{*}(> \mid =)^{*}>$	1	1
inflexion	$<(< \mid =)^{*} > \mid >(> \mid =)^{*} <$	1	1
steady	$=$	0	0
steady_sequence	$=^{+}$	0	0
zigzag	$(<>)^{+}(< \mid <>) \mid (><)^{+}(> \mid ><)$	1	1

*Find **maximal** words matching regular expression r*

Patterns (continued)



Definition of $\{s|i|e\}$-occurrences of an occurrence of pattern

Given

an input sequence $x_0, x_1, \ldots, x_{n-1}$,
its signature sequence $s_0, s_1, \ldots, s_{n-2}$,
a pattern (r, a, b),
a non-empty signature subsequence $s_i, s_{i+1}, \ldots, s_j$
forming a maximum word matching r

the s-occurrence $(i..j)$ is the index sequence $i, \ldots, j$,
the i-occurrence $[(i+b)..j]$ is the index sequence $i+b, \ldots, j$,
the e-occurrence $[[(i+b)..(j+1-a)]]$ is the index sequence
$i+b, \ldots, j+1-a$.

Definition of {s|i|e}-occurrences of an occurrence of pattern

Given

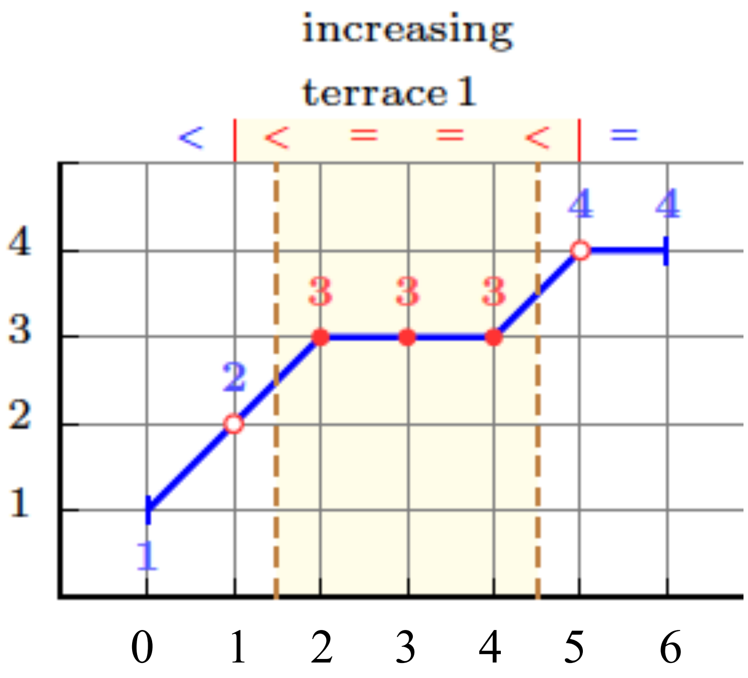
an input sequence $x_0,x_1,\ldots,x_{n-1}$,
its signature sequence $s_0,s_1,\ldots,s_{n-2}$,
a pattern (r,a,b),
a non-empty signature subsequence $s_i,s_{i+1},\ldots,s_j$
forming a maximum word matching r

the s-occurrence $(i..j)$ is the index sequence $i,\ldots,j$,
the i-occurrence $[(i+b)..j]$ is the index sequence $i+b,\ldots,j$,
the e-occurrence $[[(i+b)..(j+1-a)]]$ is the index sequence $i+b,\ldots,j+1-a$.

s-occurrences : *maximal signature sequence matching r*
i-occurrences : *do not overlap* (*footprint of the pattern*)
e-occurrences : *used to compute the feature value*

Example of {*s*|*i*|*e*}-occurrences for the **increasing_terrace** pattern

pattern	regular expression r	before b	after a
increasing terrace	$<=^{+}<$	1	1



Indices of

s-occurrences : **1..4** (<==<)

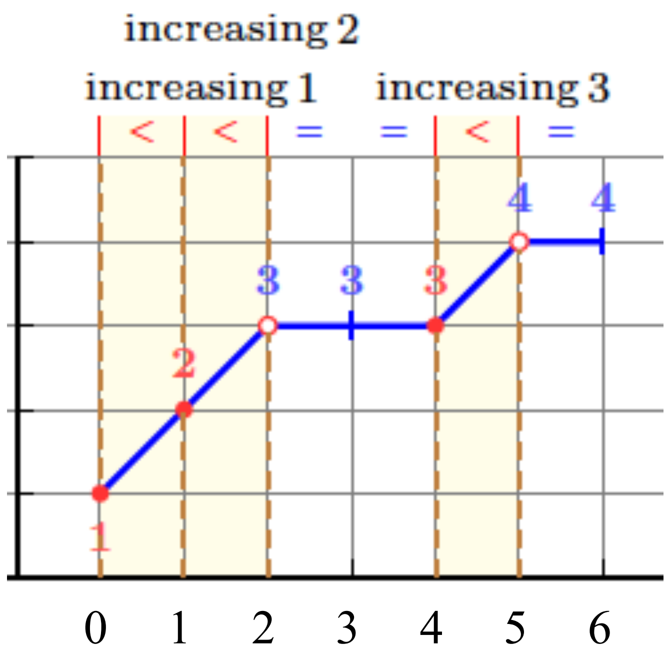
i-occurrences : **[2..4]** (333)

e-occurrences : **[[2..4]]** (333)

a=1=*b* excludes first and last input values 2 and 4

Example of {*s*|*i*|*e*}-occurrences for the **increasing** pattern

pattern	regular expression r	before b	after a
increasing	$<$	0	0



Indices of

s-occurrences : **0..0** **1..1** **4..4**

i-occurrences : **[0..0]** **[1..1]** **[4..4]**

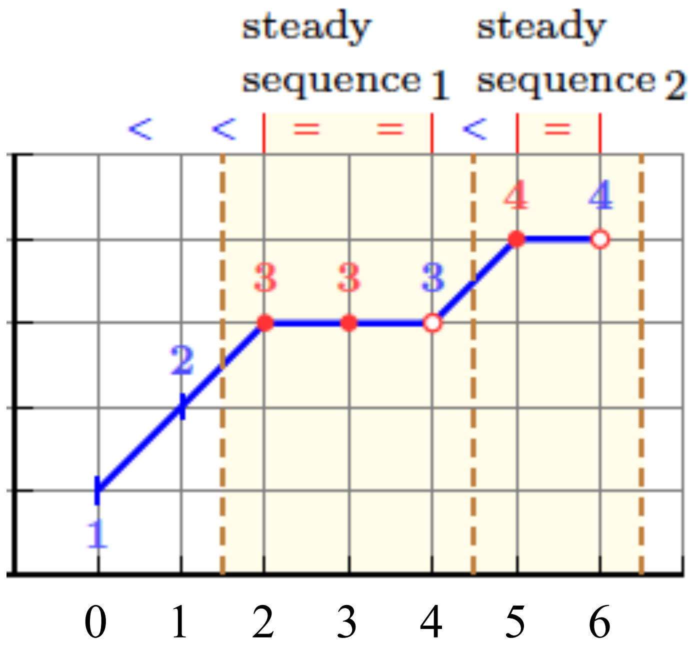
e-occurrences : **[[0..1]]** **[[1..2]]** **[[4..5]]**

since b=0 *s*-occurences and *i*-occurrences match

since a=0=b the 1 and 2 input values are part of *e*-occurrences

Example of {*s*|*i*|*e*}-occurrences for the **steady_sequence** pattern

pattern	**regular expression** r	**before** b	**after** a
steady_sequence	$=^{+}$	0	0



Indices of

s-occurrences : **2..3** **5..5**

i-occurrences : **[2..3]** **[5..5]**

e-occurrences : **[[2..4]]** **[[5..6]]**

since *b*=0 *s*-occurences and *i*-occurrences match

since *a*=0=*b* the 1 and 2 input values are part of *e*-occurrences

Features (*computed from e-occurrences*)

- `one` : value 1
- `width` : number of positions of the e-occurrence
- `surf` : sum of the values of the e-occurrence
- `max` : maximum value of the e-occurrence
- `min` : minimum value of the e-occurrence
- `range` : range of the e-occurrence: `max-min`

Aggregators (*computed from sequence of features*)

- `max` : largest value of a sequence of features
- `min` : smallest value of a sequence of features
- `sum` : sum of the features of a sequence of feature

Device for recognizing *i*-occurrences of a pattern: a seed transducer

- Define a pattern by a **transducer**
  (*reading/writing regular language*)

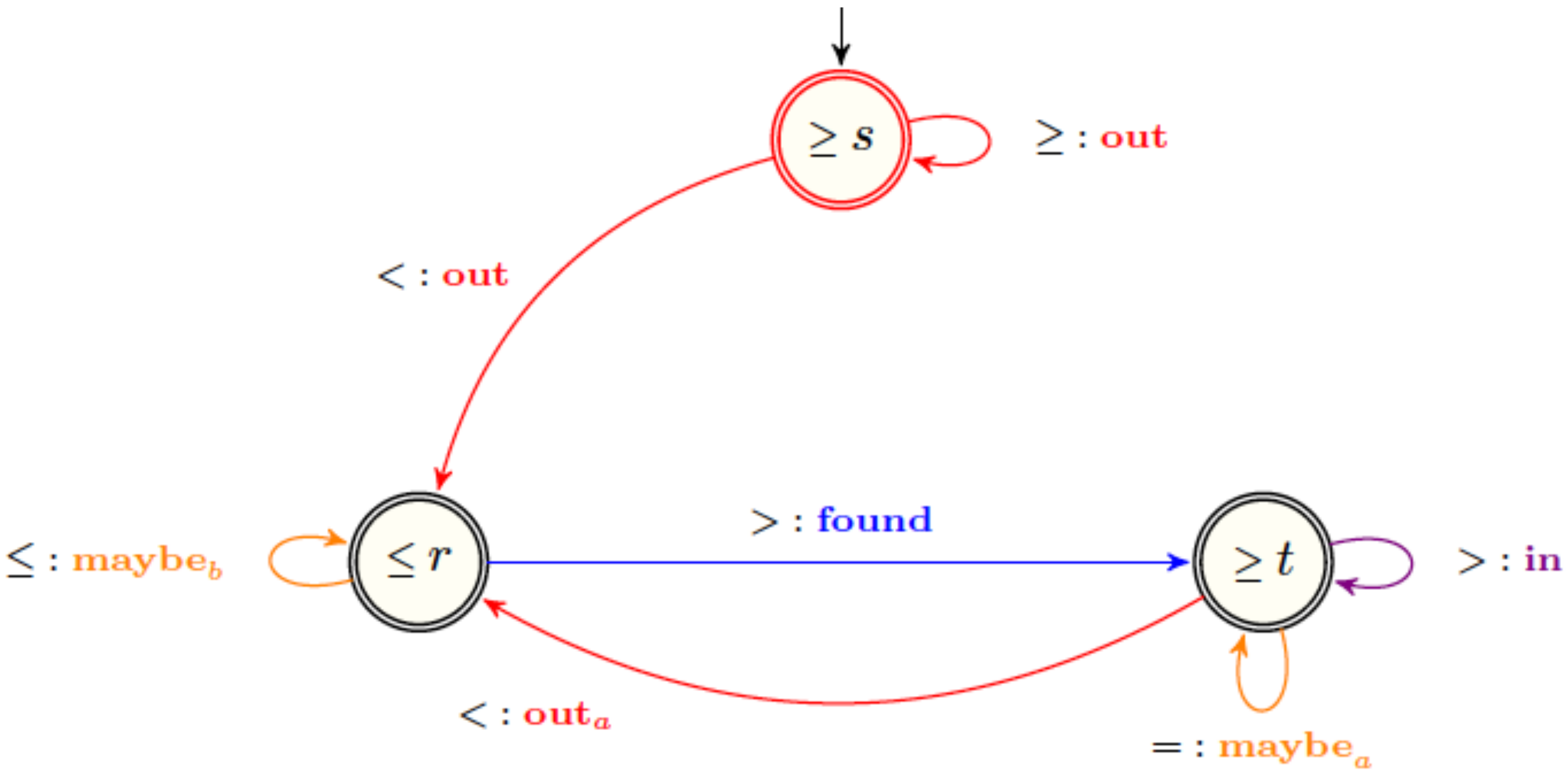
- **Input**: signature sequence

- **Output**: word of a semantic alphabet with letters:
  - **out** we are **outside** the pattern
  - **maybe** we are **possibly in** the pattern
    (*must be confirmed later on*)
  - **found** **first place** we know **we are in** the pattern
  - **in** we are **still** in the pattern

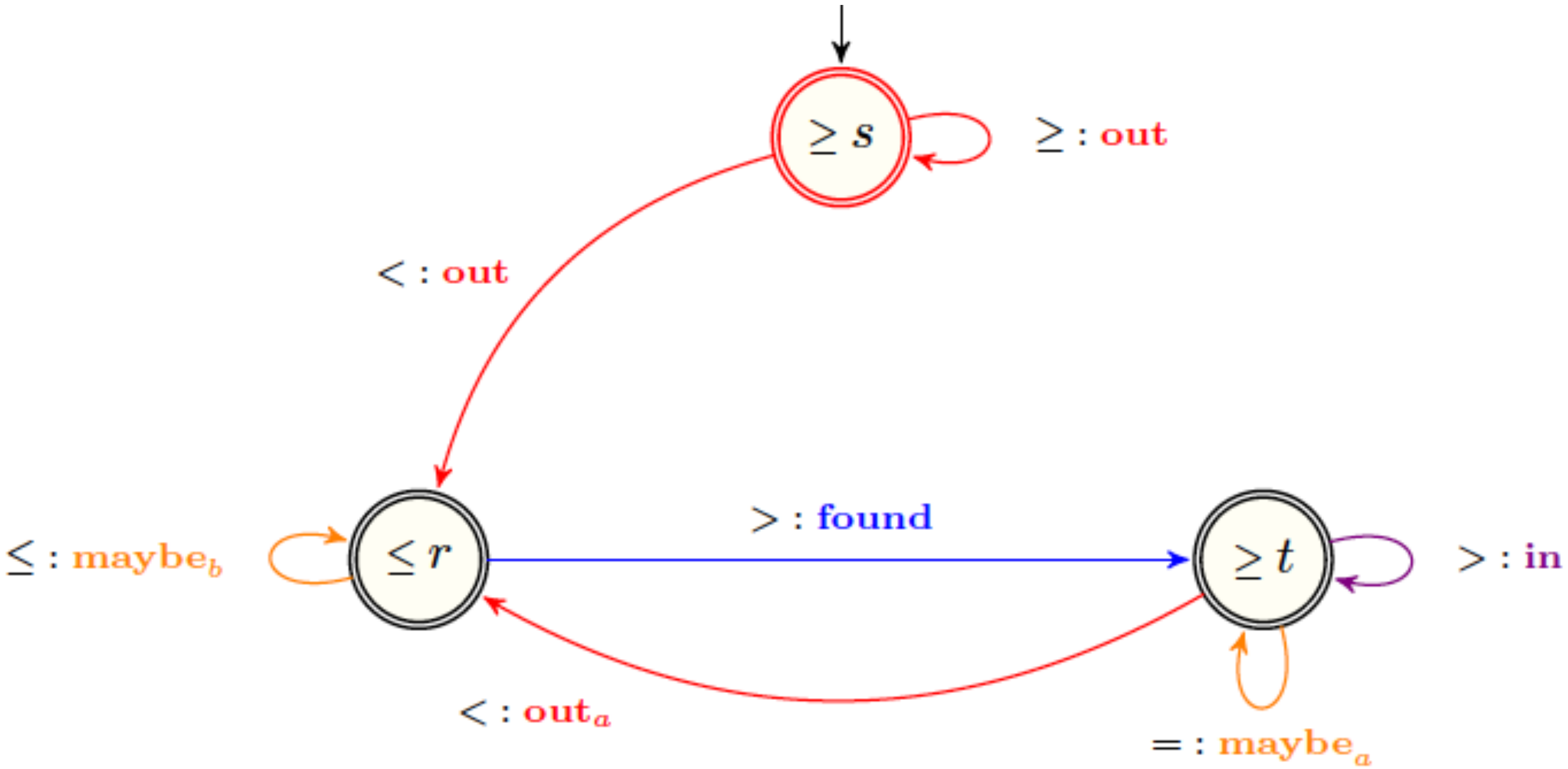
***everything** will be synthesized from the seed transducer*

Example: transducer for the **peak** pattern

pattern	**regular expression** r	**before** b	**after** a
peak	$<\ (=\mid<)^*\ (>\mid=)^*\ >$	1	1



Example: transducer for the **peak** pattern



o	o	o	o	m_b	m_b	m_b	f	in	m_a	o_a	m_b	m_b	m_b	m_b	m_b	f	output: semantic string
=	>	=	<	<	=	<	>	>	=	<	=	=	=	=	=	>	input: signature string

