Using Finite Transducers for Describing and Synthesising Structural Time-Series Constraints

Octobre 2017

Nicolas Beldiceanu

IMT Atlantique

Take-away points

- Constraints defined by **function composition**
- New family of constraints for **time-series**
- Applications in **data analysis** as well as **optimization**

Outline

- **Background**
- Using finite transducers for capturing patterns
- Types of time-series constraints

Time-series constraints

- Introduced 30 constraints to describe properties of time-series
- Constraints were implemented as **automata with counters**
- Also use **specialized** constraint checkers for analysis
- Hard to correctly implement/**maintain**
- Always **new** constraints to add (*business rules*)

Key observations (*at the root of this research*)

- Almost all automata based constraints of the catalog deal with **restricting a quantity computed** from occurrences of **non-overlapping patterns**
- Transitions of the counter automata have an **implicit semantics** (*steps for identifying a pattern*)

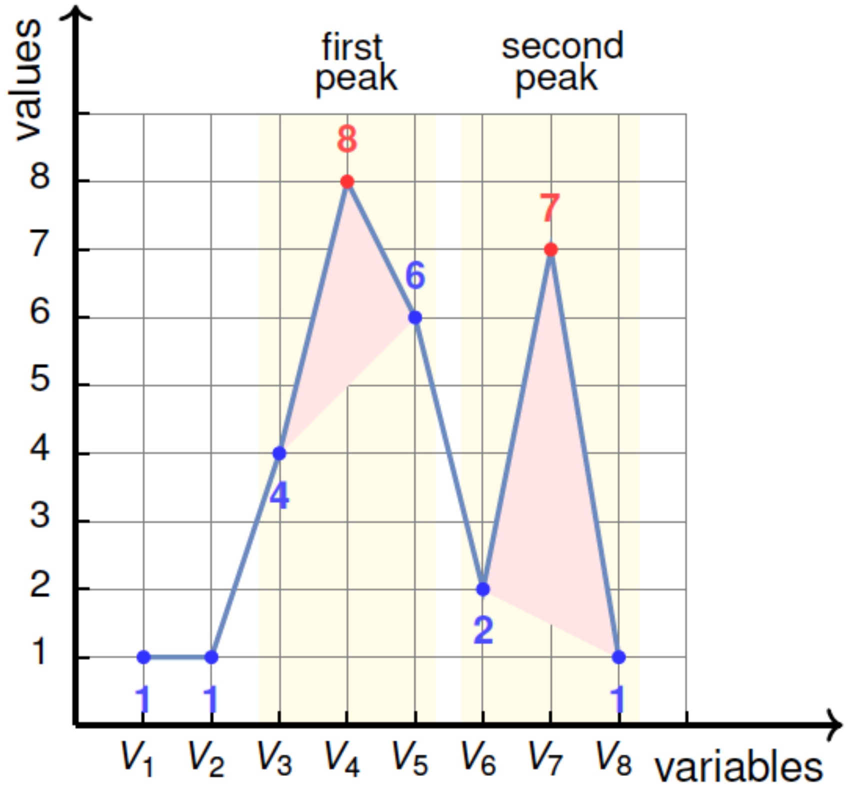
Key observations (*at the root of this research*)

- Almost all automata based constraints of the catalog deal with **restricting a quantiy computed** from occurrences of **non-overlapping patterns**

- Transitions of the counter automata have an **implicit semantics** (*steps for identifying a pattern*)

Make ***explicit*** *the implicit semantics attached to the transitions by using* ***the output alphabet of a transducer*** *that controls the* ***generation*** *of the counter updates used for computing the quantity to restrict*

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	$(< \{< \mid =\}^*)$

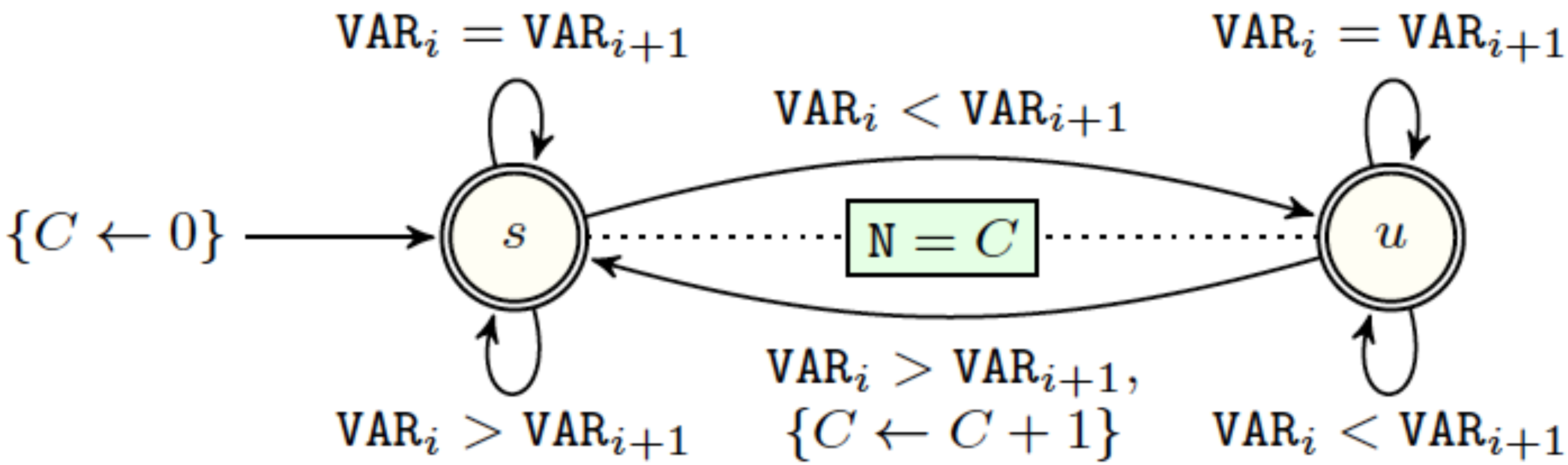


Figure 5.689: Automaton of the PEAK constraint

Outline

- Background
- **Using finite transducers for capturing patterns**
- Types of time-series constraints

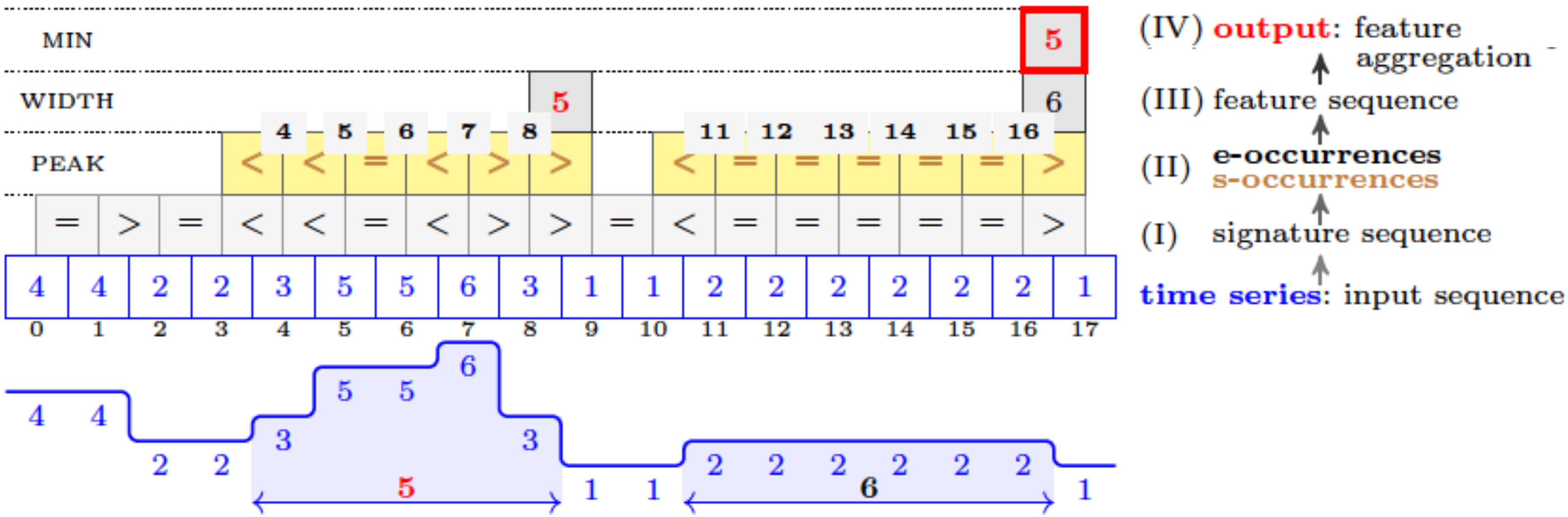
Work program: define the family of structural time-series constraints

- Define constraints in a **systematic** way **independantly** from a given technology (CP, LP, LS)
- Derive **all** documention/code **automatically** from compact description
- Apply optimizations **consistently**
- Use for **multiple** use cases