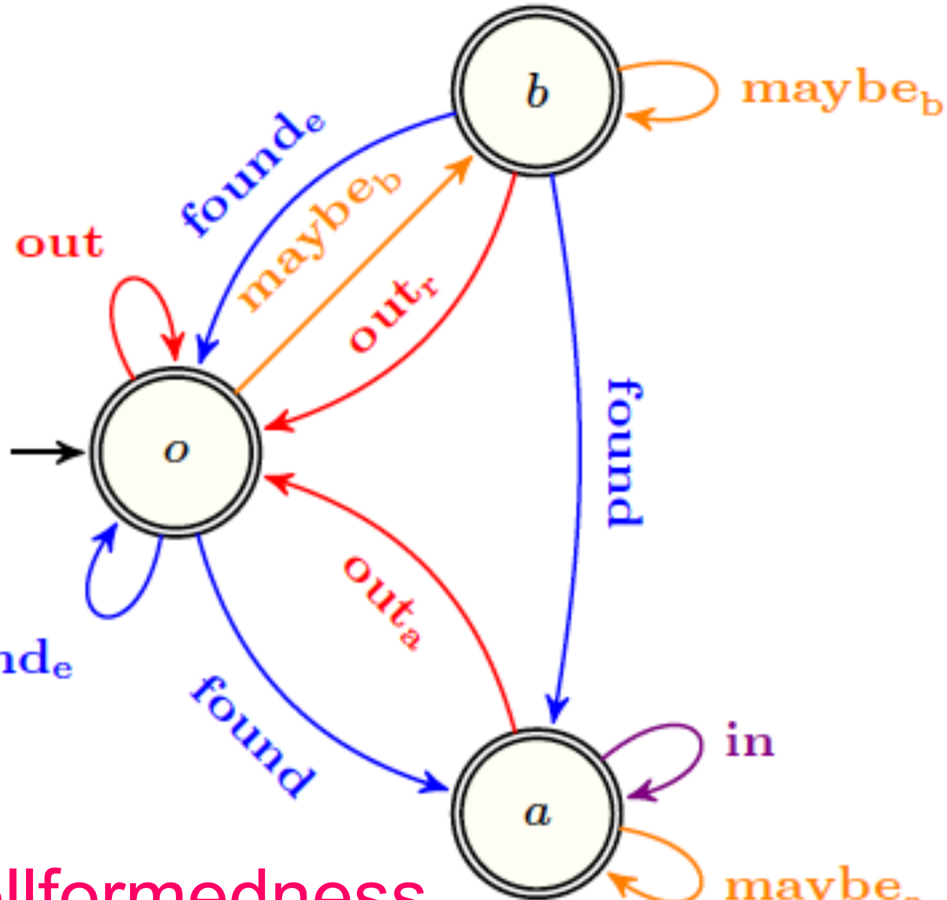
states:	*s*	*s*	*s*	*s*	*r*	*r*	*r*	*r*	*t*	*t*	*t*	*r*	*r*	*r*	*r*	*r*	*r*	*t*
	4	4	2	2	3	5	5	6	3	1	1	2	2	2	2	2	2	1

input: integer sequence

Well-formed seed transducer
(*language of the output*)

state semantics

o : outside or after the end of a pattern
b : potentially inside (before a found/found$_e$)
a : potentially inside (after a found)

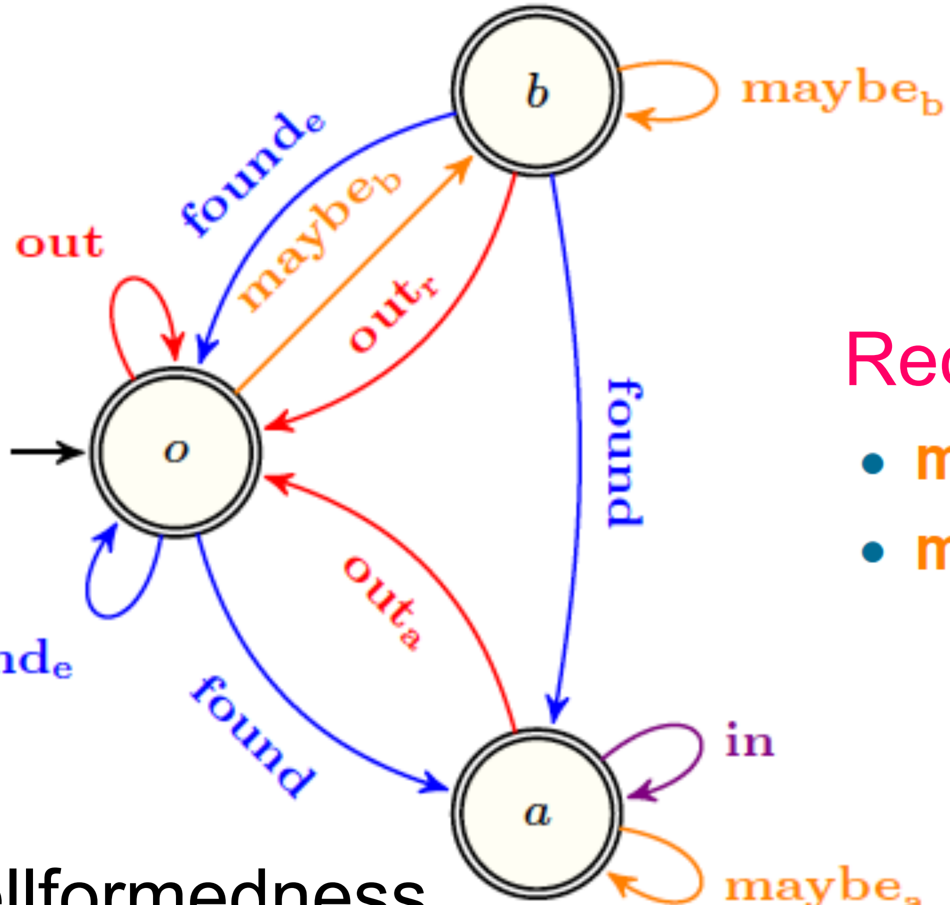


Wellformedness

Well-formed seed transducer
(*language of the output*)

state semantics

o : outside or after the end of a pattern
b : potentially inside (before a found/found$_e$)
a : potentially inside (after a found)



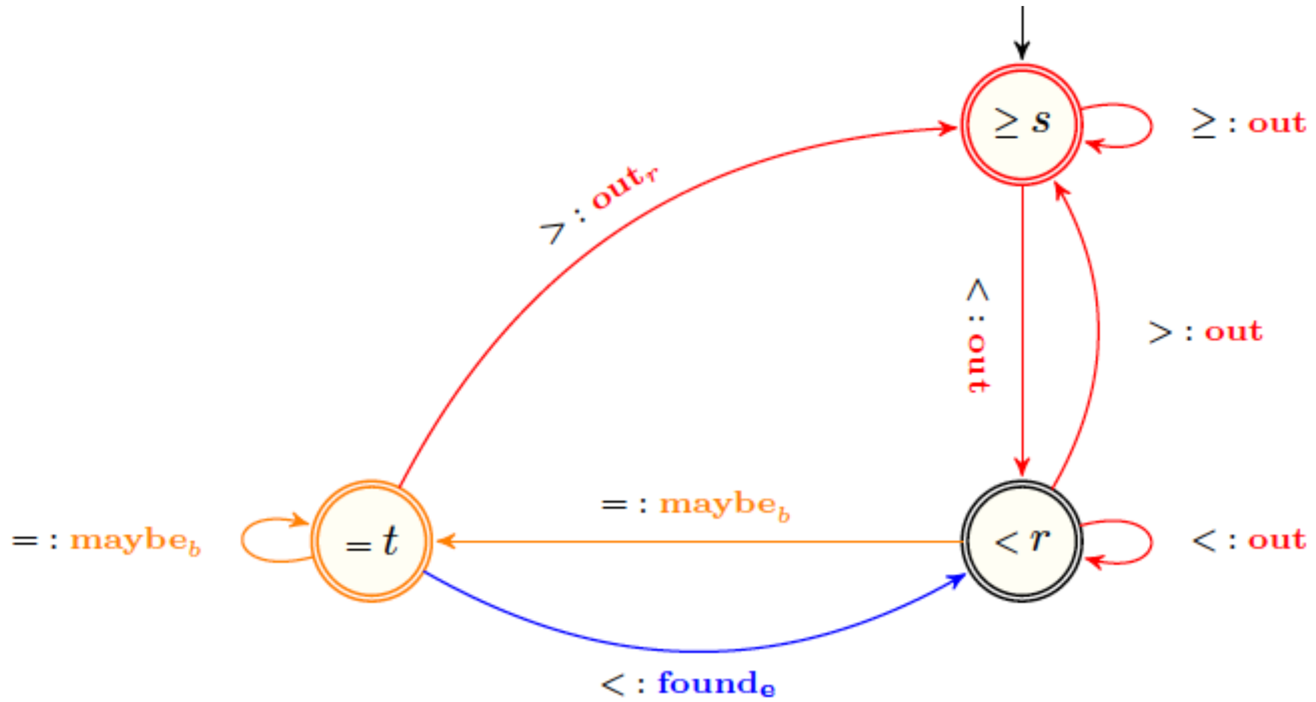
Wellformedness

Recognizing pattern

- maybe$_b$* found$_e$
- maybe$_b$* found {maybe$_a$* in$^+$}*

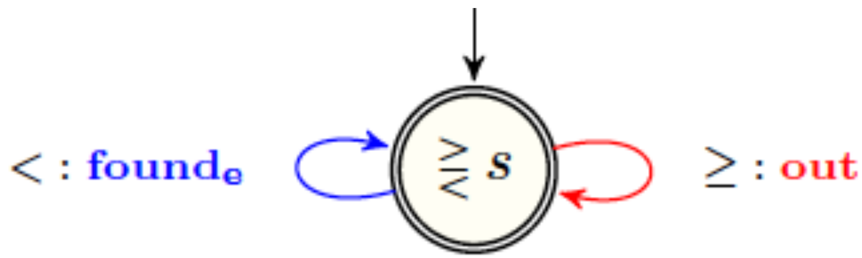
Transducer for **increasing_terrace**

pattern	regular expression r	before b	after a
`increasing_terrace`	$<=^+<$	1	1



Transducer for **increasing**

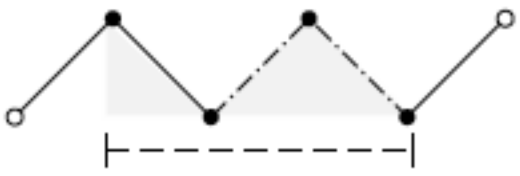
pattern	regular expression r	before b	after a
increasing	$<$	0	0