Decomposing the definition of a constraint

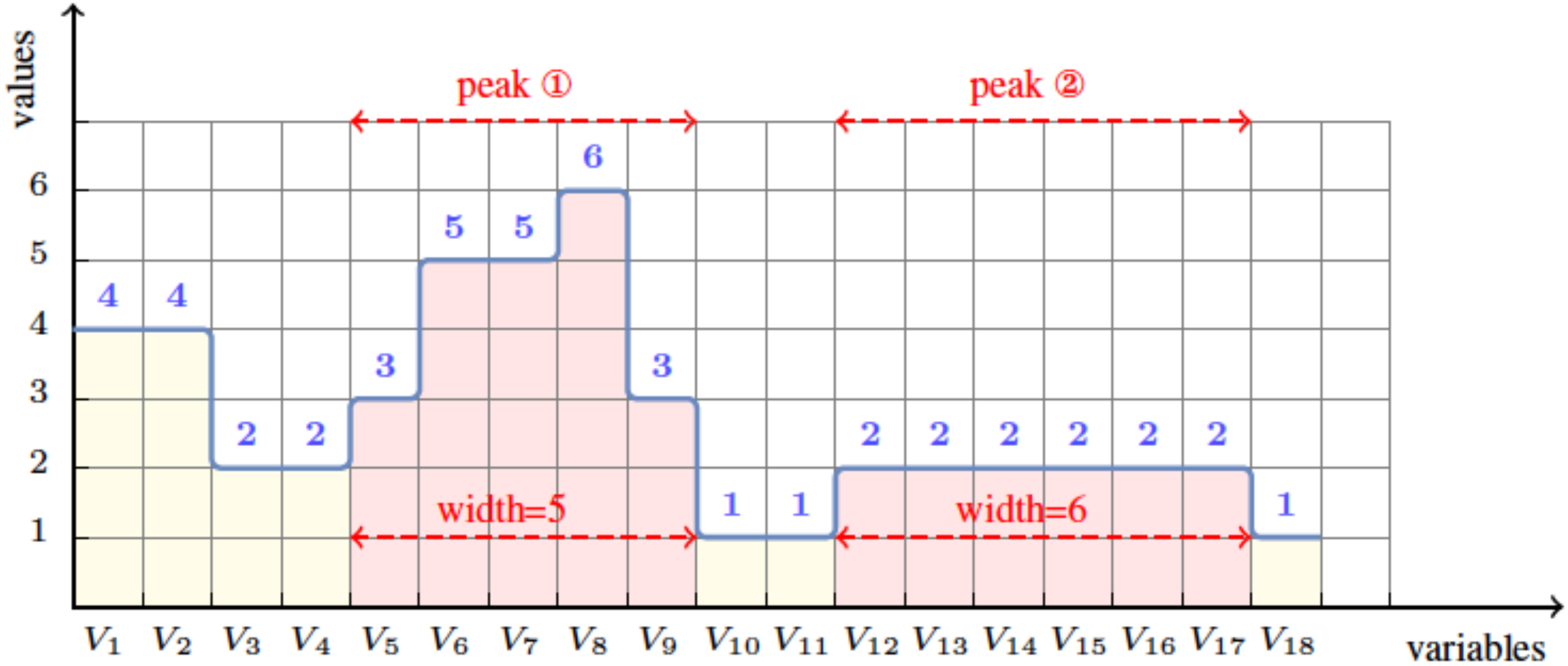
- Constraints are **pure functional dependencies** or **predicates** (*see predicates later on*)
- Implemented as **automata with counters**
- **Four steps** (layers) in definition
  - Building **signature**
  - Recognize **pattern occurrences** in sequence
  - Extract **feature** per pattern
  - **Aggregate** 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

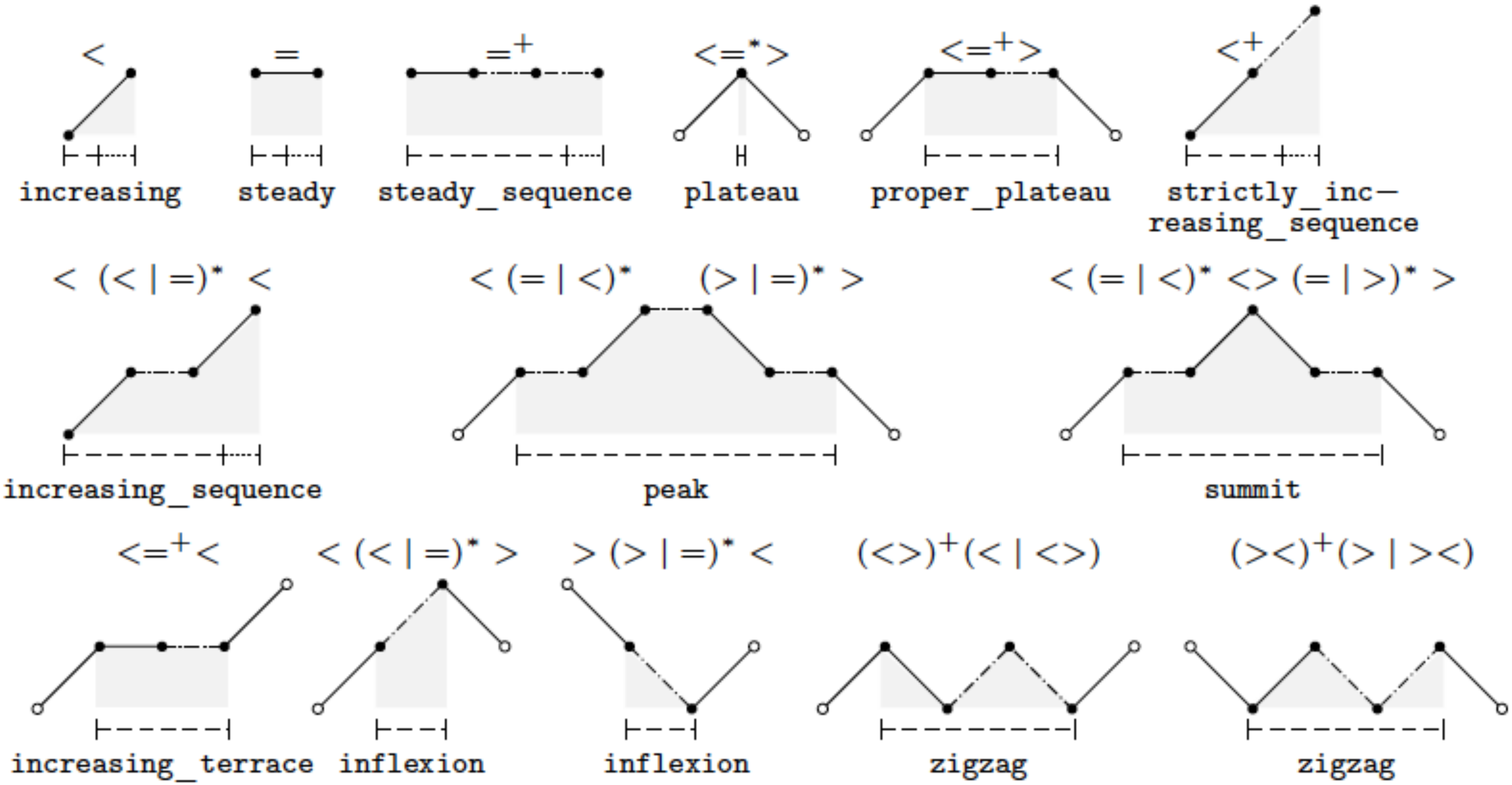
Patterns

- **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

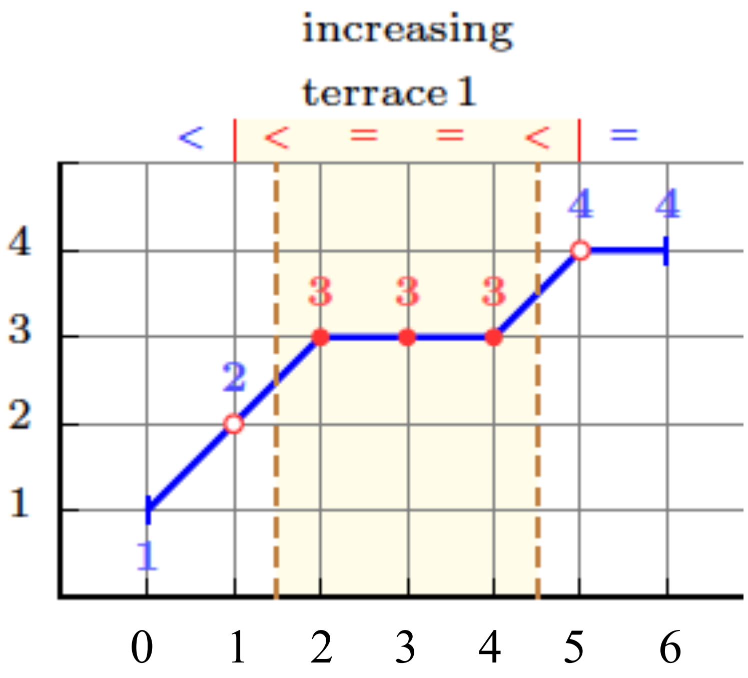
an input sequence $x_0,x_1,\ldots,x_{n-1}$,
its signature sequence $s_0,s_1,\ldots,s_{n-2}$,
a pattern (r,a,b),
a non-empty signature subsequence $s_i,s_{i+1},\ldots,s_j$ forming a maximum word matching r

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



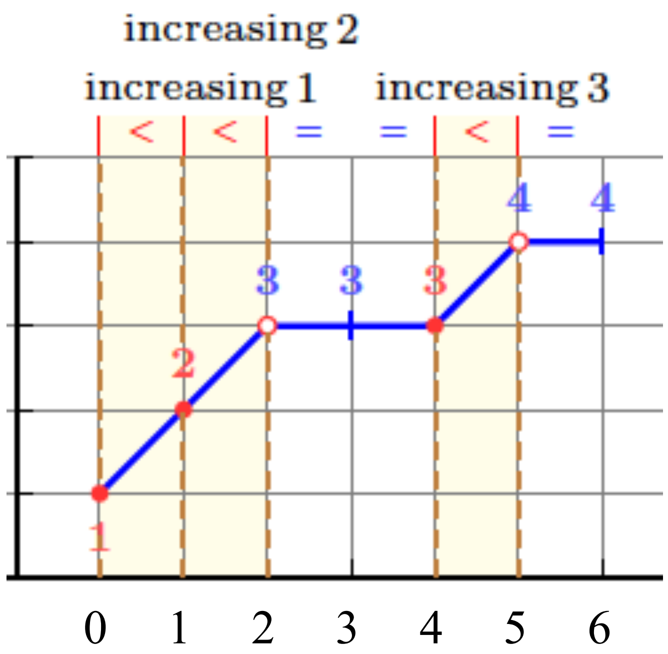
Indices of

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]**

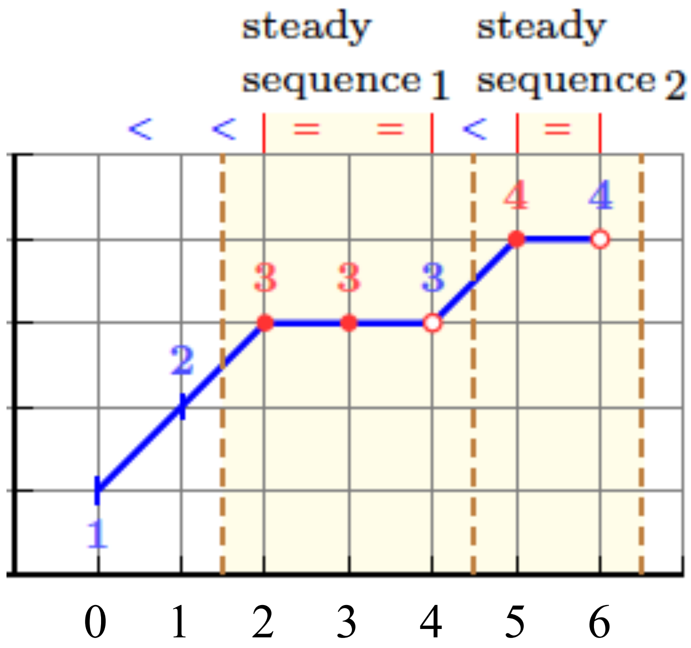
e-occurrences : **[[0..1]]** **[[1..2]]** **[[4..5]]**

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*)

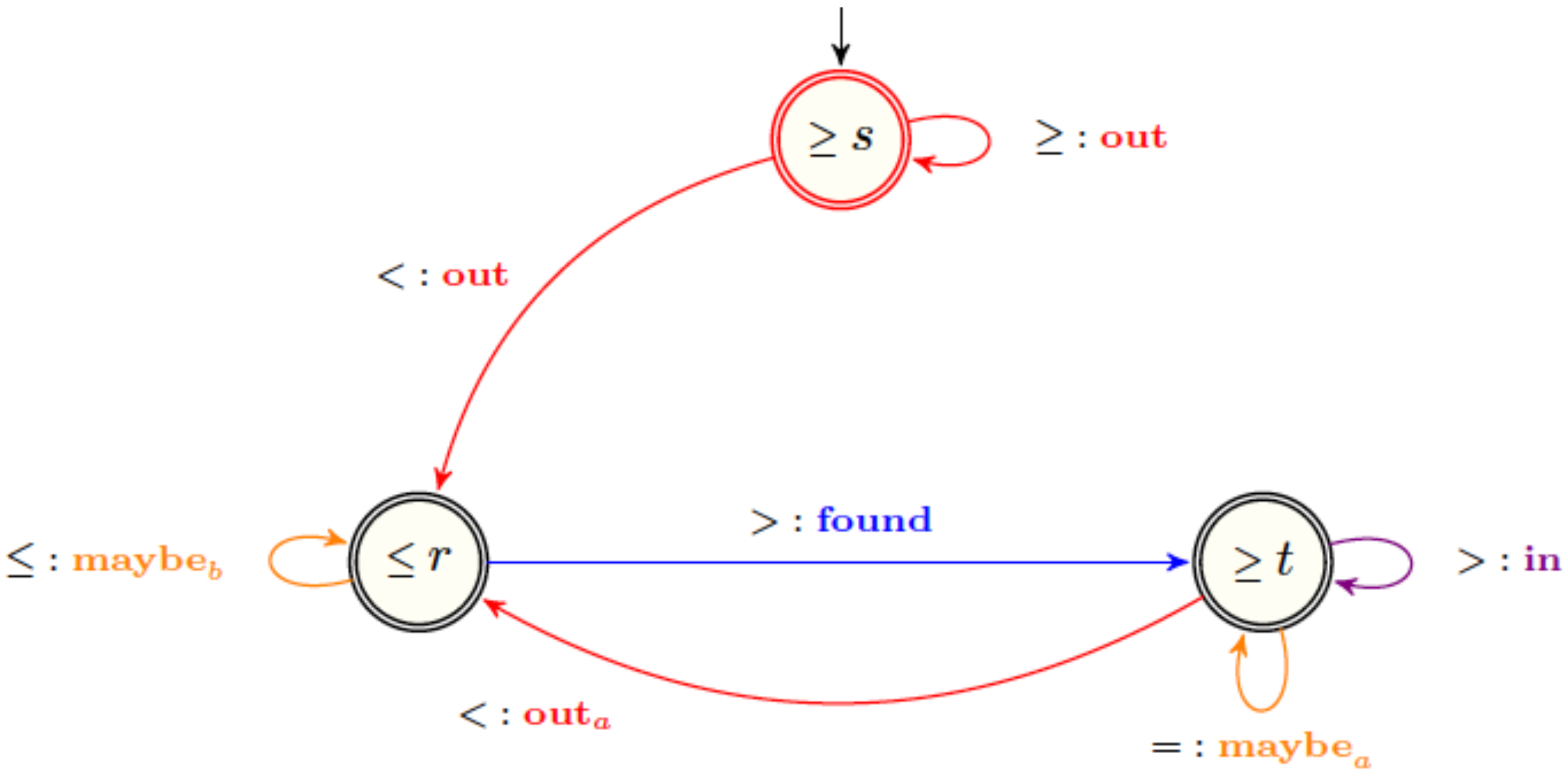
- **Input**: signature sequence

- **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

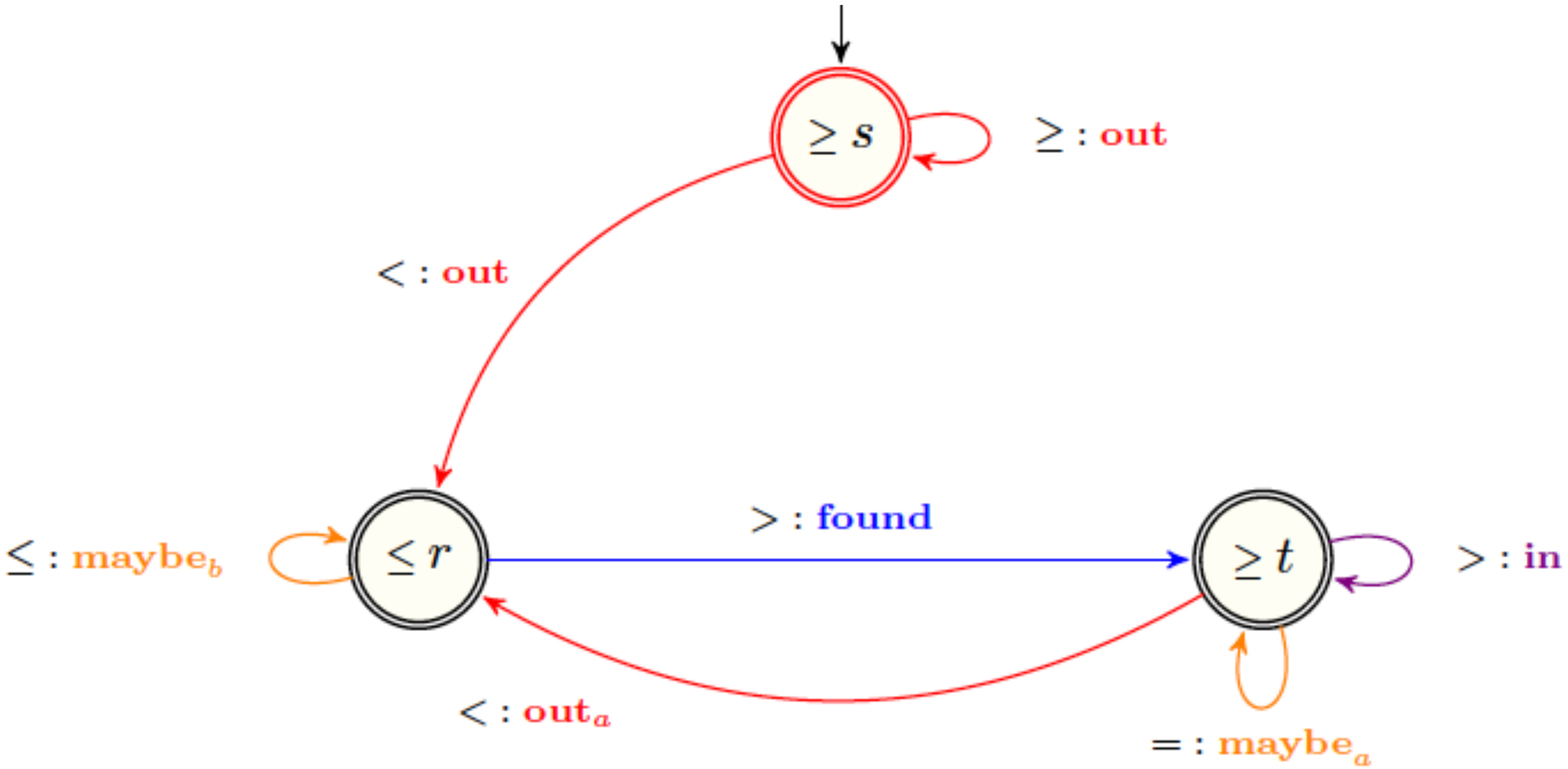
everything *will be synthesized from the seed transducer*