Transducer for **zigzag**

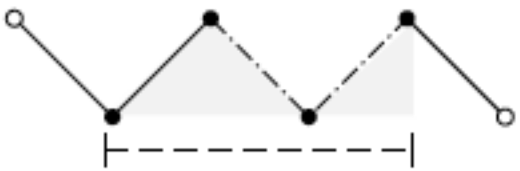
a=1=*b*

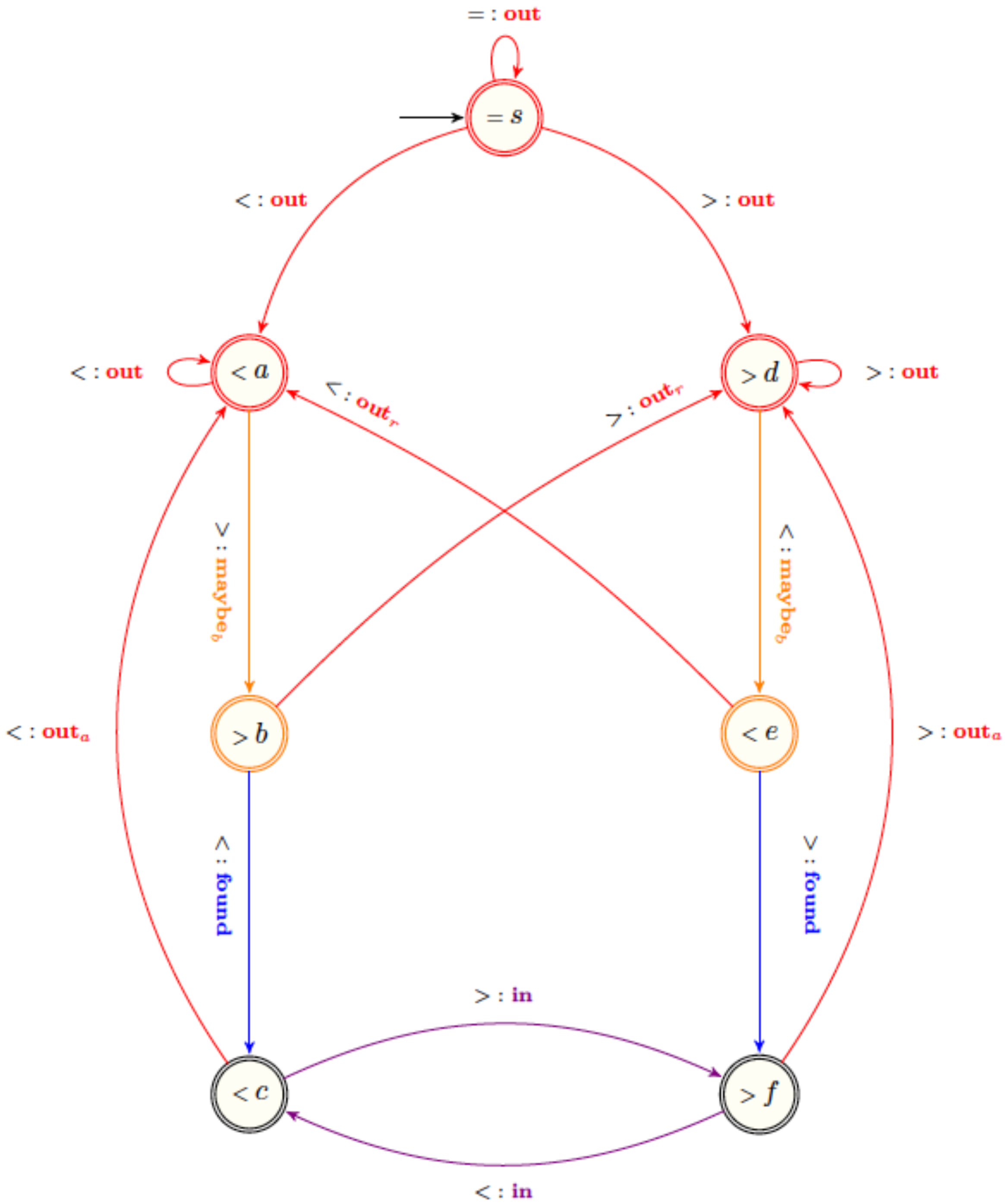
$(<>)^+(< \mid <>)$



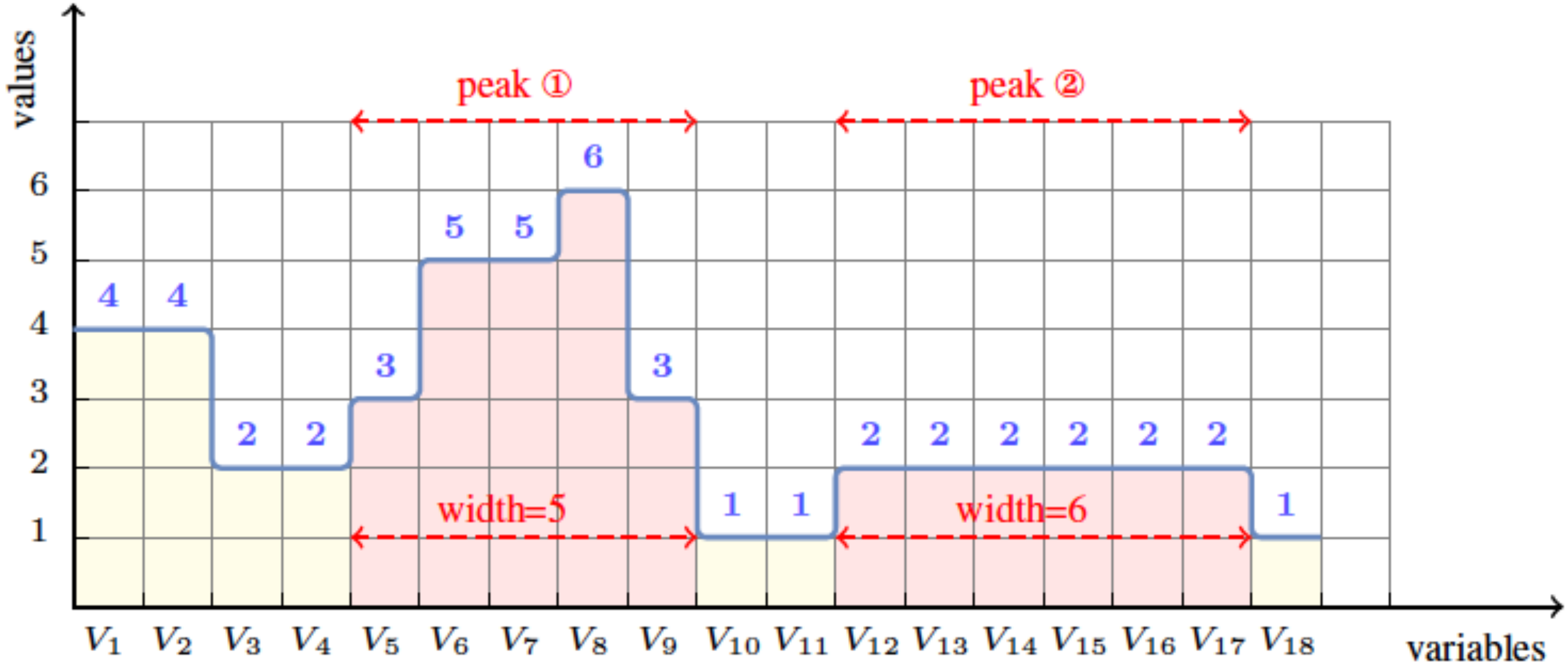
or

$(><)^+(> \mid ><)$





Footprint constraint: identifying *i*-occurrences of a pattern

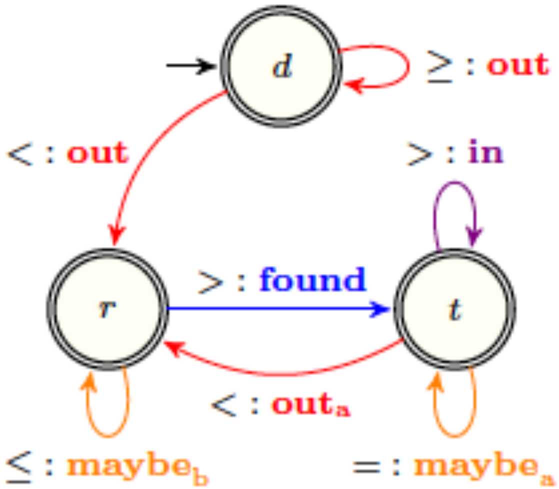


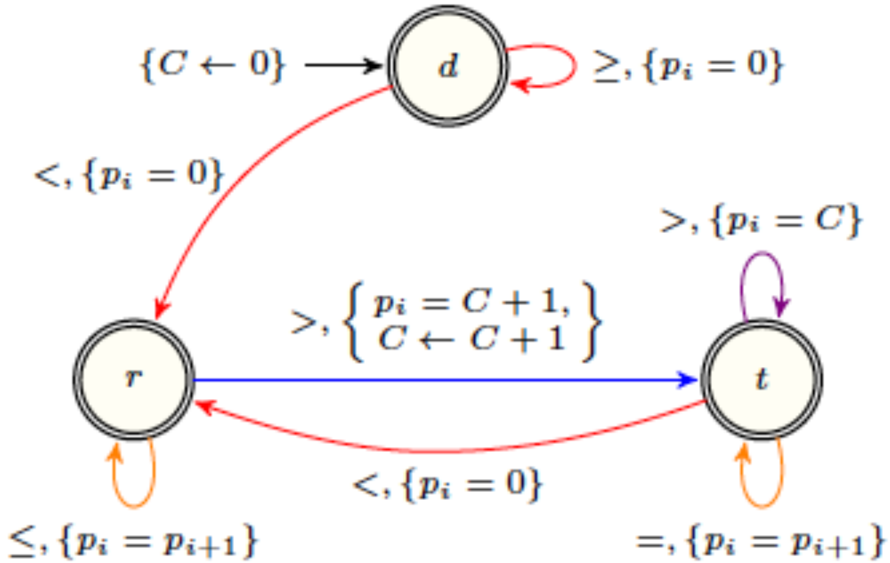
footprint(peak, [4,4,2,2,3,5,5,6,3,1,1,2,2,2,2,2,2,1],
[0,0,0,0,1,1,1,1,1,0,0,2,2,2,2,2,2,0])

Decoration table for the footprint constraint (*generating counters updates*)

initialisation	$C \leftarrow 0$	
return	$p_n = 0$	
semantic letters	**annotations**	
	guard	*update of C*
out	$p_i = 0$	
$\mathbf{out_r}$	$p_i = 0$	
$\mathbf{out_a}$	$p_i = 0$	
$\mathbf{maybe_b}$	$p_i = p_{i+1}$	
$\mathbf{maybe_a}$	$p_i = p_{i+1}$	
$\mathbf{found_e}$	$p_i = C + 1$	$C \leftarrow C + 1$
found	$p_i = C + 1$	$C \leftarrow C + 1$
in	$p_i = C$	

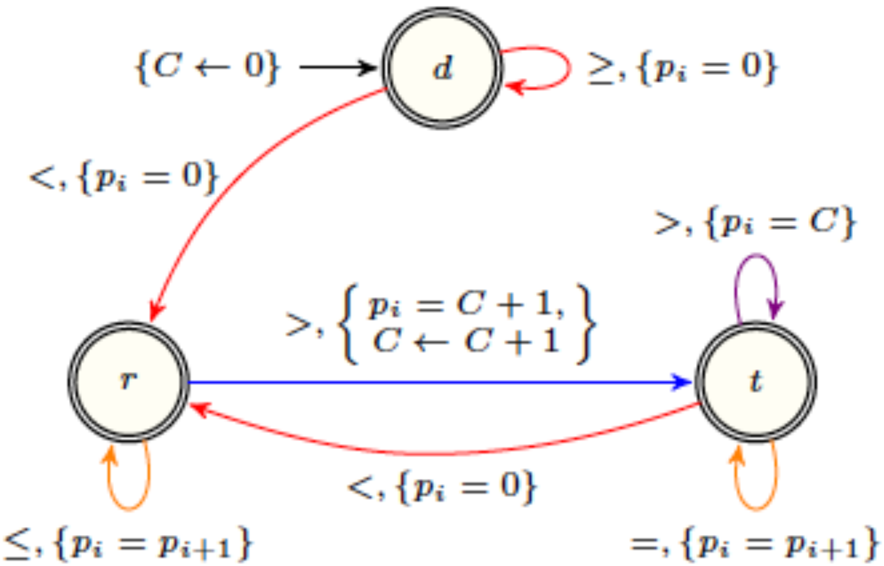
Example: synthesizing the footprint constraint for the **peak** pattern





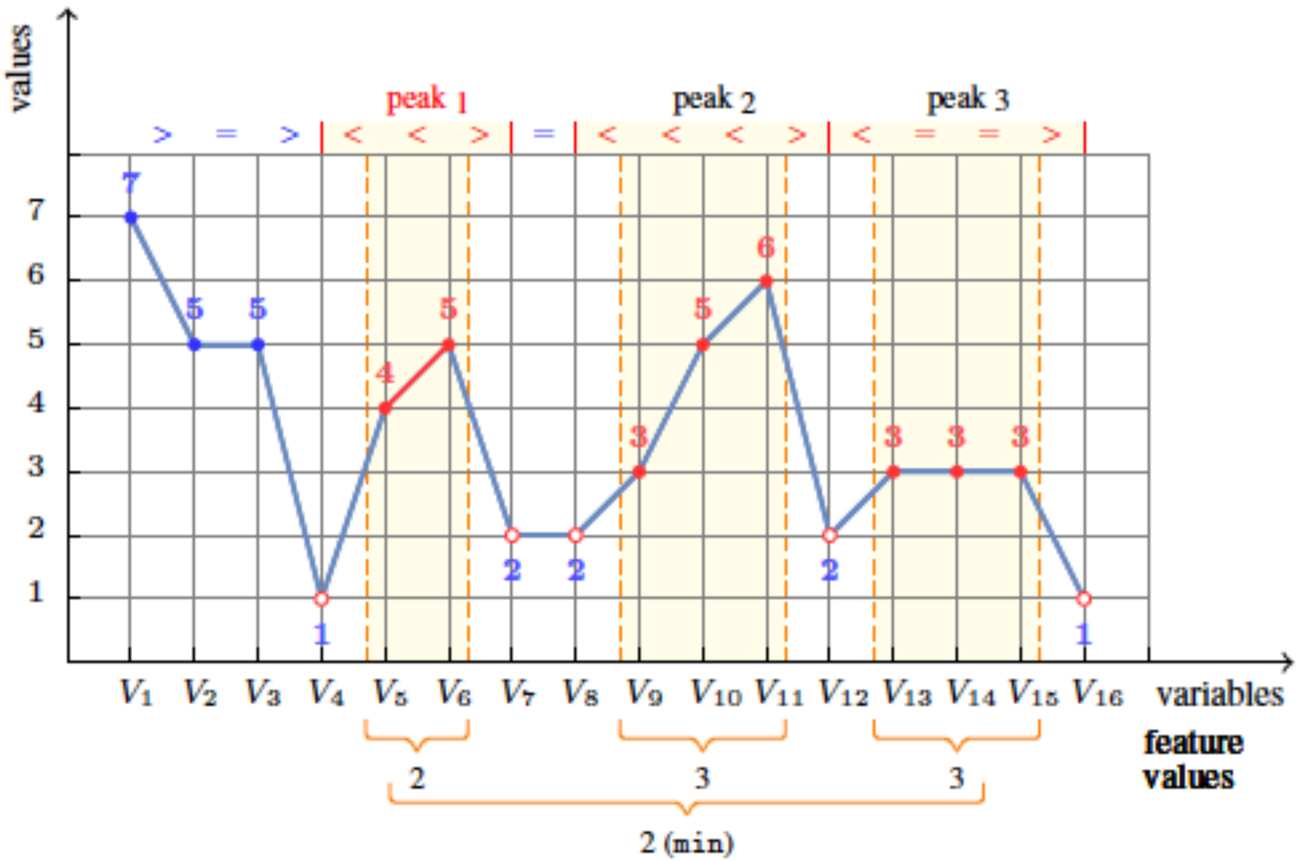
out	$p_i = 0$
$\textbf{out}_{\textbf{r}}$	$p_i = 0$
$\textbf{out}_{\textbf{a}}$	$p_i = 0$
$\textbf{maybe}_{\textbf{b}}$	$p_i = p_{i+1}$
$\textbf{maybe}_{\textbf{a}}$	$p_i = p_{i+1}$
$\textbf{found}_{\textbf{e}}$	$p_i = C + 1 \; C \leftarrow C + 1$
found	$p_i = C + 1 \; C \leftarrow C + 1$
in	$p_i = C$

Example: executing the synthesized footprint automaton of the **peak** pattern



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	$p_0 = 0$	$p_1 = 0$	$p_2 = 0$	$p_3 = 0$	$p_4 = p_5$	$p_5 = p_6$	$p_6 = p_7$	$p_7 = 1$	$p_8 = 1$	$p_9 = p_{10}$	$p_{10} = 0$	$p_{11} = p_{12}$	$p_{12} = p_{13}$	$p_{13} = p_{14}$	$p_{14} = p_{15}$	$p_{15} = p_{16}$	$p_{16} = 2$	$p_{17} = 0$
C_i	${}_0 0$	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	2	
τ_i	${}_0 o$	o	o	o	m_b	m_b	m_b	f	in	m_a	o_a	m_b	m_b	m_b	m_b	m_b	f	
q_i	${}_0 d$	d	d	d	r	r	r	r	t	t	t	r	r	r	r	r	r	t
s_i	${}_0 =$	$>$	$=$	$<$	$<$	$=$	$<$	$>$	$>$	$=$	$<$	$=$	$=$	$=$	$=$	$=$	$>$	
x_i	4	4	2	2	3	5	5	6	3	1	1	2	2	2	2	2	2	1

Feature constraints (example)



MIN_WIDTH_PEAK $(2, [7, 5, 5, 1, 4, 5, 2, 2, 3, 5, 6, 2, 3, 3, 3, 1])$

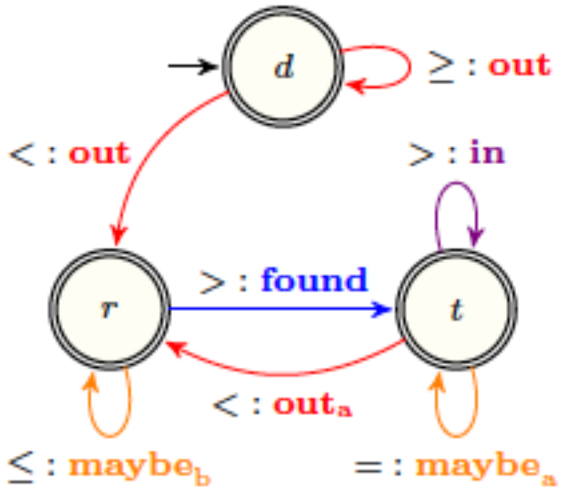
Decoration table for the feature constraint

Feature f	$\mathrm{neutral}_f$	$\min_f$	$\max_f$	ϕ_f	δ_f
one	1	1	1	max	0
width	0	0	n	+	1
surface	0	$-\infty$	$+\infty$	+	x_i
max	$-\infty$	$-\infty$	$+\infty$	max	x_i
min	$+\infty$	$-\infty$	$+\infty$	min	x_i
range	0	0	$+\infty$		x_i

Aggregator g	$\mathrm{default}_{g,f}$
Max	$\min_f$
Min	$\max_f$
Sum	0

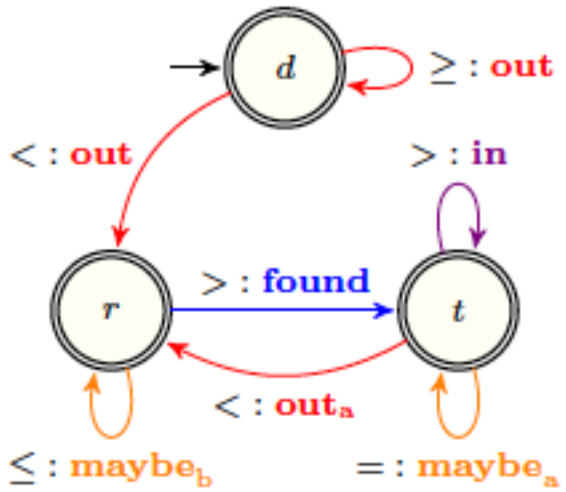
initialization		$C \leftarrow \mathrm{default}_{af}$	$D \leftarrow \mathrm{neutral}_f$	$R \leftarrow \mathrm{default}_{af}$
return		$a(R,C)$		
Semantic Letter		Decoration		
	After	Update of C	Update of D	Update of R
out				
$\mathbf{out}_r$			$D \leftarrow \mathrm{neutral}_f$	
$\mathbf{out}_a$		$C \leftarrow \mathrm{default}_{af}$	$D \leftarrow \mathrm{neutral}_f$	$R \leftarrow a(R,C)$
$\mathbf{maybe}_b$			$D \leftarrow \phi_f(D,\delta_f)$	
$\mathbf{maybe}_a$	0		$D \leftarrow \phi_f(D,\delta'_f)$	
$\mathbf{maybe}_a$	1		$D \leftarrow \phi_f(D,\delta_f)$	
found	0	$C \leftarrow \phi_f(\phi_f(D,\delta_f),\delta'_f)$	$D \leftarrow \mathrm{neutral}_f$	
found	1	$C \leftarrow \phi_f(D,\delta_f)$	$D \leftarrow \mathrm{neutral}_f$	
in	0	$C \leftarrow \phi_f(C,\phi_f(D,\delta'_f))$	$D \leftarrow \mathrm{neutral}_f$	
in	1	$C \leftarrow \phi_f(C,\phi_f(D,\delta_f))$	$D \leftarrow \mathrm{neutral}_f$	
$\mathbf{found}_e$	0		$D \leftarrow \mathrm{neutral}_f$	$R \leftarrow a(R,\phi_f(\phi_f(D,\delta_f),\delta'_f))$
$\mathbf{found}_e$	1		$D \leftarrow \mathrm{neutral}_f$	$R \leftarrow a(R,\phi_f(D,\delta_f))$

Example: synthesizing the automaton for min_width_peak

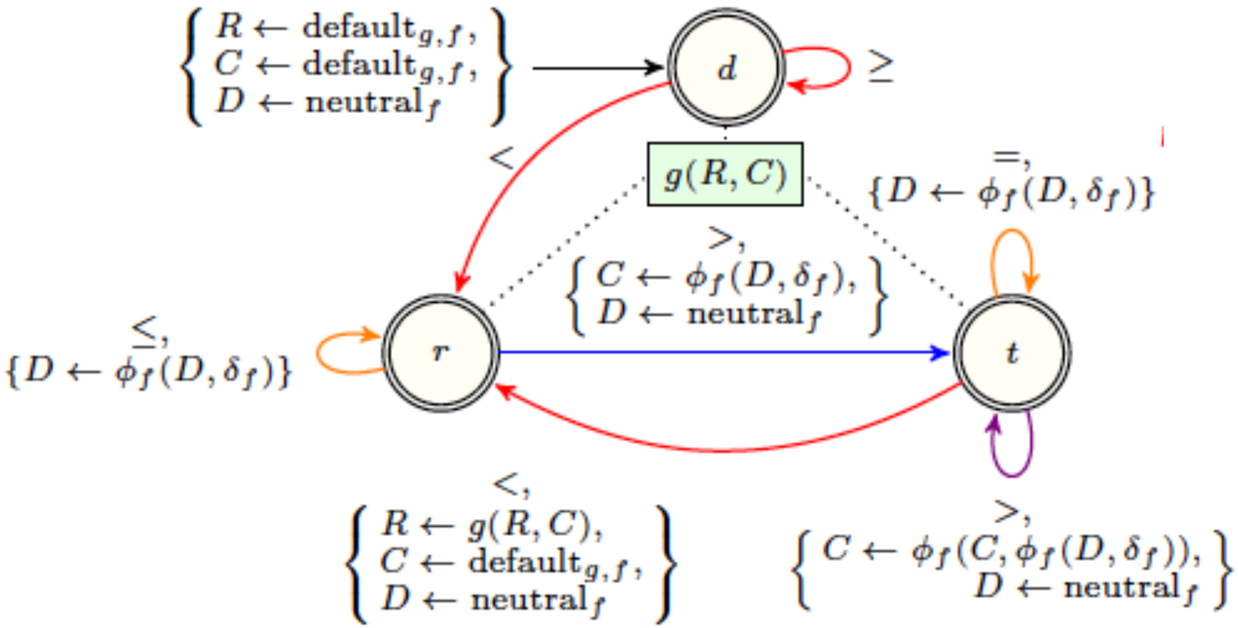


seed transducer

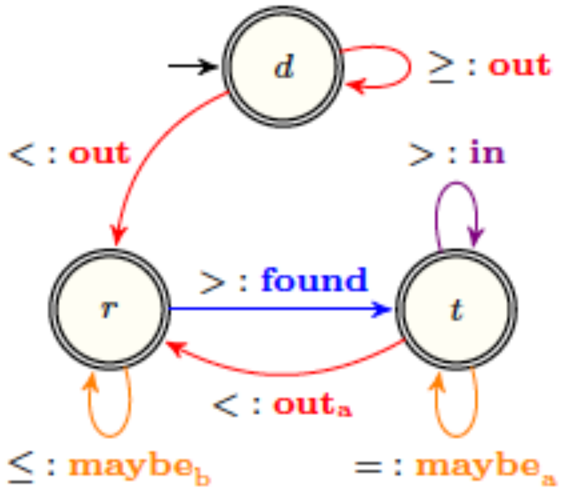
Example: synthesizing the automaton for min_width_peak



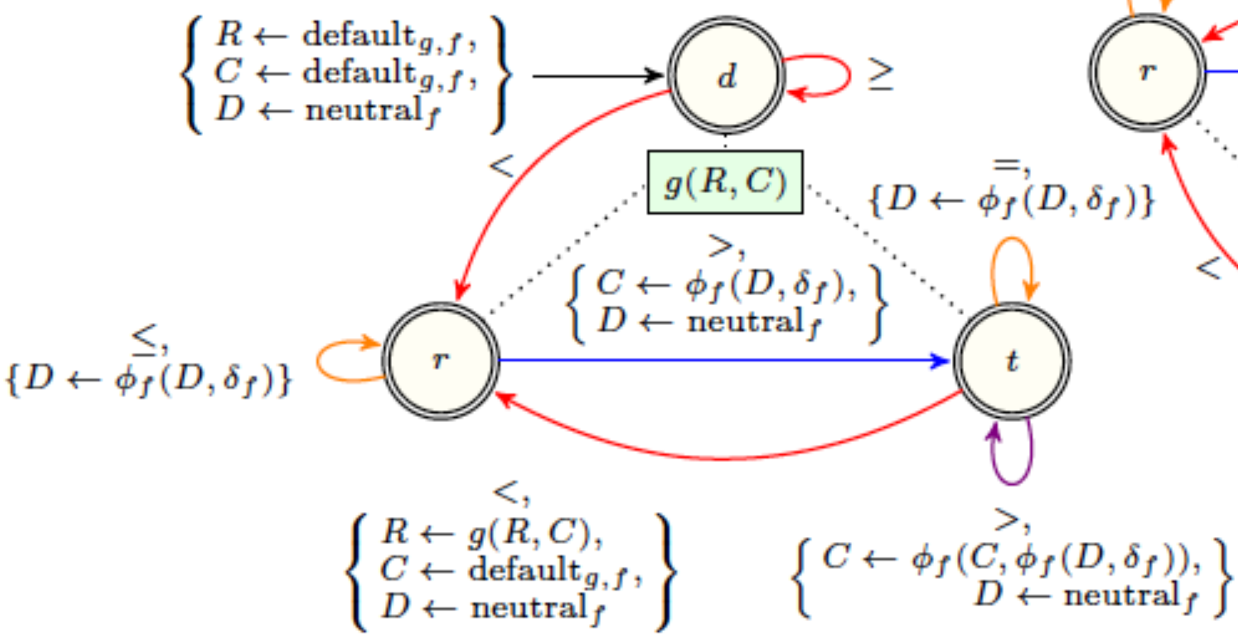
parametrized constraint
(feature f, aggregator g)

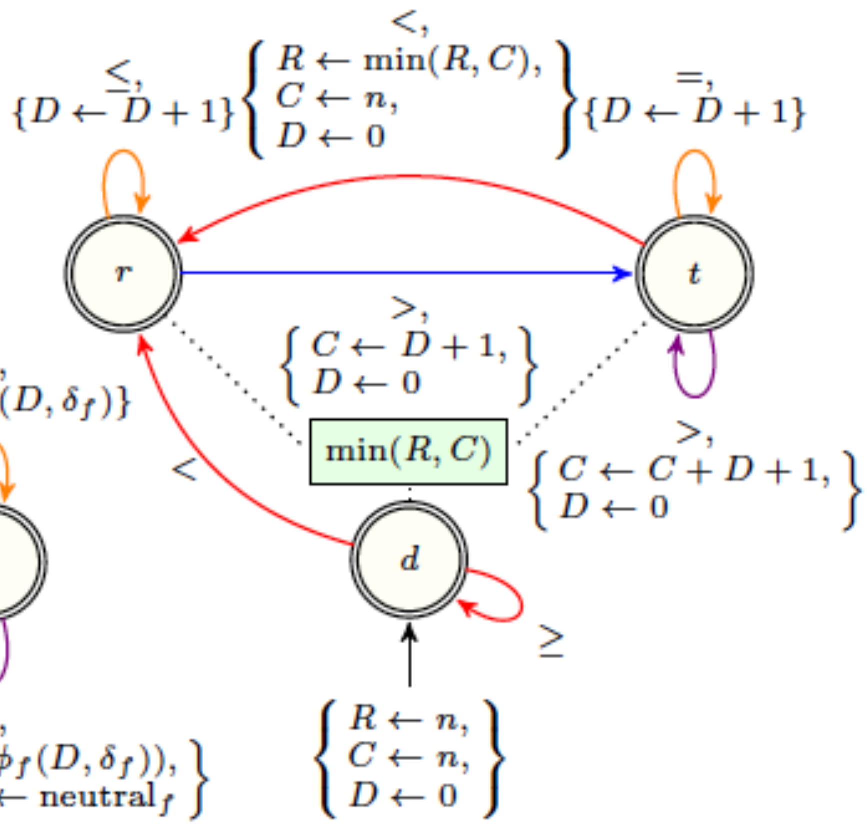


Example: synthesizing the automaton for min_width_peak

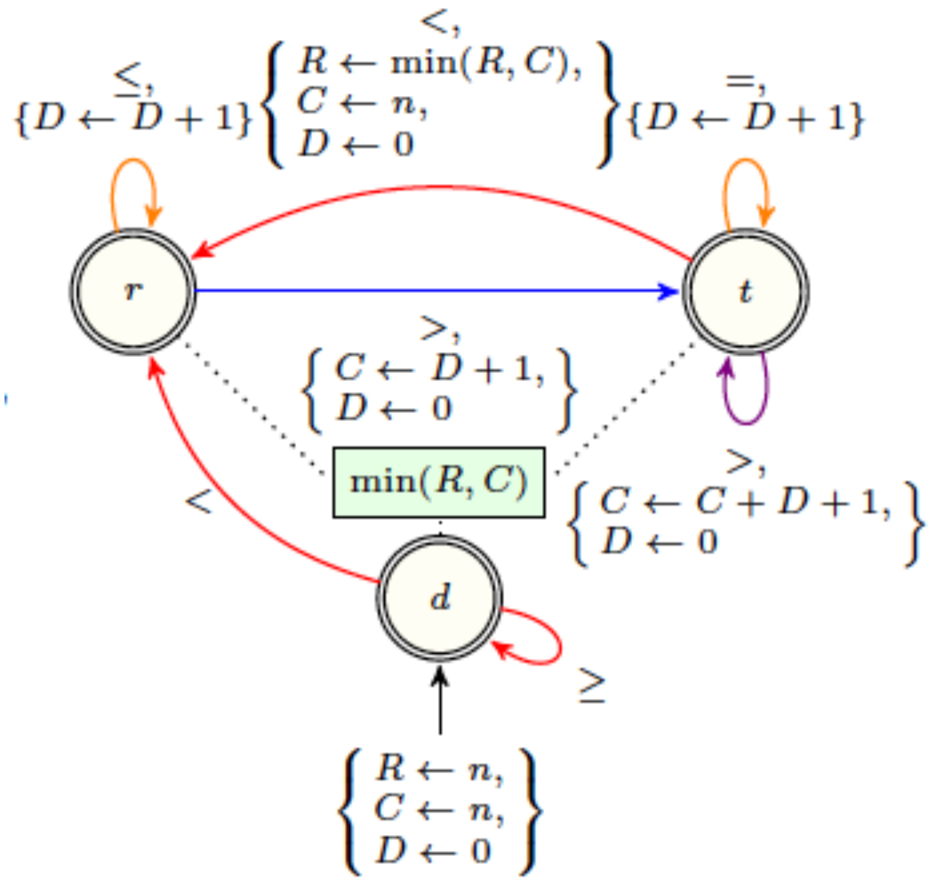


specific constraint
(min_width_peak)





Running min_width_peak



aggregate R	18	18	18	18	18	18	18	18	18	18	18	5	5	5	5	5	5	5
current C	18	18	18	18	18	18	18	18	4	5	5	18	18	18	18	18	18	6
potential D	0	0	0	0	0	1	2	3	0	0	1	0	1	2	3	4	5	0
semantic action		o	o	o	o	$\mathrm{m_b}$	$\mathrm{m_b}$	$\mathrm{m_b}$	f	in	$\mathrm{m_a}$	$\mathrm{o_a}$	$\mathrm{m_b}$	$\mathrm{m_b}$	$\mathrm{m_b}$	$\mathrm{m_b}$	$\mathrm{m_b}$	f
signatures		=	>	=	<	<	=	<	>	>	=	<	=	=	=	=	=	>
states	s	s	s	s	r	r	r	r	t	t	t	r	r	r	r	r	r	t
input	4	4	2	2	3	5	5	6	3	1	1	2	2	2	2	2	2	1

$\min(5, 6)$

- Background
- Synthesizing automata with accumulators from transducers
- Parametric glue matrices
- Simplifying automata with accumulators
- Reformulating in LP
- Deriving necessary conditions (as linear constraints)
- Bounds

Decoration table for the feature constraint

Problem: since too many time-series constraints can not afford computing the glue matrix for each constraint independently

Solution: compute **parametrized glue matrices** at the level of the transducer (*rather than at the level of each automaton*)

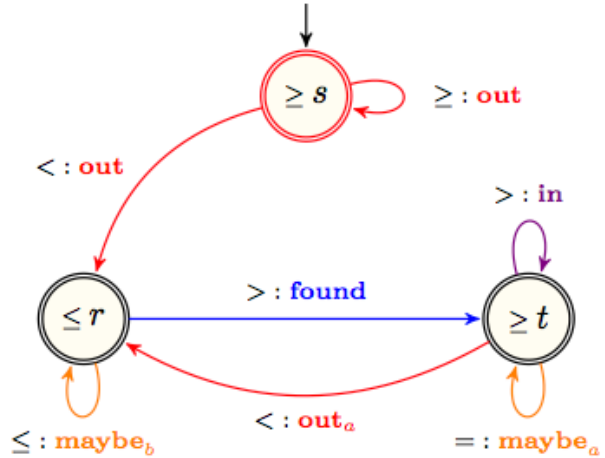
Values and functions used to parametrize generated automata (***and glue matrices***)

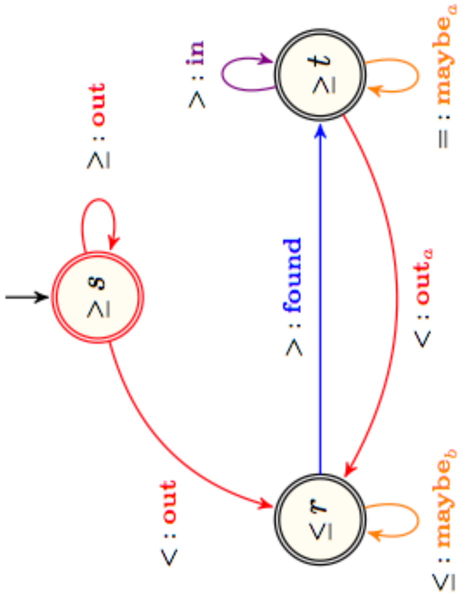
Feature f	id_f	$\min_f$	$\max_f$	ϕ_f	δ_f^i
`one`	1	1	1	1	1
`width`	0	$n+1$	$+\infty$	$+$	1
`surface`	0	$-\infty$	$+\infty$	$+$	x_i
`max`	$-\infty$	$-\infty$	$+\infty$	max	x_i
`min`	$+\infty$	$-\infty$	$+\infty$	min	x_i
`range`	0	0	$+\infty$	n/a	x_i

Aggregator g	ϕ_g	$\mathrm{default}_{gf}$
`Max`	max	$\min_f$
`Min`	min	$\max_f$
`Sum`	$+$	0

Table 2.1: (Left) Features: identity, minimum, and maximum values; the operators ϕ_f and δ_f^i recursively define the feature value v_u of a time series $x_\ell, \ldots, x_u$ by $v_\ell = \phi_f(\mathrm{id}_f, \delta_f^\ell)$ and $v_i = \phi_f(v_{i-1}, \delta_f^i)$ for $i > \ell$, where δ_f^i is the contribution of x_i to v_u; n stands for the length of the time-series. (Right) Aggregators: operators and identity values relative a feature f.

Parametrize glue matrix for peak (*is its own reverse*)



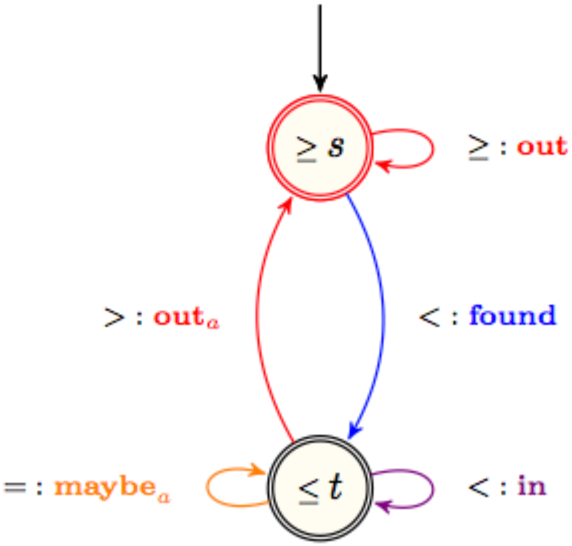


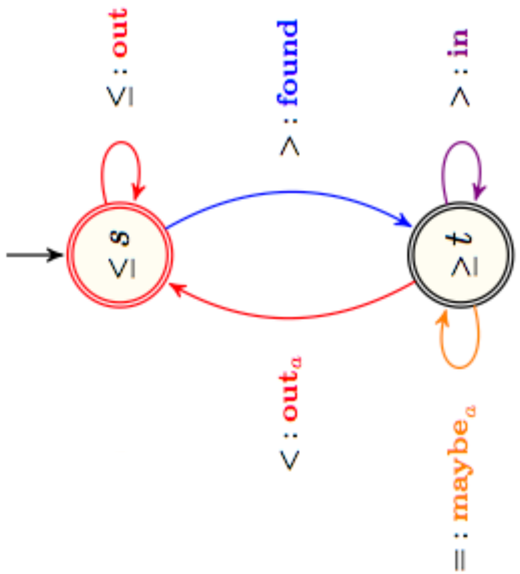
	s	r	t
s	$\phi_g(\overrightarrow{C},\overleftarrow{C})$	$\phi_g(\overrightarrow{C},\overleftarrow{C})$	$\phi_g(\overrightarrow{C},\overleftarrow{C})$
r	$\phi_g(\overrightarrow{C},\overleftarrow{C})$	$\phi_f(\overrightarrow{D},\overleftarrow{D},\delta_f^i)$	$\phi_f(\overleftarrow{C},\overrightarrow{D},\overleftarrow{D},\delta_f^i)$
t	$\phi_g(\overrightarrow{C},\overleftarrow{C})$	$\phi_f(\overrightarrow{C},\overrightarrow{D},\overleftarrow{D},\delta_f^i)$	$\phi_g(\overrightarrow{C},\overleftarrow{C})$

Parametrised glue matrix for any g_f_PEAK constraint

(define the correction term)

Parametrize glue matrix for increasing_sequence (*its reverse is* decreasing_sequence)





	s	t
s	$\phi_g(\overrightarrow{C},\overleftarrow{C})$	$\phi_g(\overrightarrow{C},\overleftarrow{C})$
t	$\phi_g(\overrightarrow{C},\overleftarrow{C})$	$\phi_f(\overrightarrow{C},\overleftarrow{C},\overrightarrow{D},\overleftarrow{D},\delta_f^i)$

Parametrised glue matrix for any g_f_INCREASING_SEQUENCE constraint

(define the correction term)

- Background
- Synthesizing automata with accumulators from transducers
- Parametric glue matrices
- Simplifying automata with accumulators
- Reformulating in LP
- Deriving necessary conditions (as linear constraints)
- Bounds

Automata simplifications

Goal

- **Reduce** the number of accumulators and aggregate as **early** as possible
- Simplify the automata at the stage of their synthesis

Three simplification types

- Simplifications coming from the properties of patterns, ex.: aggregate-once
- Simplifications coming from the properties of the feature/aggregator pairs, ex.: immediate-aggregation
- Removing the never used accumulators.