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

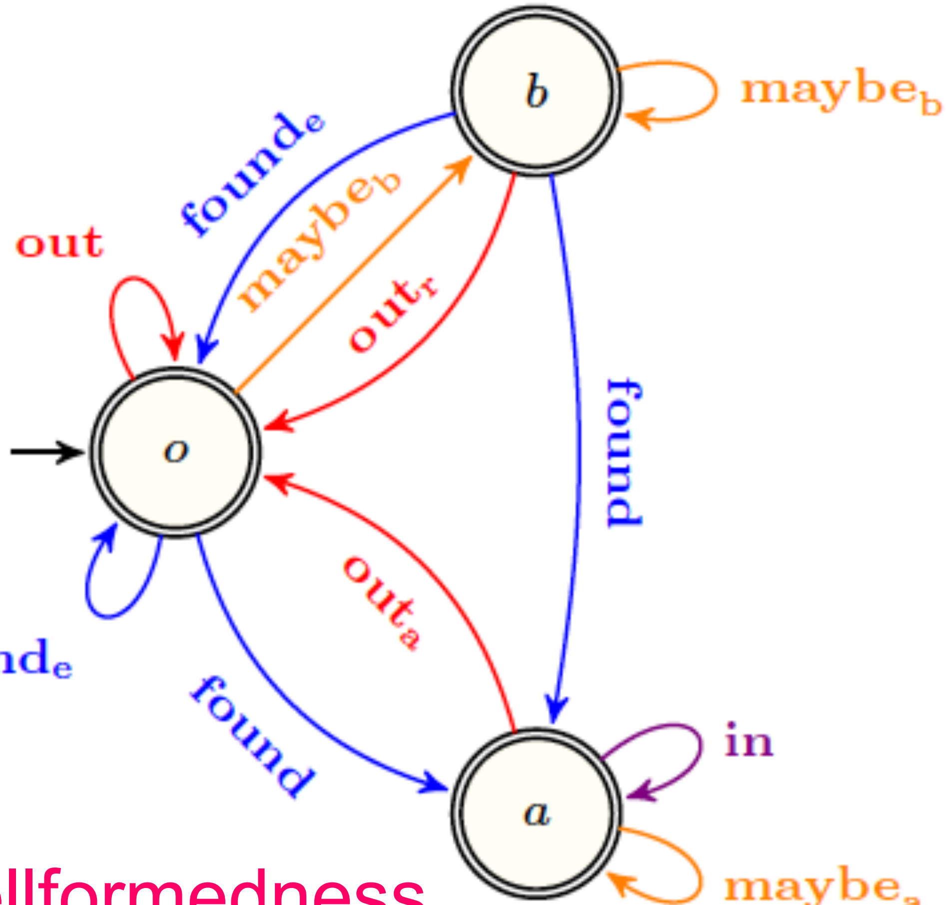


output: semantic string	o	o	o	o	m_b	m_b	m_b	f	in	m_a	o_a	m_b	m_b	m_b	m_b	m_b	f	
input: signature string	=	>	=	<	<	=	<	>	>	=	<	=	=	=	=	=	>	
states:	*s*	*s*	*s*	*s*	*r*	*r*	*r*	*r*	*t*	*t*	*t*	*r*	*r*	*r*	*r*	*r*	*r*	*t*
input: integer sequence	4	4	2	2	3	5	5	6	3	1	1	2	2	2	2	2	2	1

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)

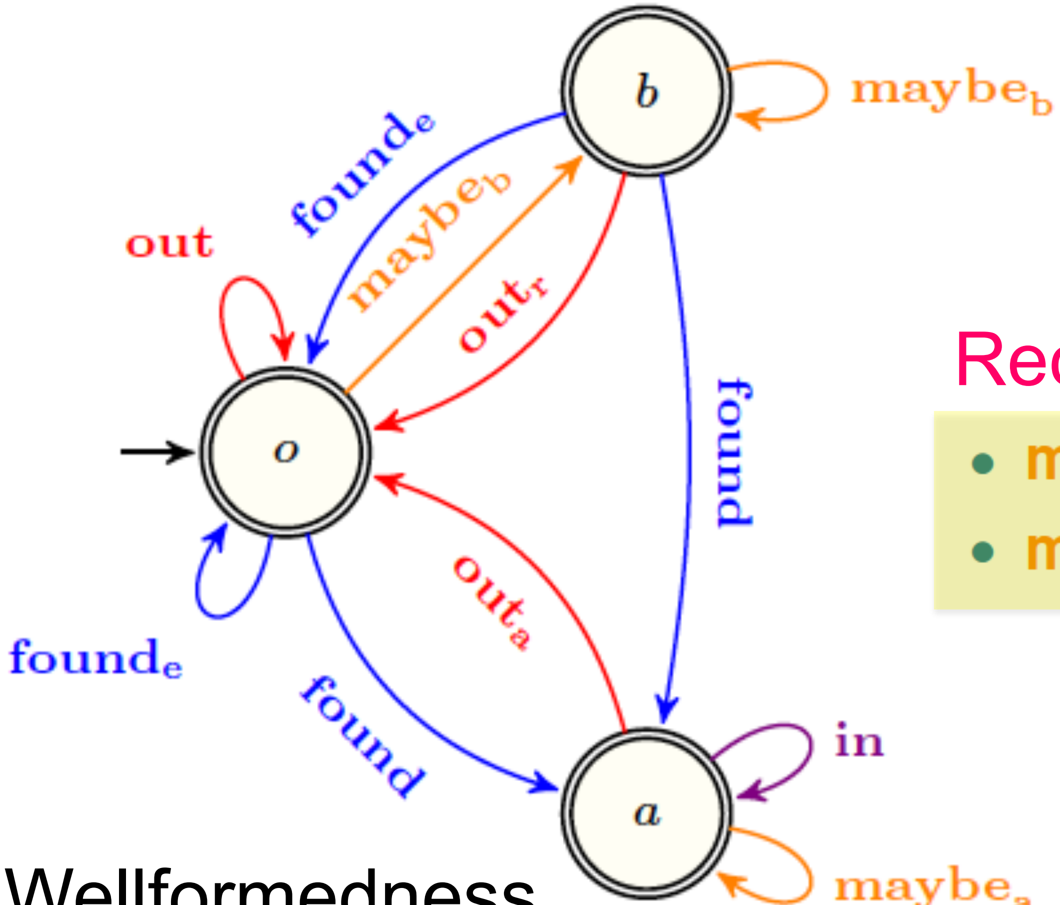


Wellformedness

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)