“Aggregate-once” simplification

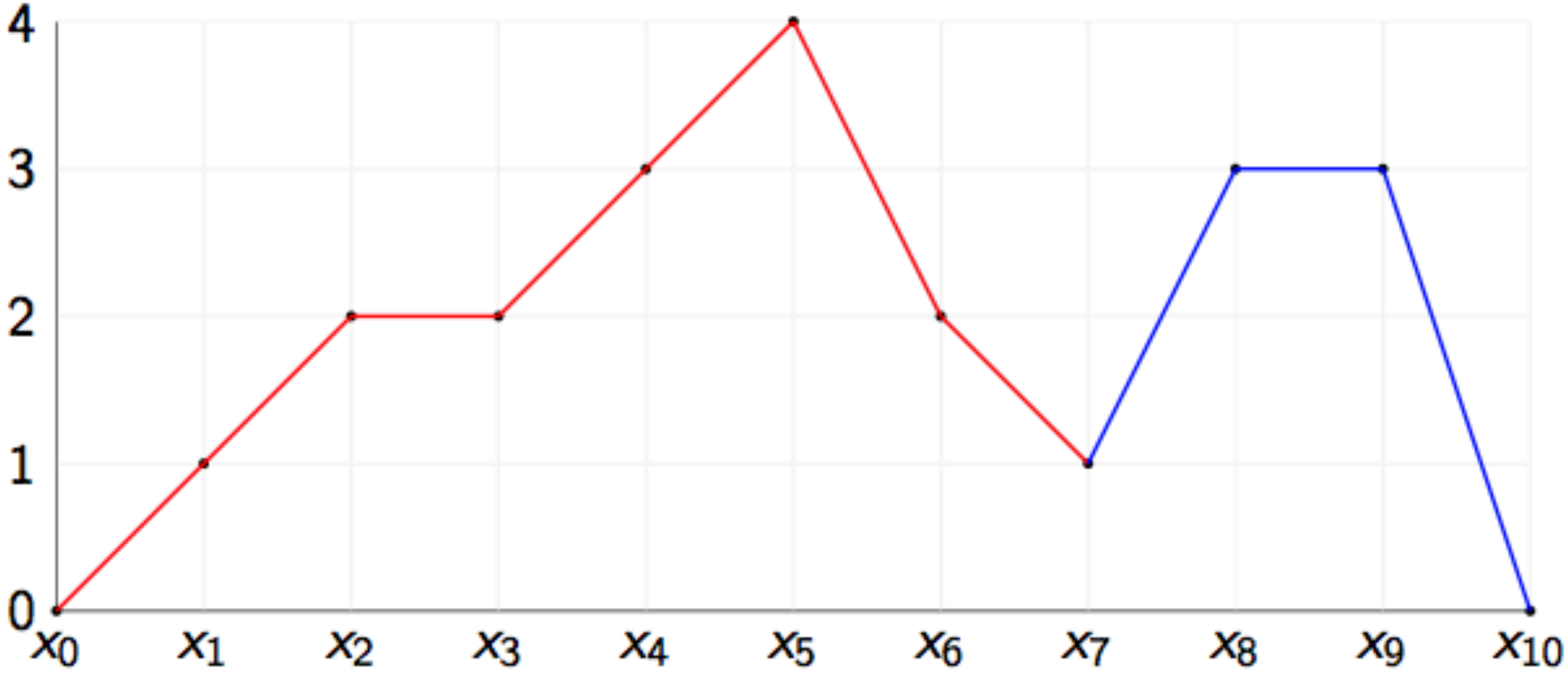
What is the “Aggregate-once” simplification ?

It allows to compute the feature value of a curent pattern occurrence only once and, possibly, earlier than the end of a pattern occurrence.

When is the simplification applicable ?

There must exist a transition on which the value of the feature from the current pattern occurrence is known.

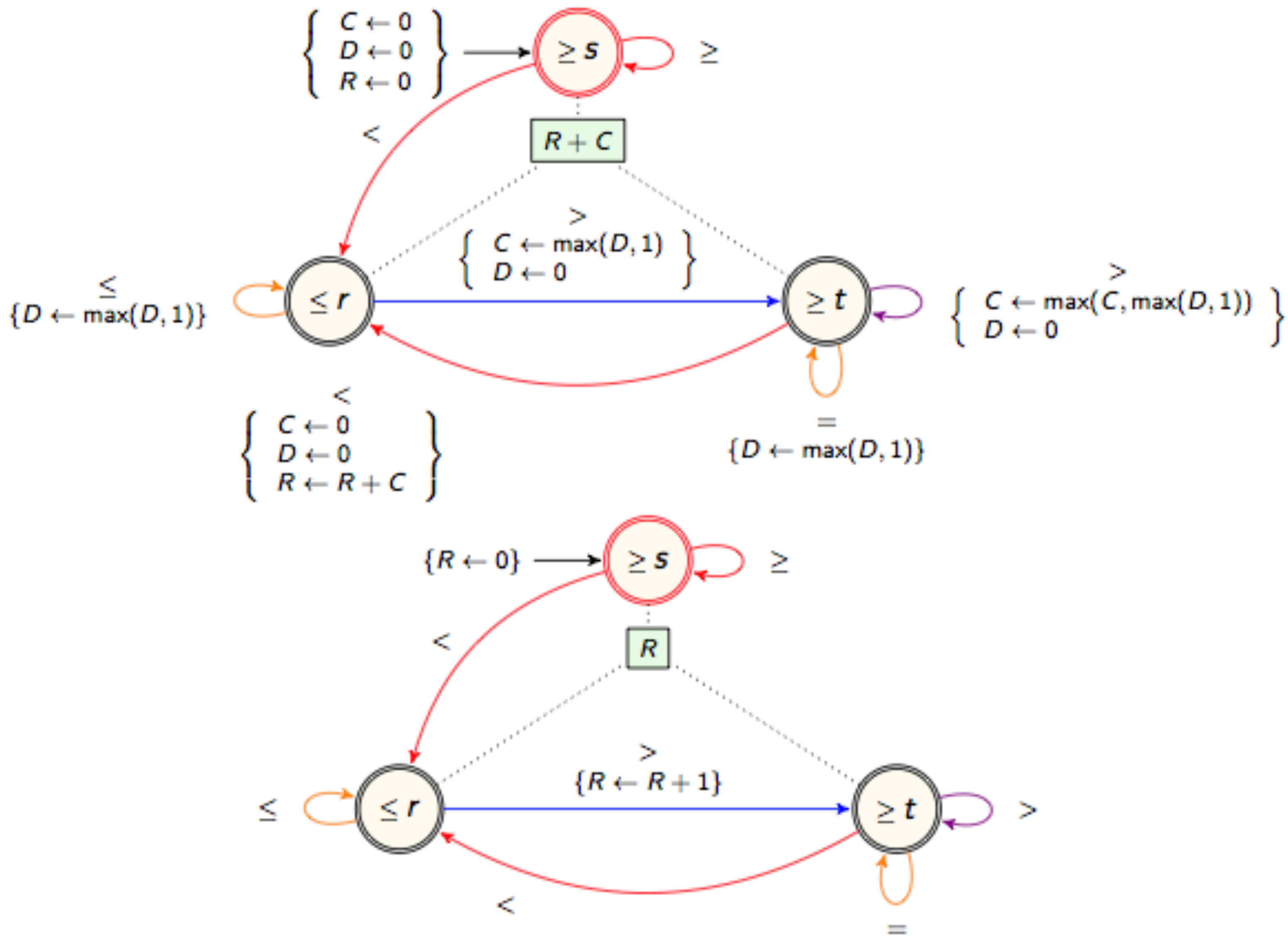
Example: counting number of peaks



$s_0 =$'$<$' $s_1 =$'$<$' $s_2 =$'$=$' $s_3 =$'$<$' $s_4 =$'$<$' $s_5 =$'$>$' $s_6 =$'$>$' $s_7 =$'$<$' $s_8 =$'$=$' $s_9 =$'$>$'

1. First peak is detected upon consuming s_5
2. Second peak is detected upon consuming s_9

Two automata for nb_peak



Percentage of simplified constraints

Simplification	Percentage
aggregate once	28.9 %
immediate aggreg.	45.9 %
other properties	11.6 %
unchanged automata	13.6 %

- Background
- Synthesizing automata with accumulators from transducers
- Parametric glue matrices
- Simplifying automata with accumulators
- Reformulating in LP
- Deriving necessary conditions (as linear constraints)
- Bounds

Goal

Goal

A way to generate a model for an automaton with linear or linearisable accumulator updates, for example containing min and max.

Linear decomposition of automata without accumulators

Côté, M.C., Gendron, B., Rousseau, L.M.: Modeling the regular constraint with integer programming. In: CPAIOR 2007. LNCS, vol. 4510, pp. 29–43. Springer (2007)

Signature constraint

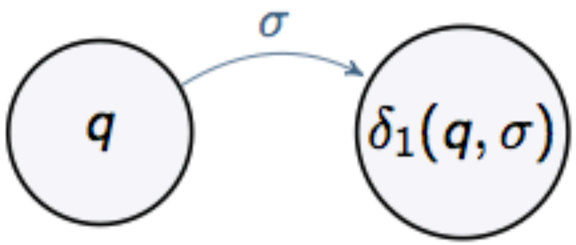
Introduced variables: S_i over Σ with $i \in [0, n-2]$.

What do the values of S_i mean ?

$$S_i = \text{'>'} \Leftrightarrow X_i > X_{i+1}, \forall i \in [0, n-2]$$
$$S_i = \text{'='} \Leftrightarrow X_i = X_{i+1}, \forall i \in [0, n-2]$$
$$S_i = \text{'<'} \Leftrightarrow X_i < X_{i+1}, \forall i \in [0, n-2]$$

Transition function constraints

Introduced variables: Q_i over Q with $i \in [0, n-1]$; T_i over $Q \times \Sigma$ with $i \in [0, n-2]$



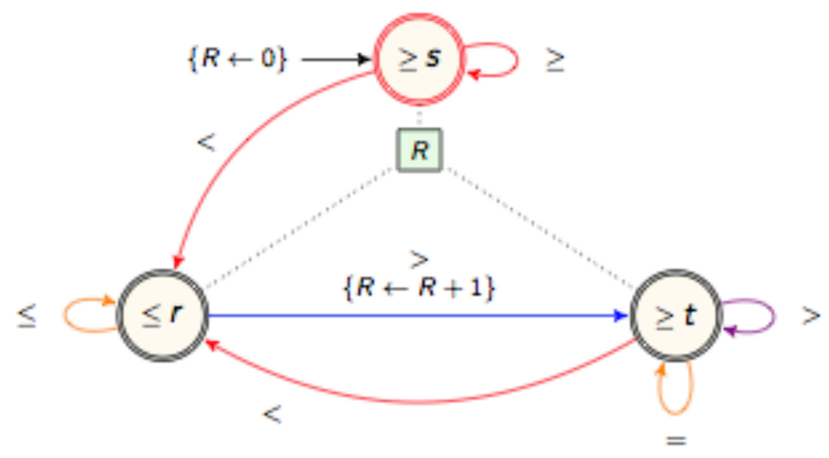
Each transition constraint has a form:

$Q_i = q \wedge S_i = \sigma \Leftrightarrow Q_{i+1} = \delta_1(q, \sigma) \wedge T_i = \langle q, \sigma \rangle$,
$\forall i \in [0, n-2],\ \forall q \in Q,\ \forall \sigma \in \Sigma$

Initial state is fixed

$Q_0 = q_0$

Accumulator updates



Accumulator updates

R_i over $[a, b]$ with i in $[0, n-1]$; T_i over $Q \times \Sigma$ with i in $[0, n-2]$.

- $R_0 = 0$
- $T_i = \langle r, > \rangle \Rightarrow \mathbf{R_{i+1} = R_i + 1}, \forall i \in [0, n-2]$
- $T_i = \langle q, \sigma \rangle \Rightarrow \mathbf{R_{i+1} = R_i}, \forall i \in [0, n-2], \forall \langle q, \sigma \rangle \in (Q \times \Sigma) \setminus \langle r, > \rangle$
- $M = R_{n-1}$

New variables for the linear model

New variables

- Q_i is replaced by 0-1 variables Q_i^q for all q in Q.
  $Q_i^q = 1 \Leftrightarrow Q_i = q$
- New constraint: $\sum_{q \in Q} Q_i^q = 1, \forall i \in [0, \dots, n-1]$
- The same procedure for T_i and S_i wrt their domains
- X_i and R_i remain integer variables!

- Every constraint of the logical model is made linear by applying some standard techniques
- The linear model has $O(n)$ variables and $O(n)$ constraints

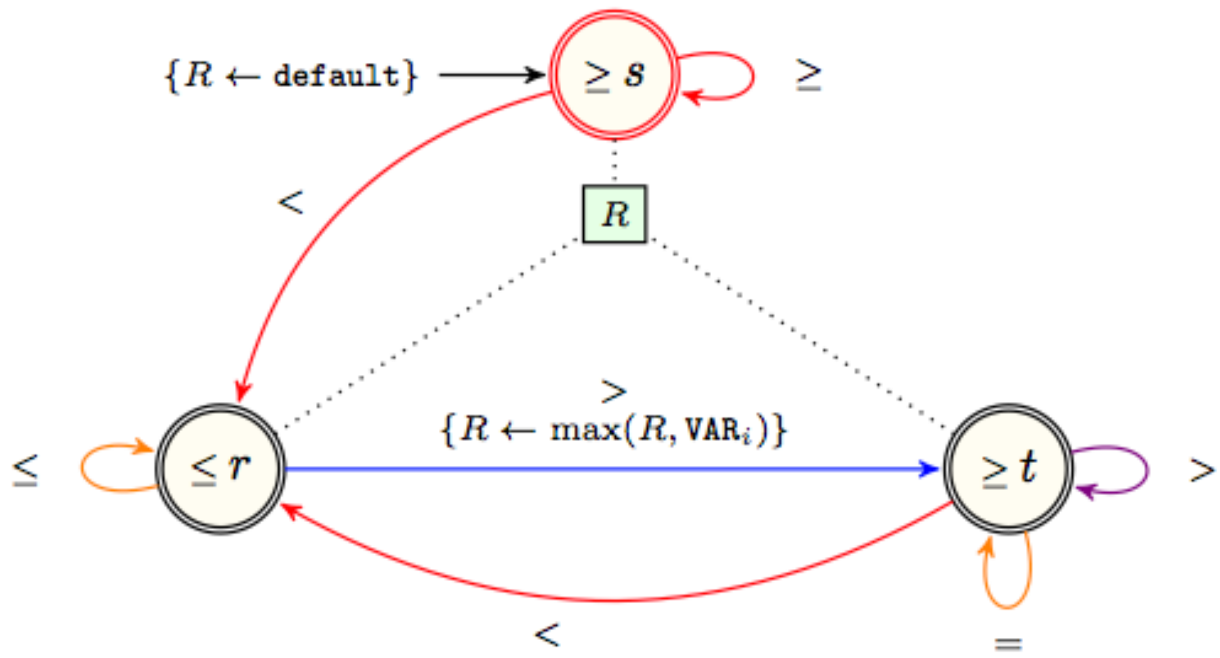
- Background
- Synthesizing automata with accumulators from transducers
- Parametric glue matrices
- Simplifying automata with accumulators
- Reformulating in LP
- Deriving necessary conditions (as linear constraints)
- Bounds

Generating necessary conditions (*as linear constraints*)

- Good news:
  - Can use/adapt standard LP techniques (Farkas Lemma) to generate necessary conditions (*expressed as linear constraints*)
  - Can use this even if accumulators updates are min/max operations
  - Can rank the linear constraints
  - Useful both for CP and LP
  - The invariants neither depend of the domain of the variables, nor on the size of the sequence.