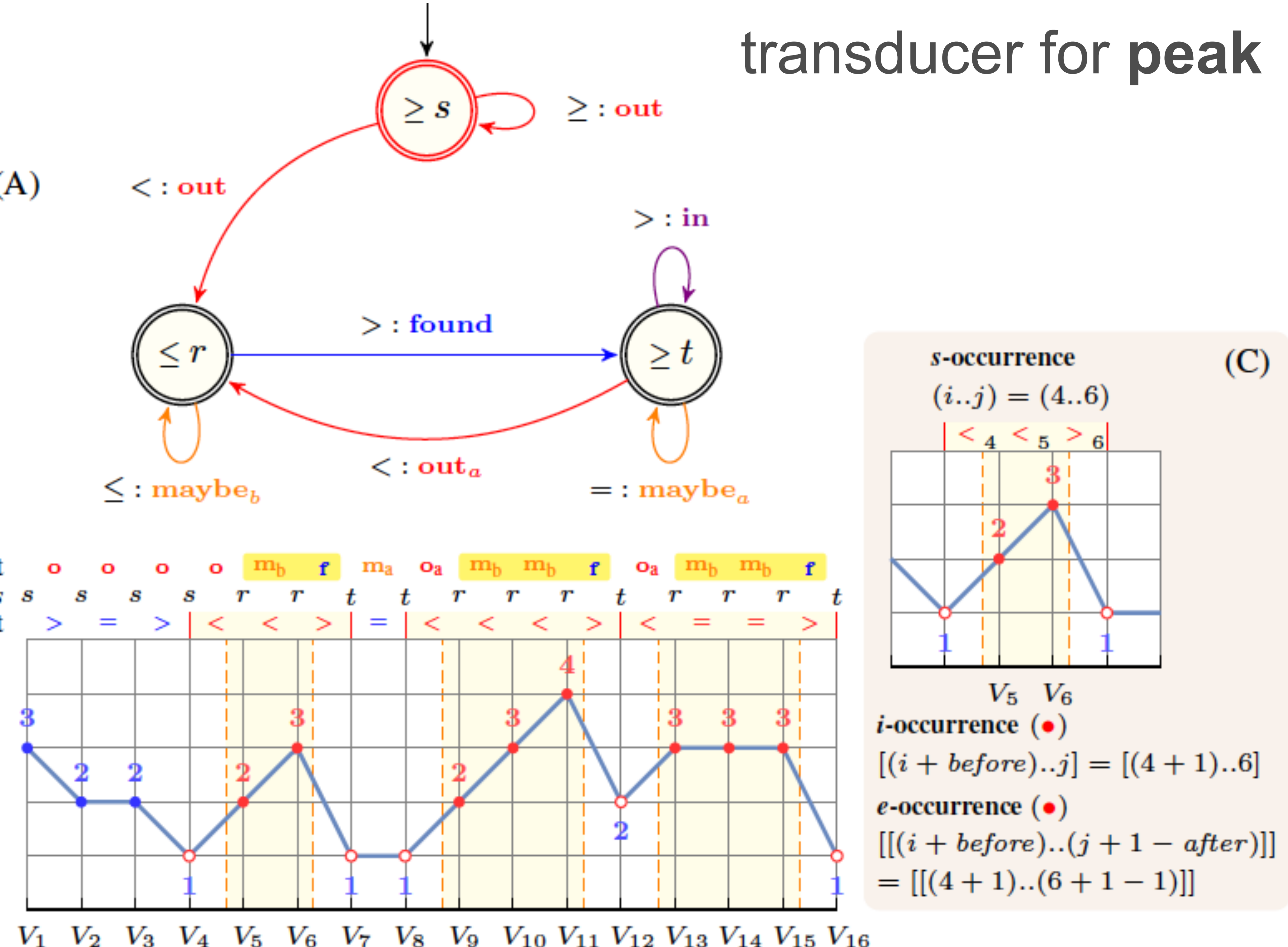


Wellformedness

Recognizing pattern

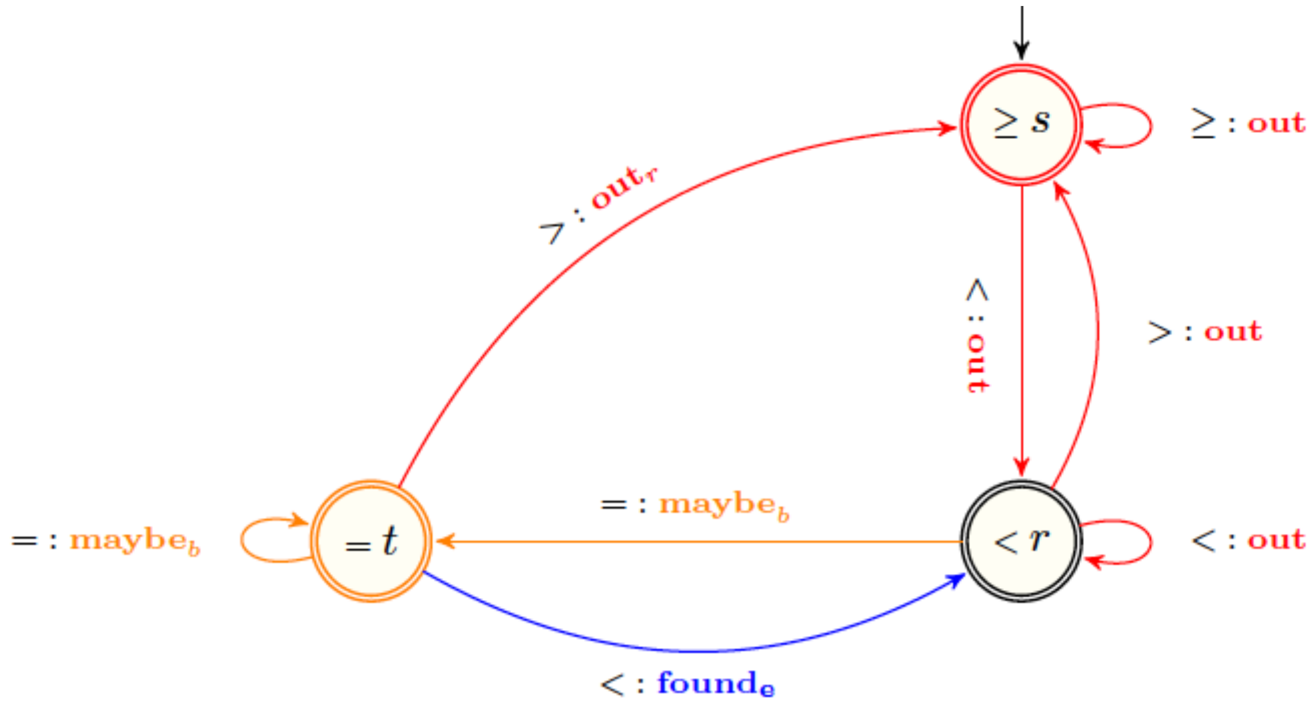
- maybe$_b$* found$_e$
- maybe$_b$* found {maybe$_a$* in$^+$}*

transducer for **peak**



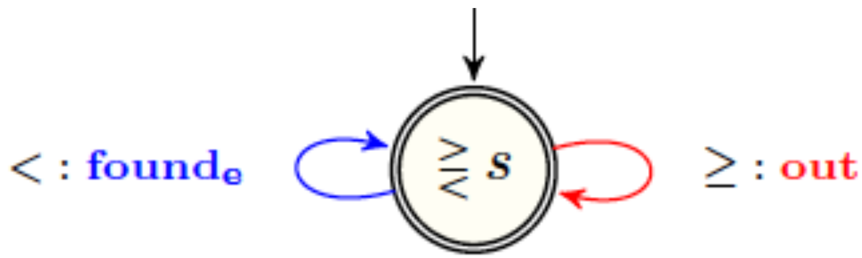
Transducer for **increasing_terrace**

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



Transducer for **increasing**

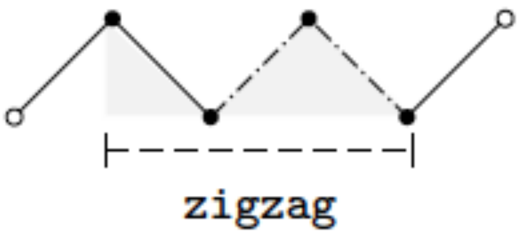
pattern	**regular expression** r	**before** b	**after** a
`increasing`	$<$	0	0