Leads to a data base of cuts for time-series constraints

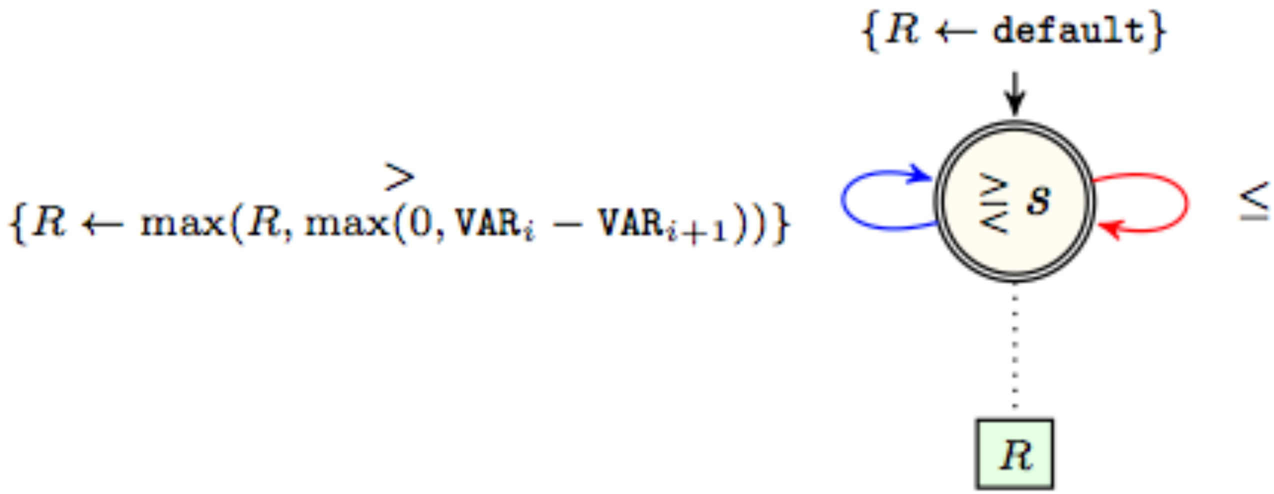
Examples of linear constraints
(max_max_peak)



$$R_i - R_{i-1} \geq 0$$

(increasing R since use max aggregator)

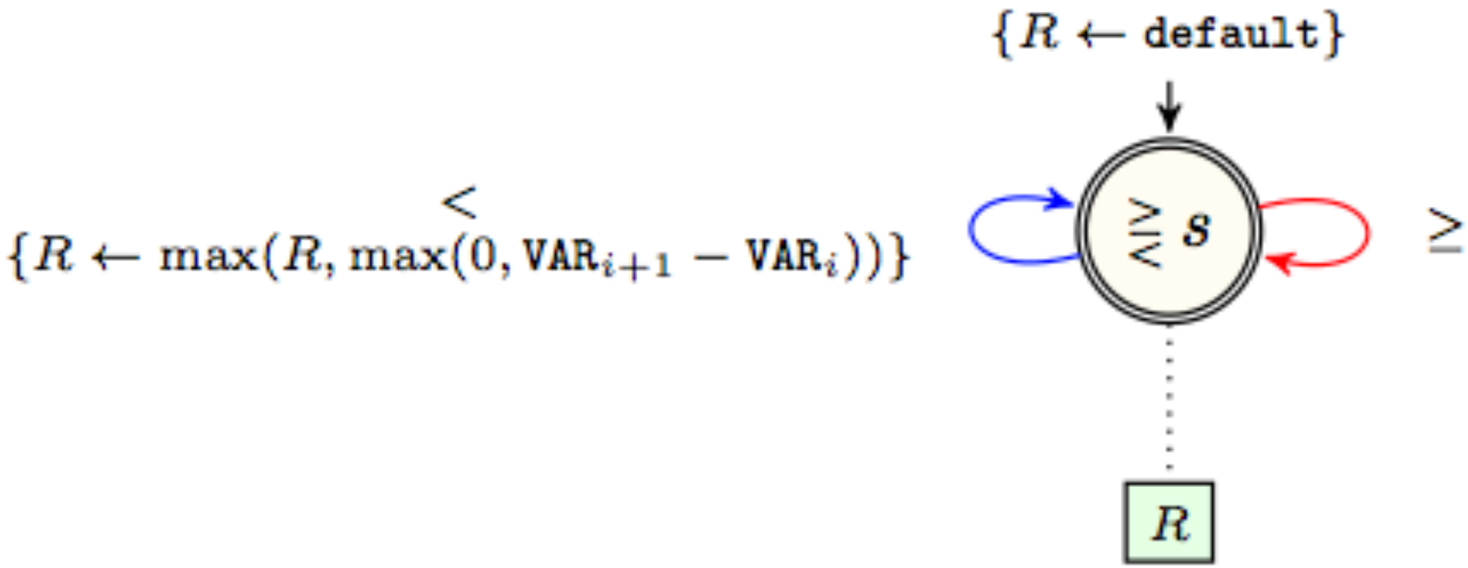
Examples of linear constraints (max_range_decreasing)



$R_i - R_{i-1} \geq 0$ **(increasing R since use max aggregator)**

$R_i + \mathtt{VAR}_{i-1} - \mathtt{VAR}_{i-2} \geq 0$

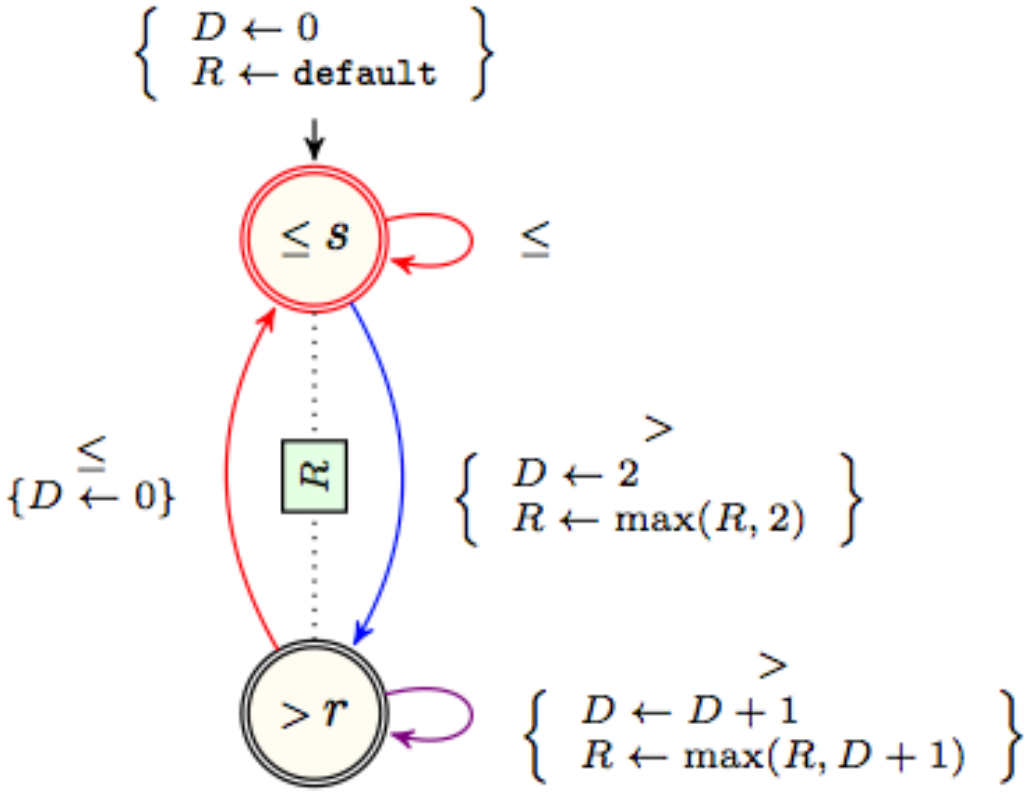
Examples of linear constraints (max_range_increasing)



$$R_i - R_{i-1} \geq 0$$ **(increasing R since use max aggregator)**

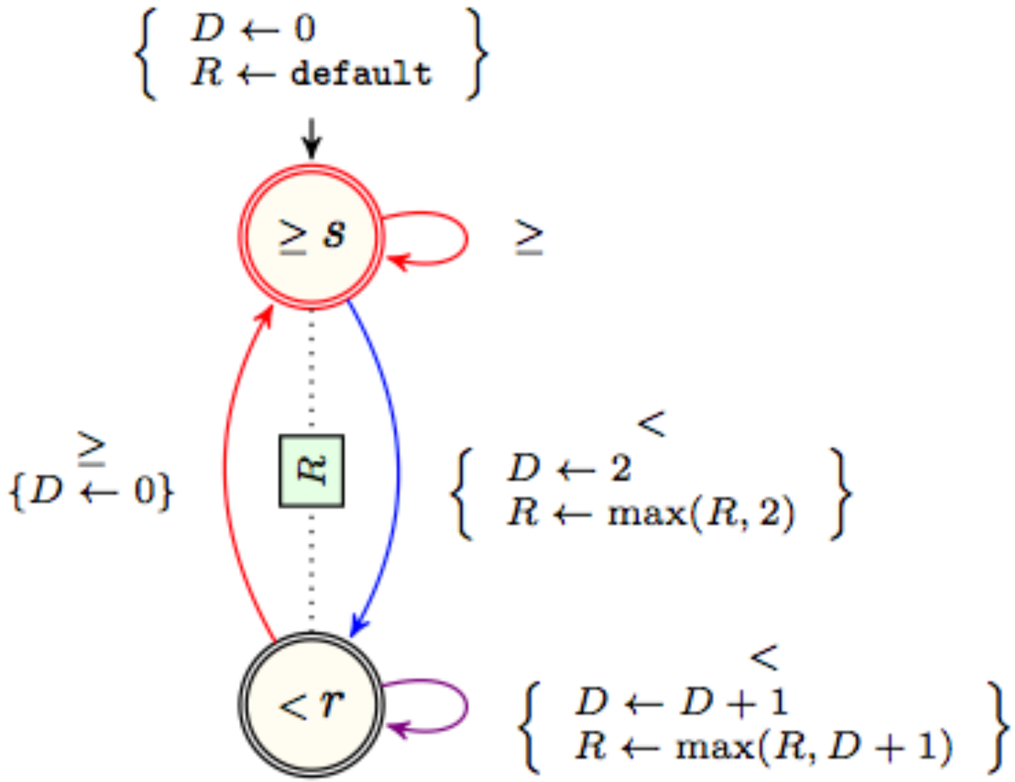
$$R_i - \mathtt{VAR}_{i-1} + \mathtt{VAR}_{i-2} \geq 0$$

Examples of linear constraints (max_width_strictly_decreasing_sequence)



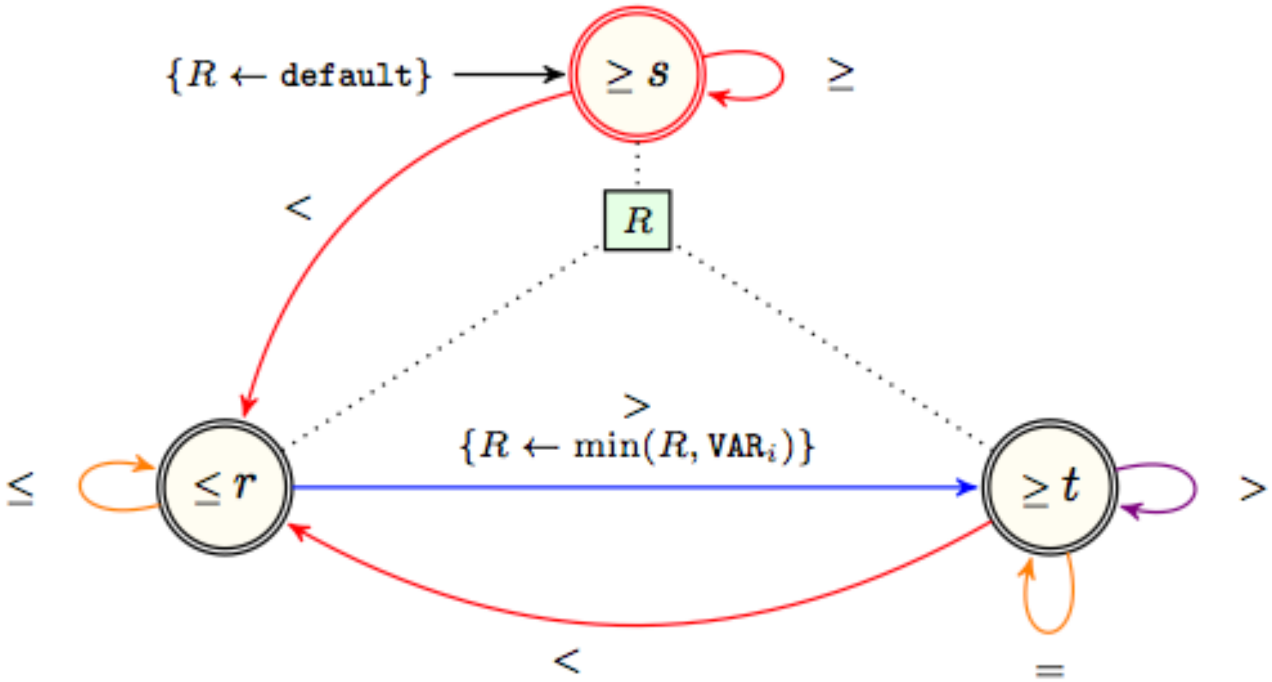
$R_i - R_{i-1} \geq 0$ **(increasing R since use max aggregator)**

Examples of linear constraints
(max_width_strictly_increasing_sequence)



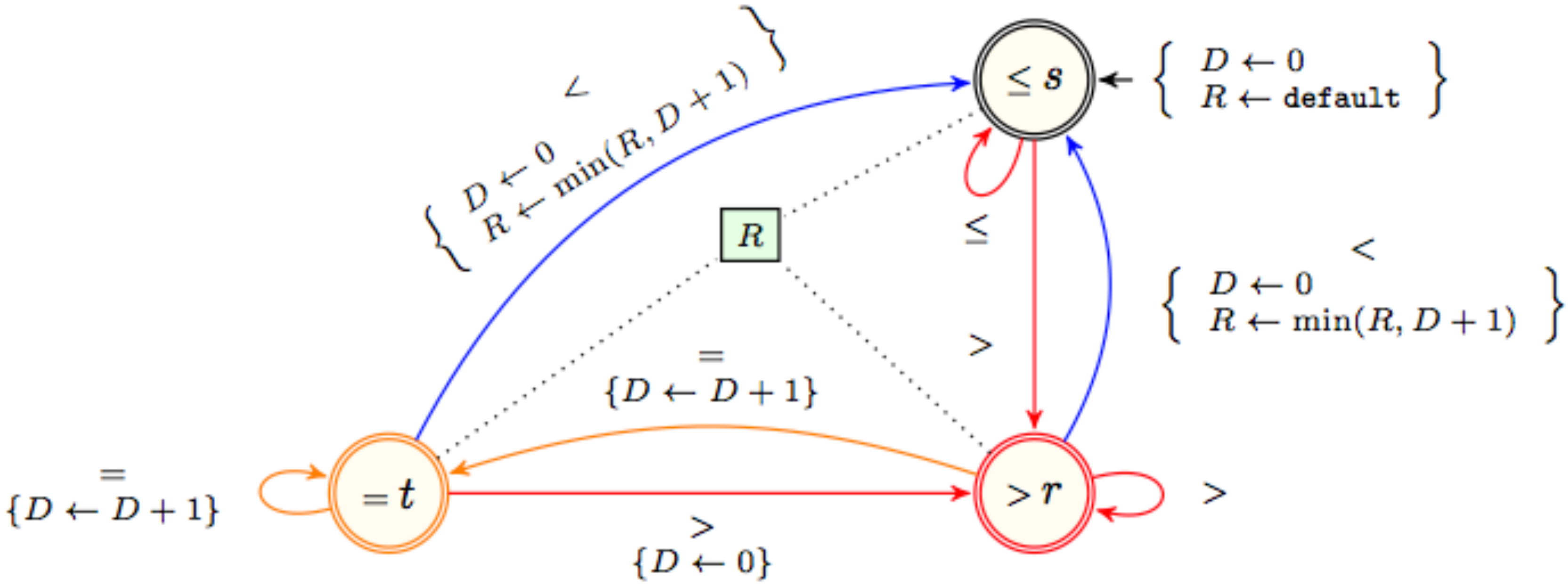
$R_i - R_{i-1} \geq 0$ **(increasing R since use max aggregator)**