Transducer for **zigzag**

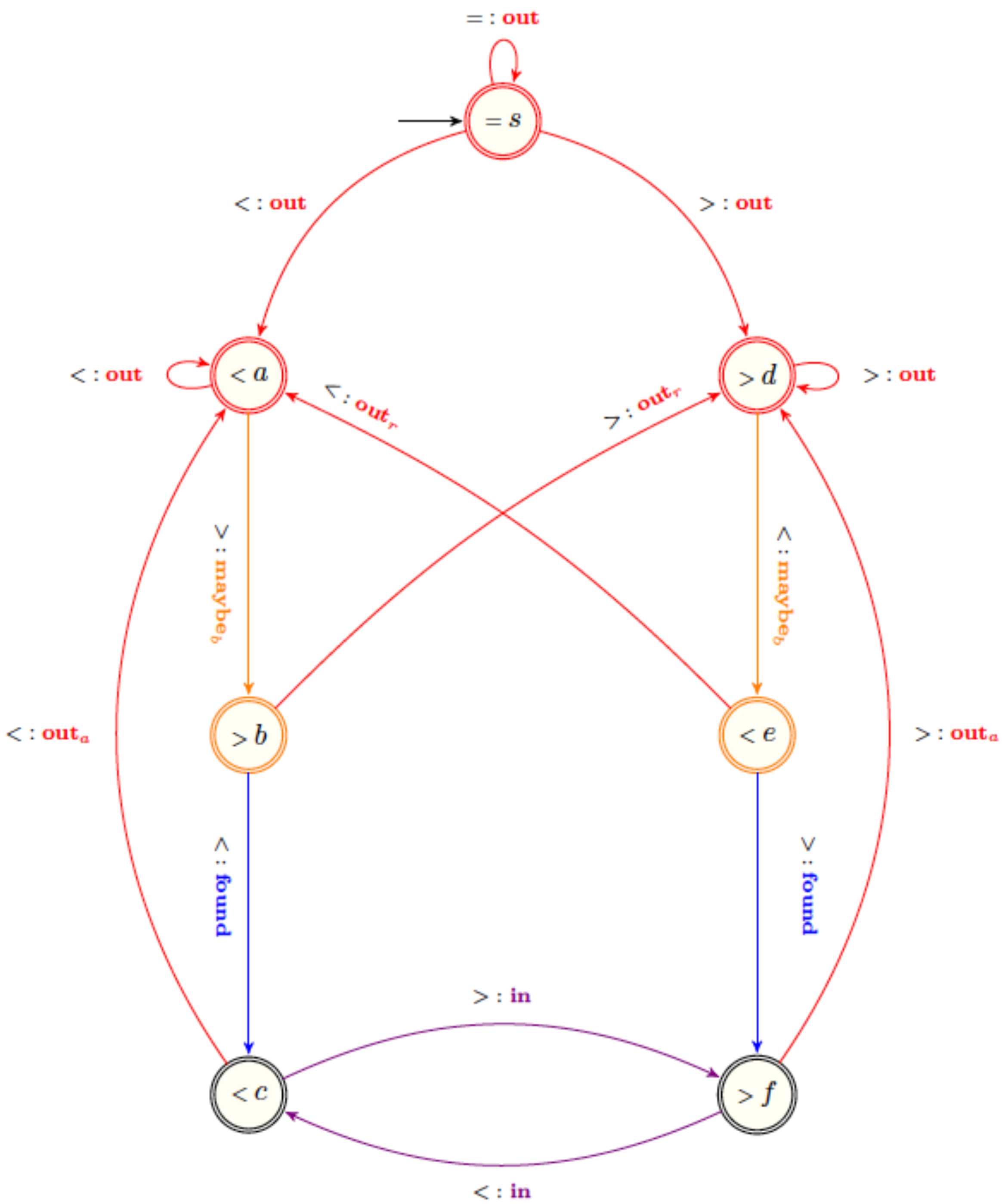
a=1=*b*

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



or

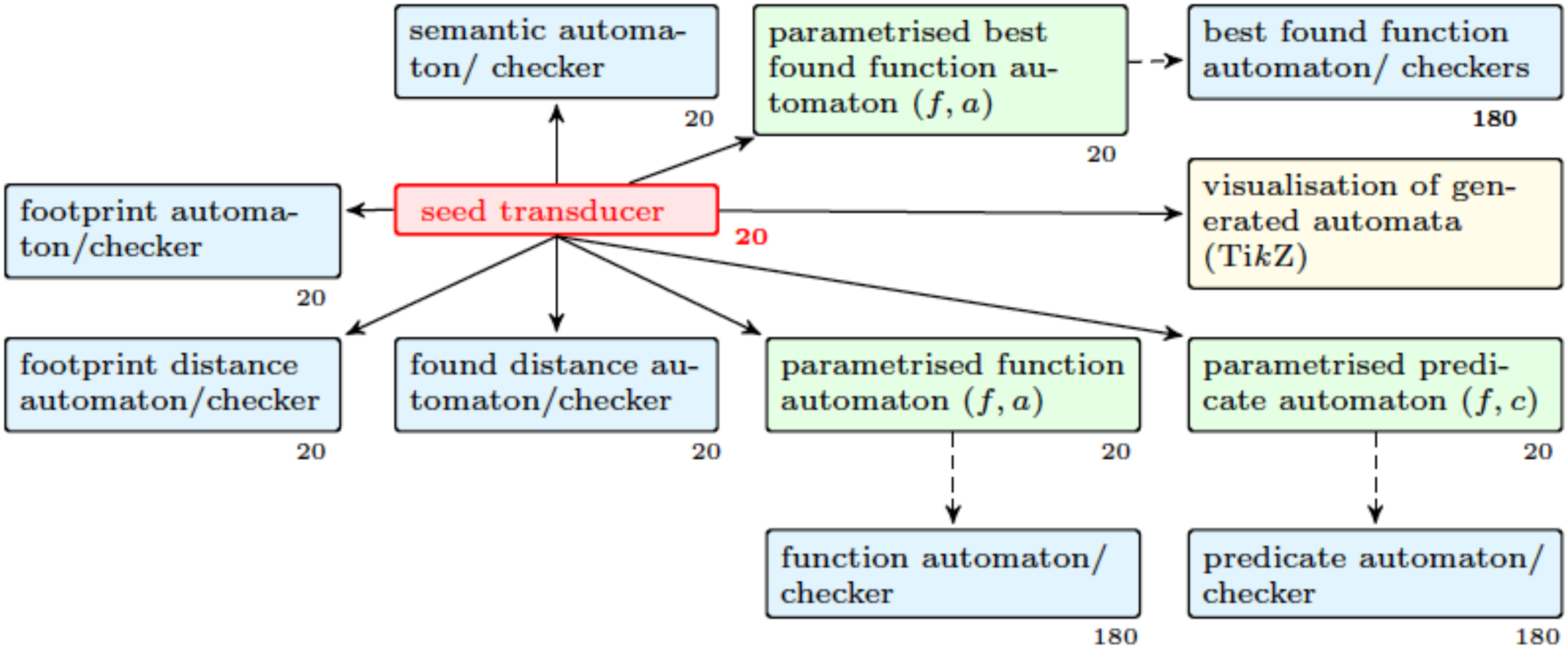
$(><)^+(> \mid ><)$



Outline

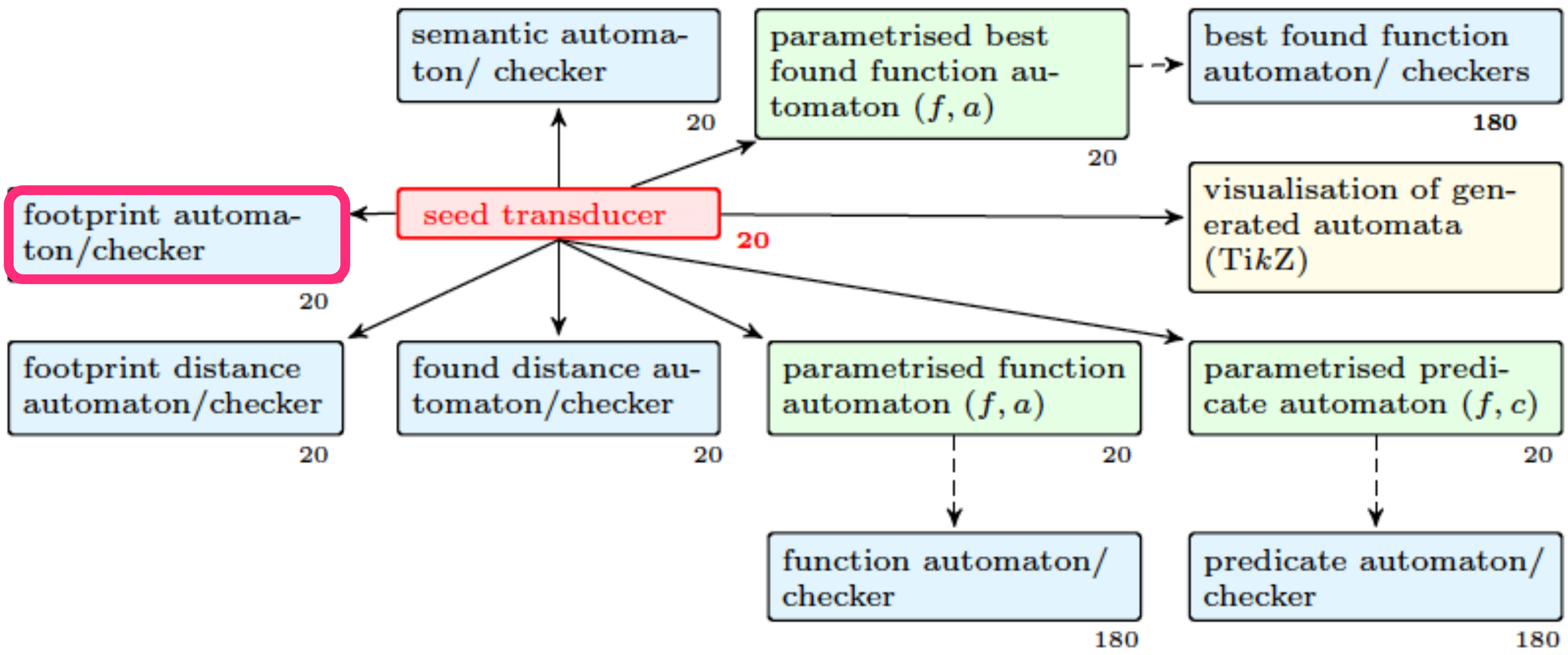
- Background
- Using finite transducers for capturing patterns
- **Types of time-series constraints**

Producing different types of constraints



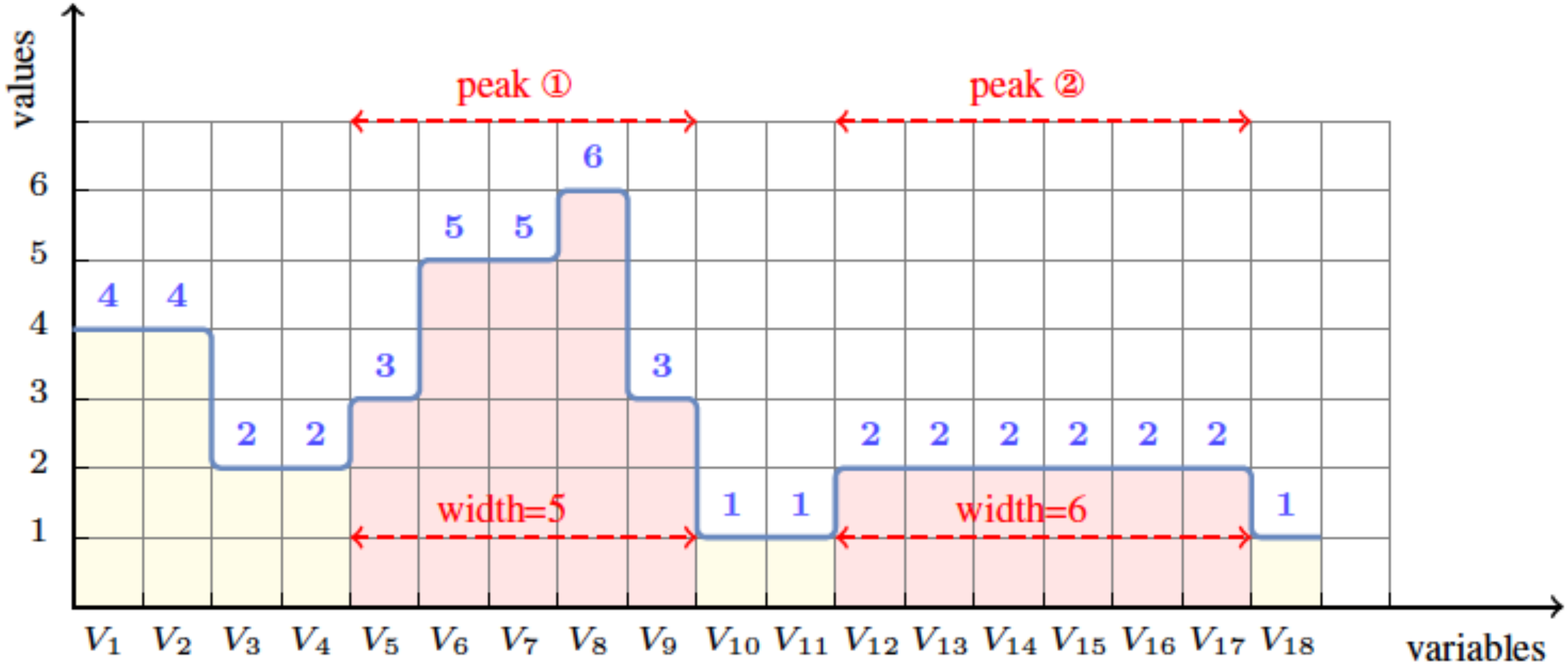
use an appropriate decoration table for synthesizing
the corresponding automaton with counters

Identifying where the occurrences of a pattern are



use an appropriate decoration table for synthesizing the corresponding automaton with counters

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

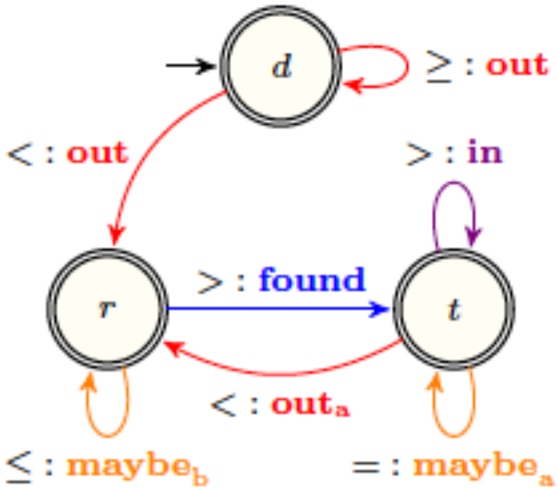


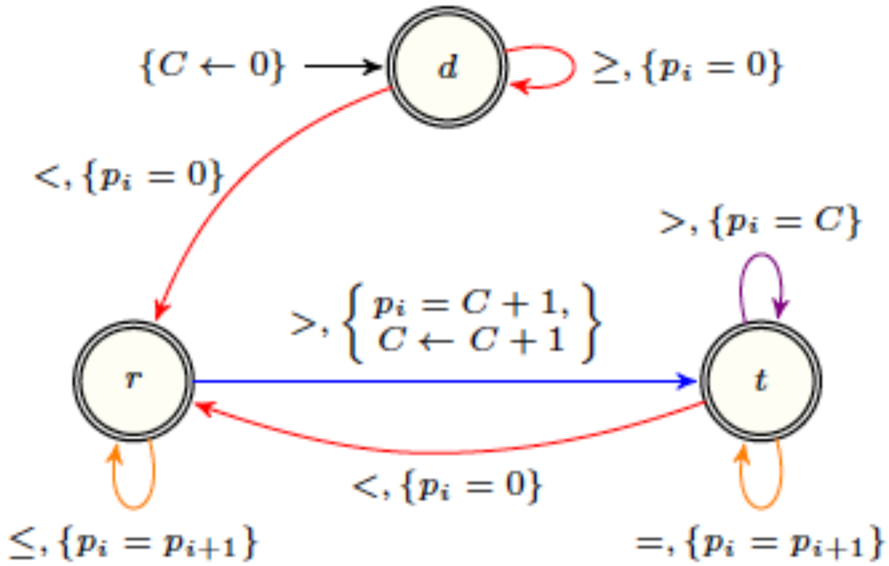
footprint(peak, [4,4,2,2,3,5,5,6,3,1,1,2,2,2,2,2,2,1],
[0,0,0,0,1,1,1,1,1,0,0,2,2,2,2,2,2,0])

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

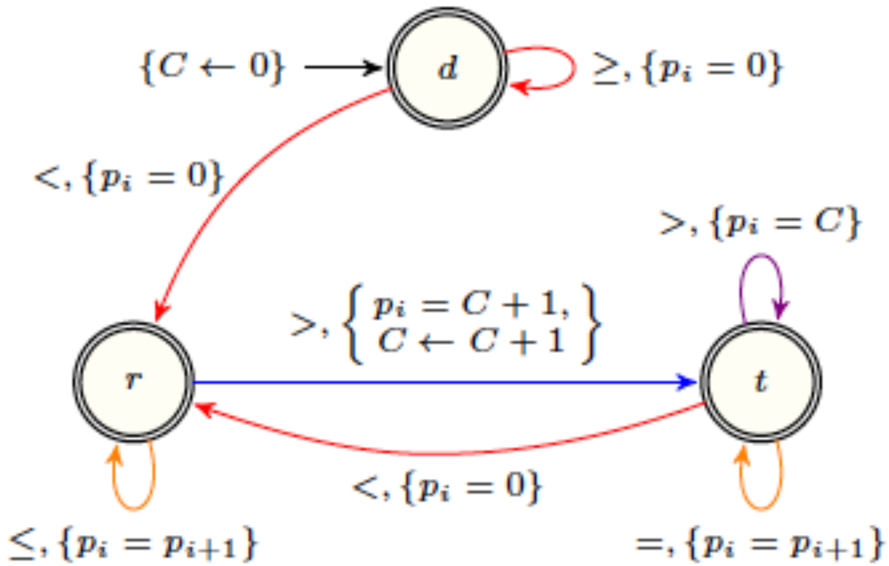
Example: synthesizing the footprint constraint for the **peak** pattern