Examples of linear constraints
(min_max_peak)



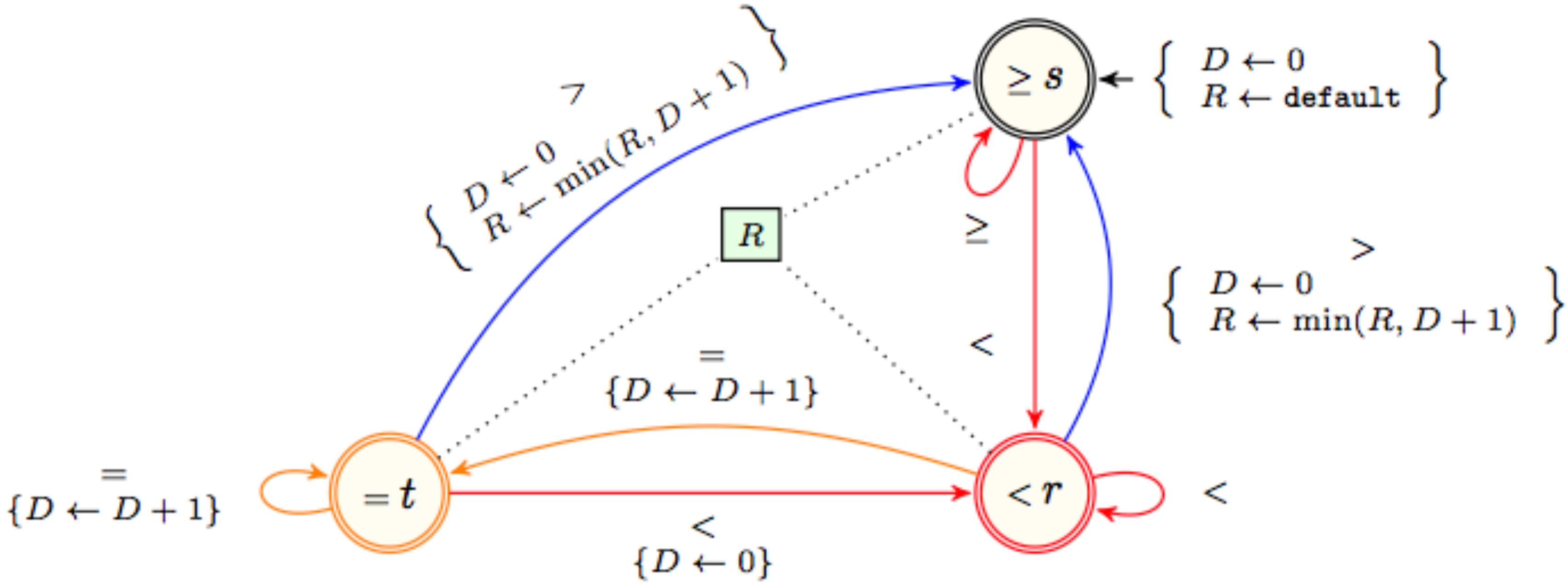
$$-R_i + R_{i-1} \geq 0$$ **(decreasing R since use min aggregator)**

Examples of linear constraints
(min_width_plain)



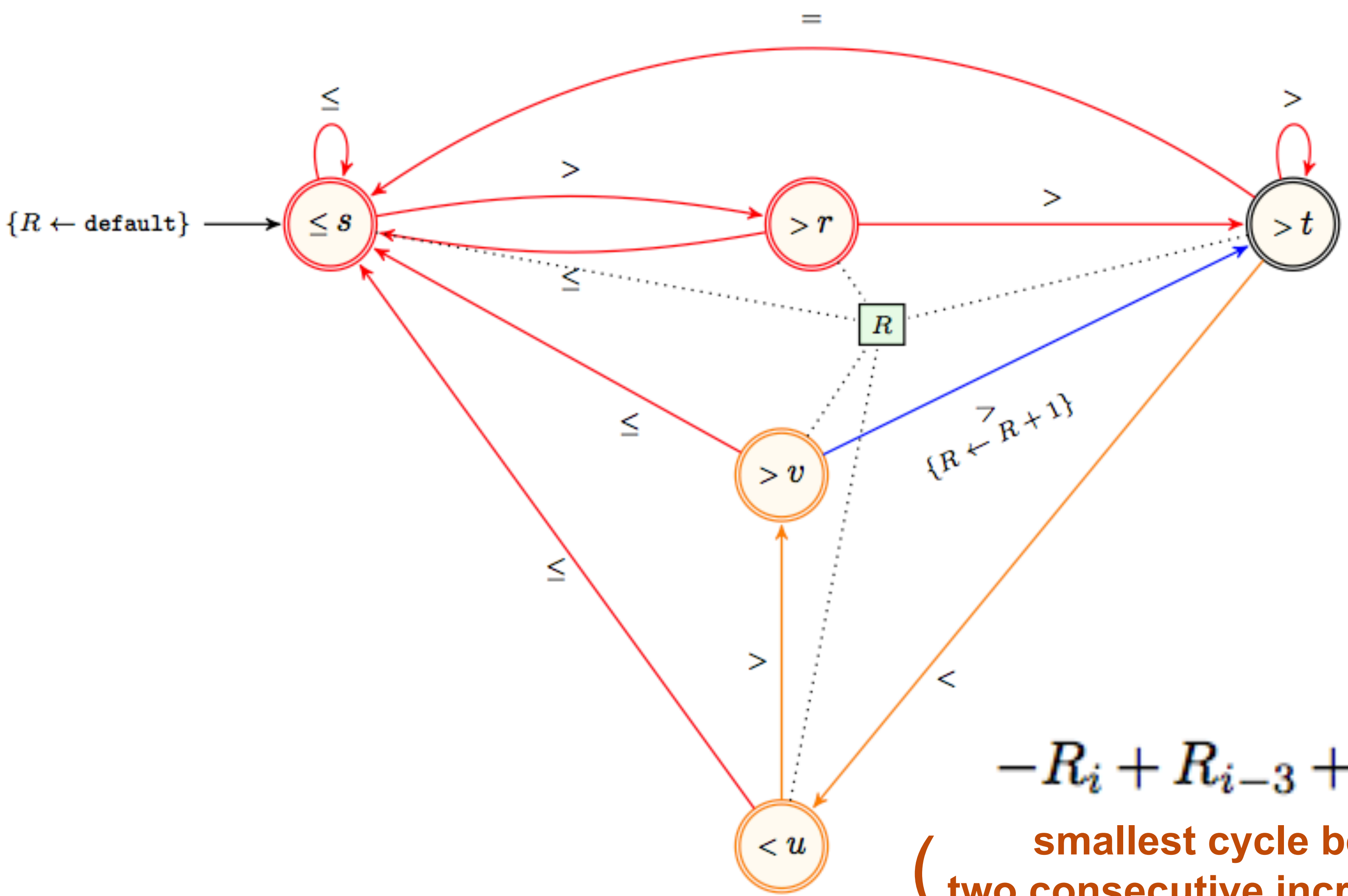
$$-R_i + R_{i-1} \geq 0$$ **(decreasing R since use min aggregator)**

Examples of linear constraints
(min_width_plateau)



$$-R_i + R_{i-1} \geq 0$$ **(decreasing R since use min aggregator)**

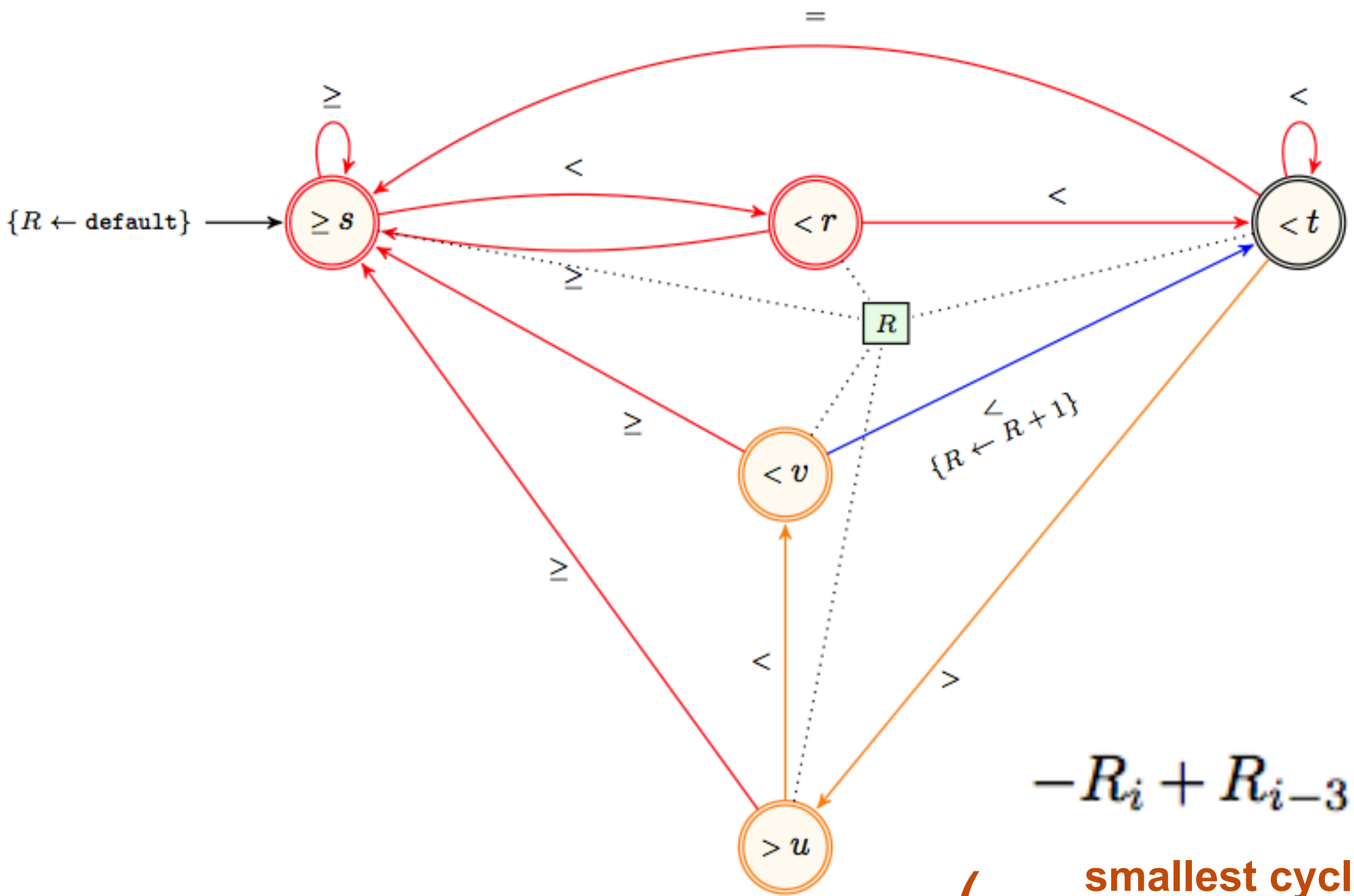
Examples of linear constraints
(nb_bump_on_decreasing_sequence)



$$-R_i + R_{i-3} + 1 \geq 0$$

(smallest cycle between two consecutive incrementation)

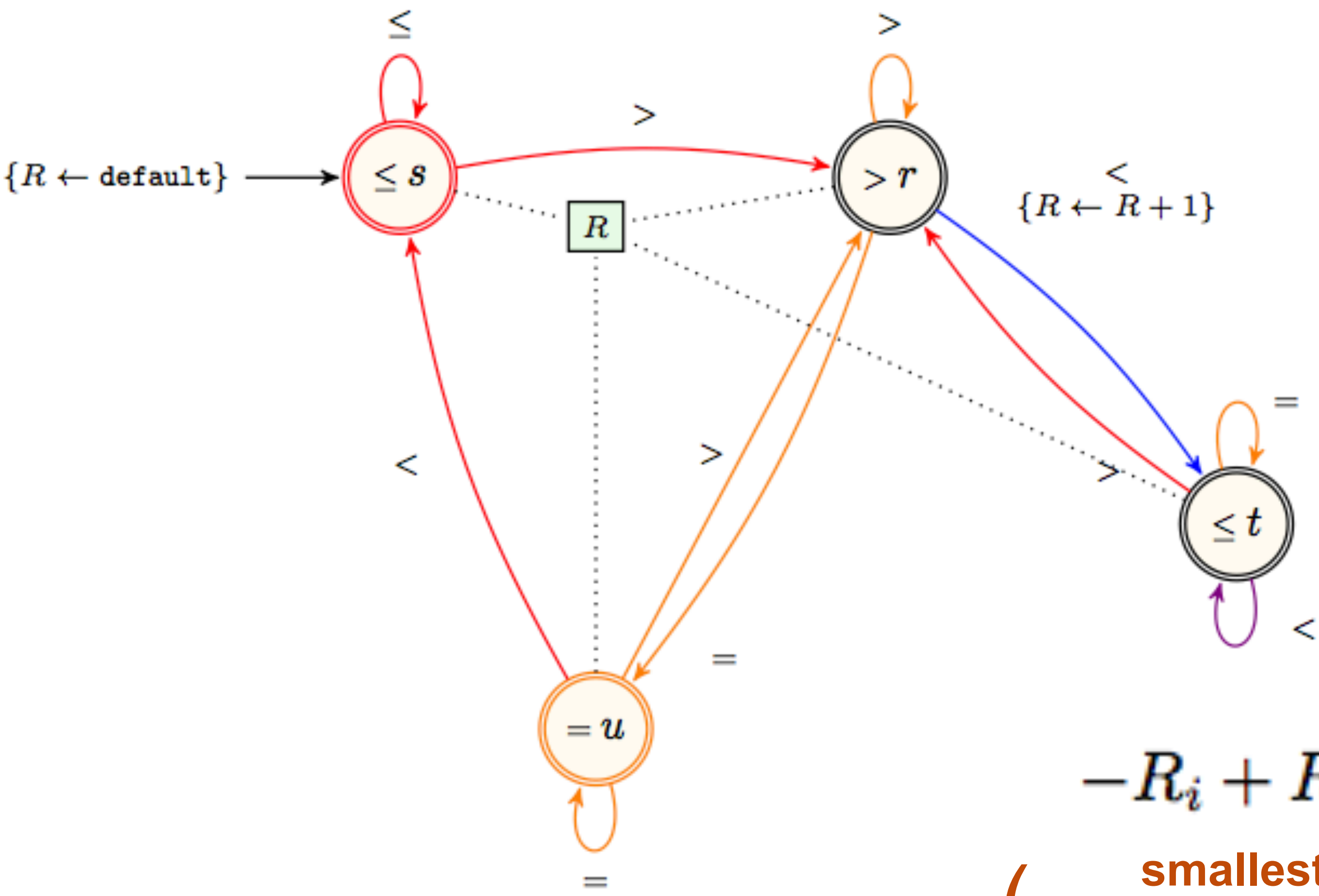
Examples of linear constraints
(nb_dip_on_increasing_sequence)



$$-R_i + R_{i-3} + 1 \geq 0$$

(smallest cycle between two consecutive incrementation)

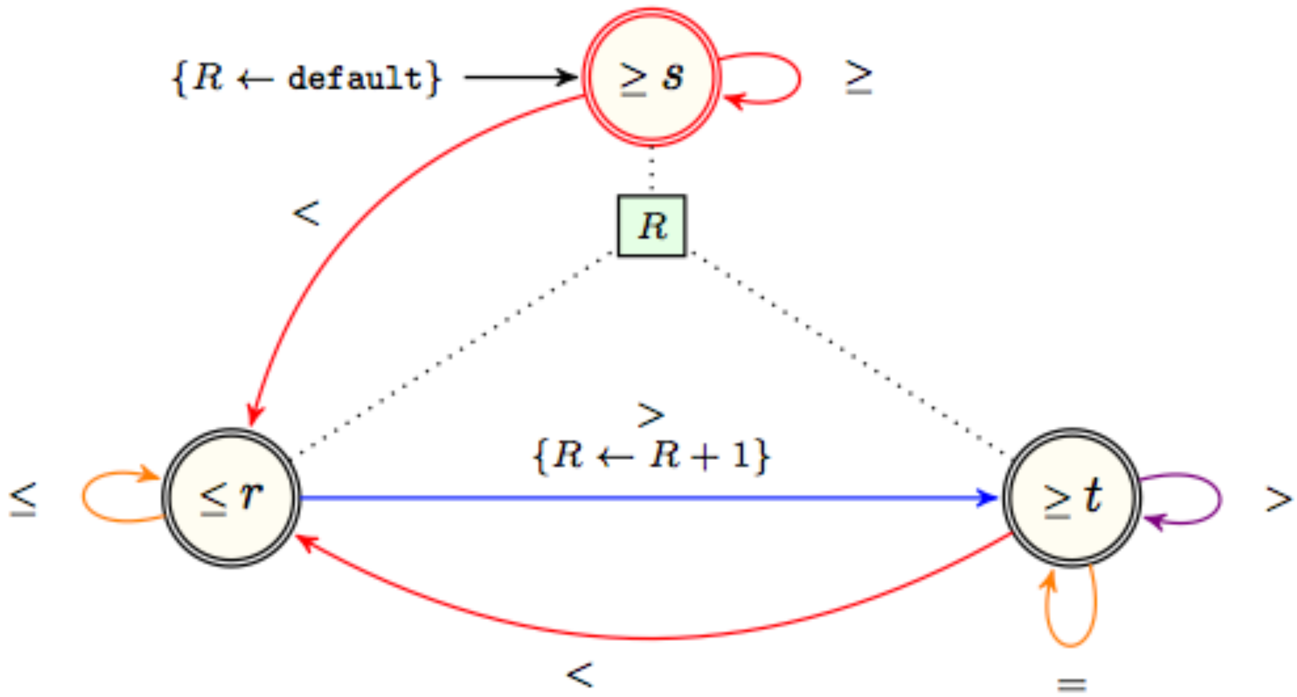
Examples of linear constraints
(nb_gorge)



$$-R_i + R_{i-2} + 1 \geq 0$$

(**smallest cycle between two consecutive incrementation**)

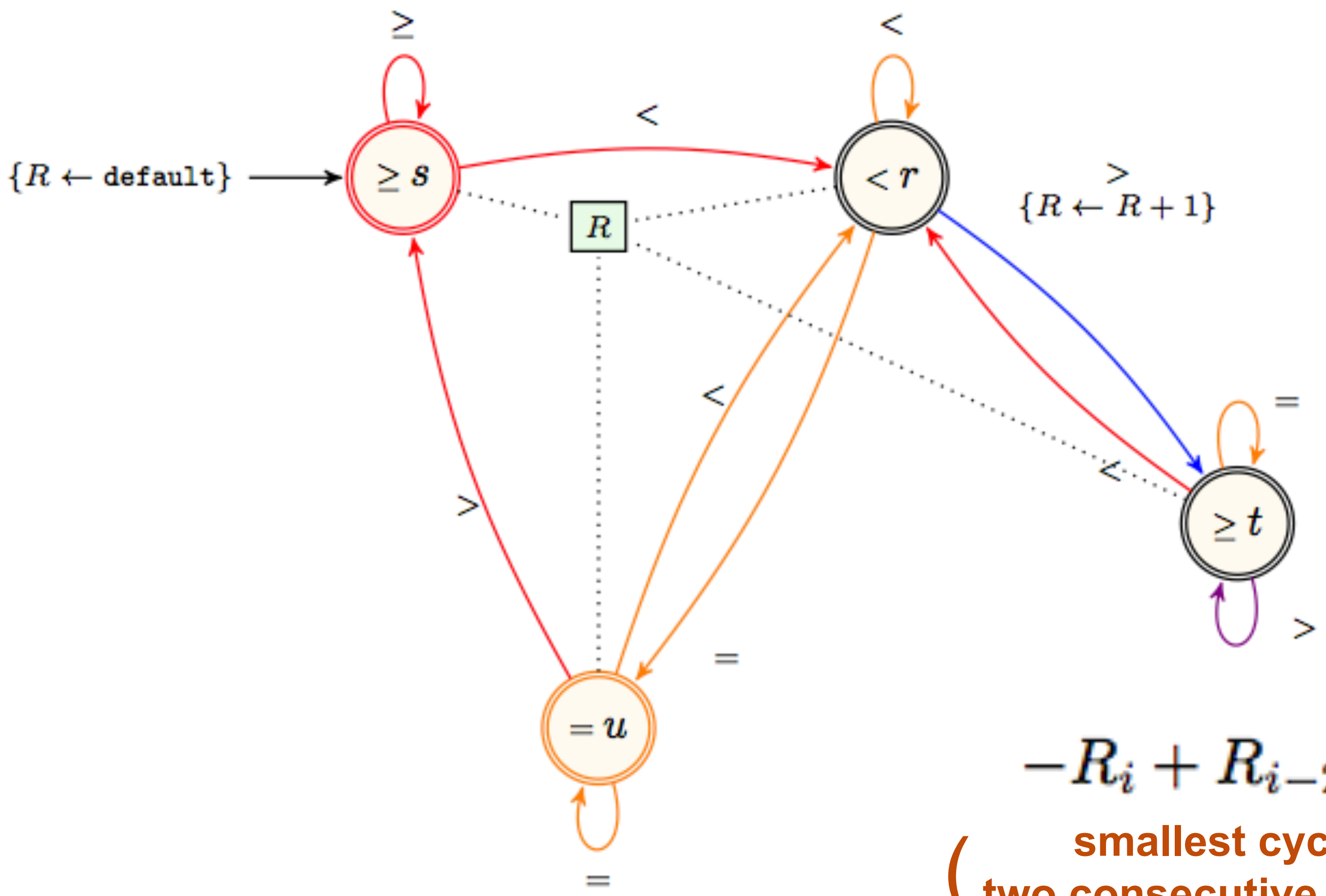
Examples of linear constraints
(nb_peak)



$$-R_i + R_{i-2} + 1 \geq 0 \qquad -2R_i + R_{i-1} + R_{i-2} + 2 \geq 0$$

(smallest cycle between two consecutive incrementation)

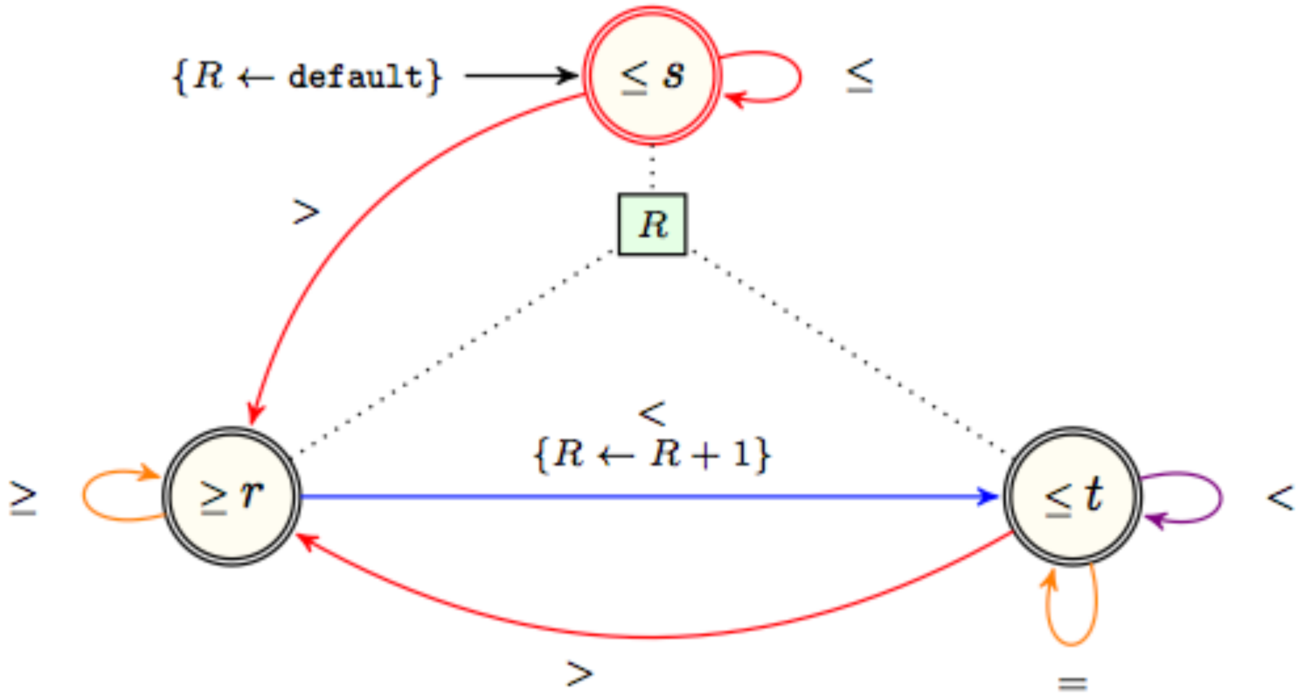
Examples of linear constraints
(nb_summit)



$$-R_i + R_{i-2} + 1 \geq 0$$

(**smallest cycle between two consecutive incrementation**)

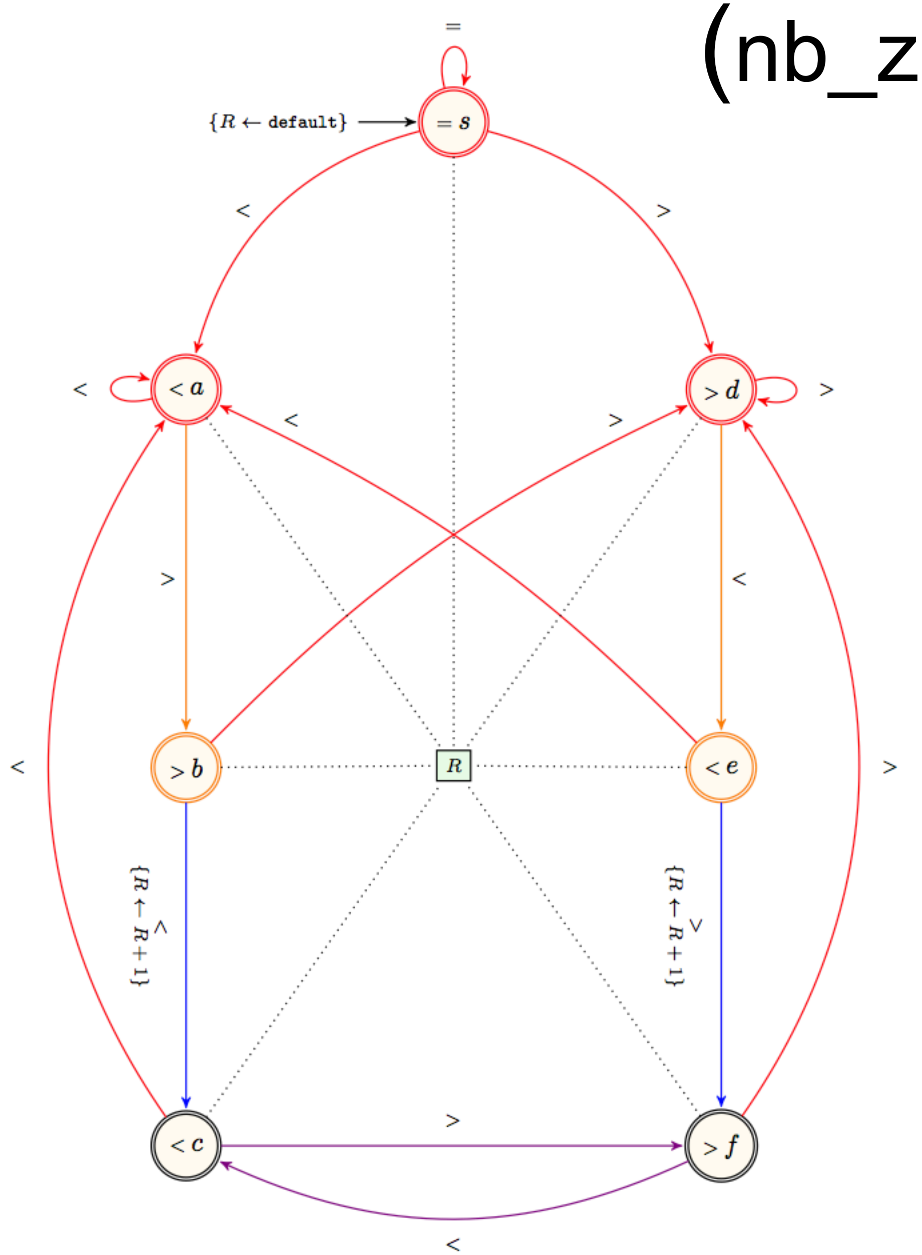
Examples of linear constraints
(nb_valley)



$$-R_i + R_{i-2} + 1 \geq 0 \qquad -2R_i + R_{i-1} + R_{i-2} + 2 \geq 0$$

(smallest cycle between two consecutive incrementation)

Examples of linear constraints
(nb_zigzag)



$$R_i - R_{i-1} \geq 0$$

(**increasing R since use sum aggregator and feature one**)

$$-R_i + R_{i-3} + 1 \geq 0$$

(**smallest cycle between two consecutive incrementation**)

- Background
- Synthesizing automata with accumulators from transducers
- Parametric glue matrices
- Simplifying automata with accumulators
- Reformulating in LP
- Deriving necessary conditions (as linear constraints)
- Bounds

The need for bound

- Want to have lower/upper bound on the result returned by a time-series constraint parametrized by:
  - The sequence length
  - The smallest or largest values of the variables in the sequence

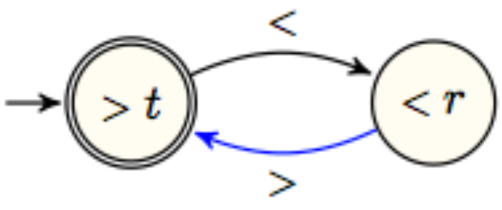
Having a bound is good,
but having a way to characterize
all solutions reaching this bound is even better

Point

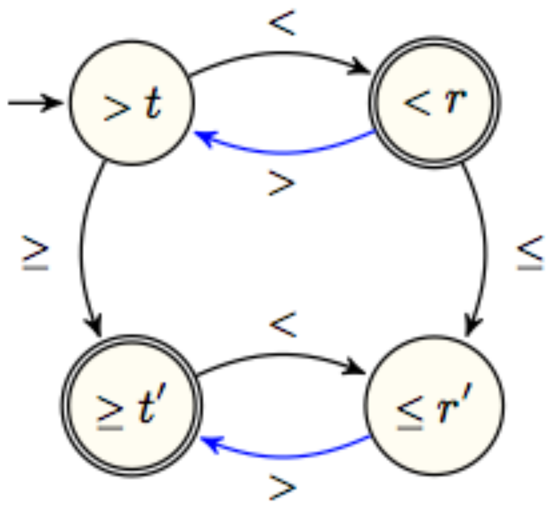
- In a significant number of cases we can **just use the standard regular constraint** (with an automaton having a **fixed number of states**) for characterizing all solutions reaching a given bound

when the bound does not depend of the domain size

Example 1: nb_peak (upper bound $\lfloor \frac{n-1}{2} \rfloor$)



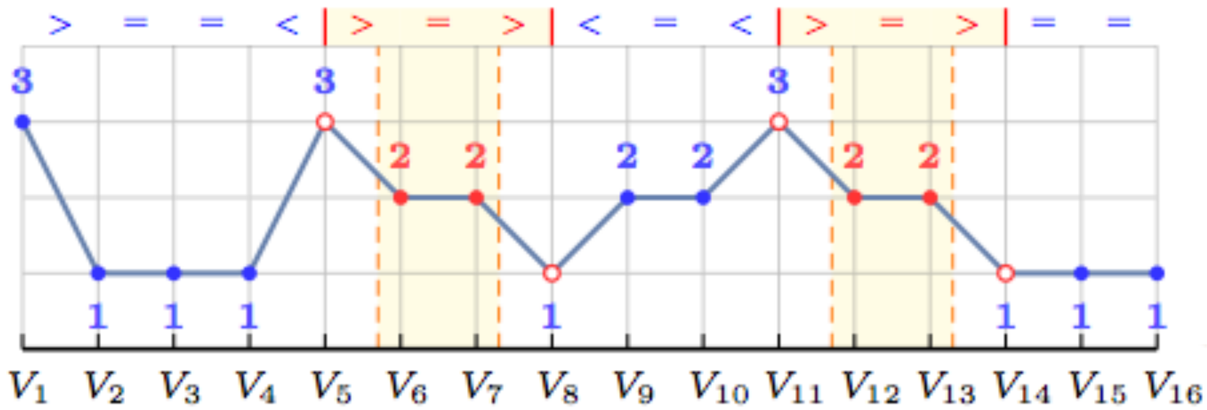
$n \bmod 2 = 1$



$n \bmod 2 = 0$

$0, 1, 0$	$0, 0, 1, 0$	$0, 1, 0, 0$	$0, 1, 0, 1$	$0, 1, 1, 0$	$1, 0, 1, 0$
$<>$	$=<>$	$<>=$	$<><$	$<=>$	$><>$

Exercise: get a transducer for the decreasing_terrace pattern (>=+>)



Exercise: transducer for decreasing_terrace