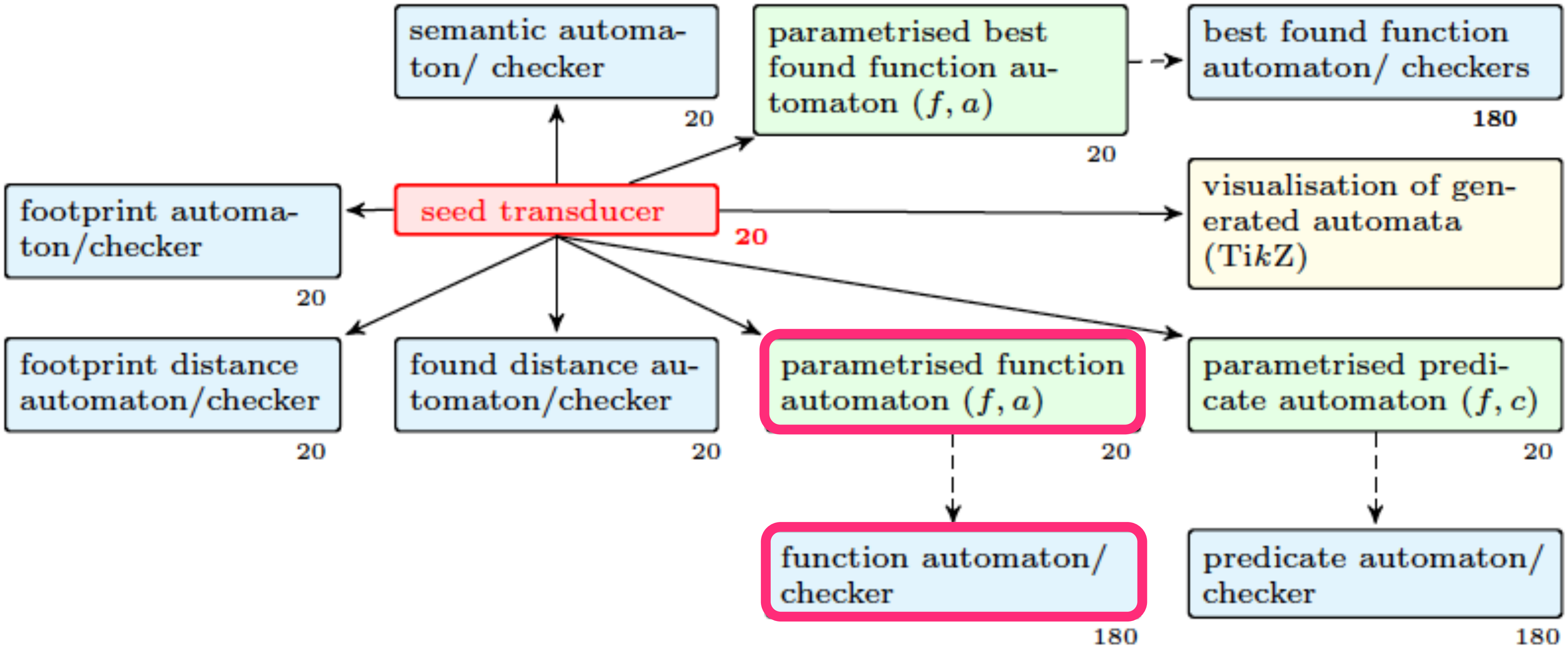
out	$p_i = 0$
out$_r$	$p_i = 0$
out$_a$	$p_i = 0$
maybe$_b$	$p_i = p_{i+1}$
maybe$_a$	$p_i = p_{i+1}$
found$_e$	$p_i = C + 1 \; C \leftarrow C + 1$
found	$p_i = C + 1 \; C \leftarrow C + 1$
in	$p_i = C$

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



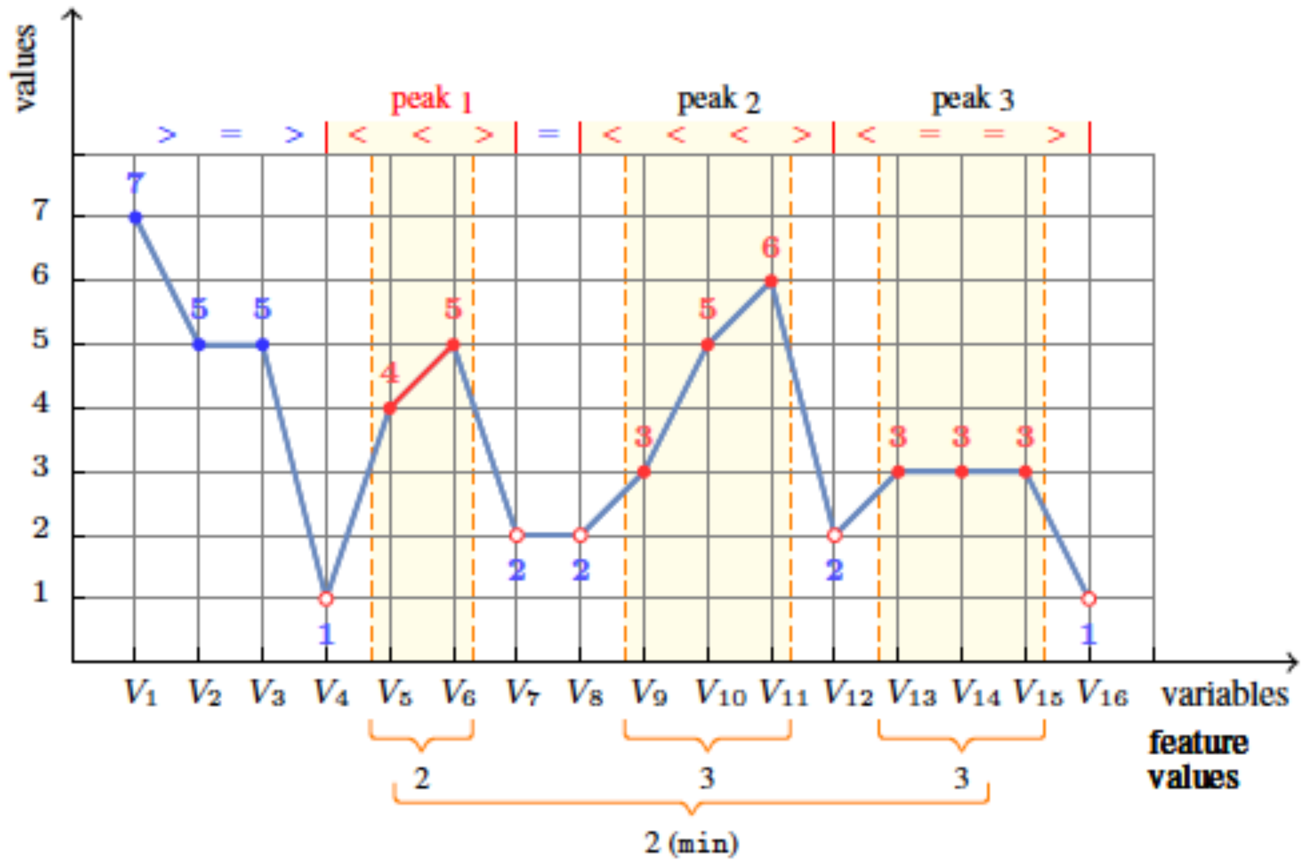
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	$p_0 = 0$	$p_1 = 0$	$p_2 = 0$	$p_3 = 0$	$p_4 = p_5$	$p_5 = p_6$	$p_6 = p_7$	$p_7 = 1$	$p_8 = 1$	$p_9 = p_{10}$	$p_{10} = 0$	$p_{11} = p_{12}$	$p_{12} = p_{13}$	$p_{13} = p_{14}$	$p_{14} = p_{15}$	$p_{15} = p_{16}$	$p_{16} = 2$	$p_{17} = 0$
C_i	${}_0 0$	0	0	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

Restricting feature values of occurrences of a pattern



use an appropriate decoration table for synthesizing the corresponding automaton with counters

Feature constraints (example)



$$\text{MIN\_WIDTH\_PEAK}\,(2, [7, 5, 5, 1, 4, 5, 2, 2, 3, 5, 6, 2, 3, 3, 3, 1])$$

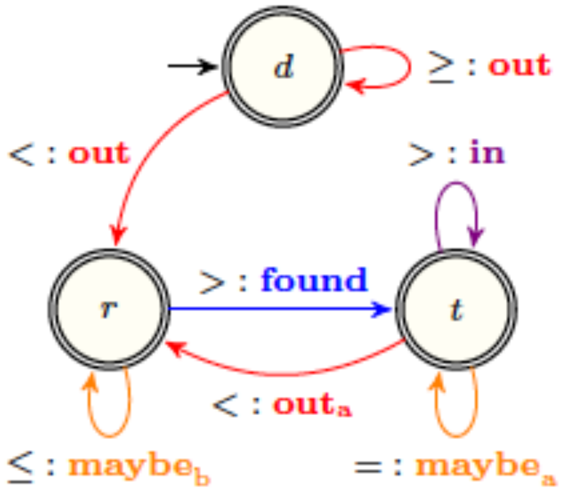
Decoration table for the feature constraint

Feature f	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	$\text{default}_{g,f}$
Max	$\min_f$
Min	$\max_f$
Sum	0

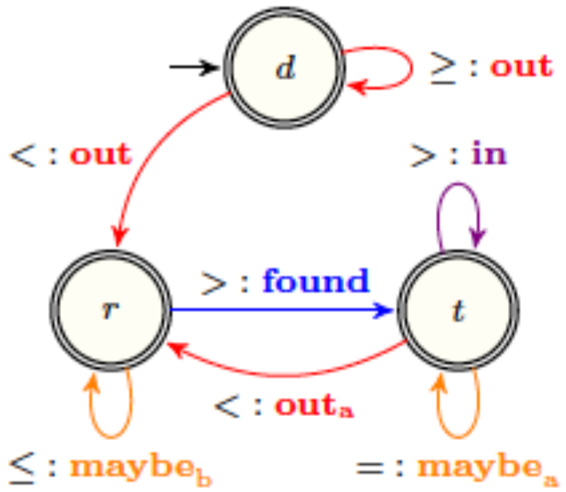
initialization		$C \leftarrow \text{default}_{af}$	$D \leftarrow \text{neutral}_f$	$R \leftarrow \text{default}_{af}$
return		$a(R, C)$		
Semantic Letter		Decoration		
	After	Update of C	Update of D	Update of R
out				
out$_r$			$D \leftarrow \text{neutral}_f$	
out$_a$		$C \leftarrow \text{default}_{af}$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

Example: synthesizing the automaton for min_width_peak

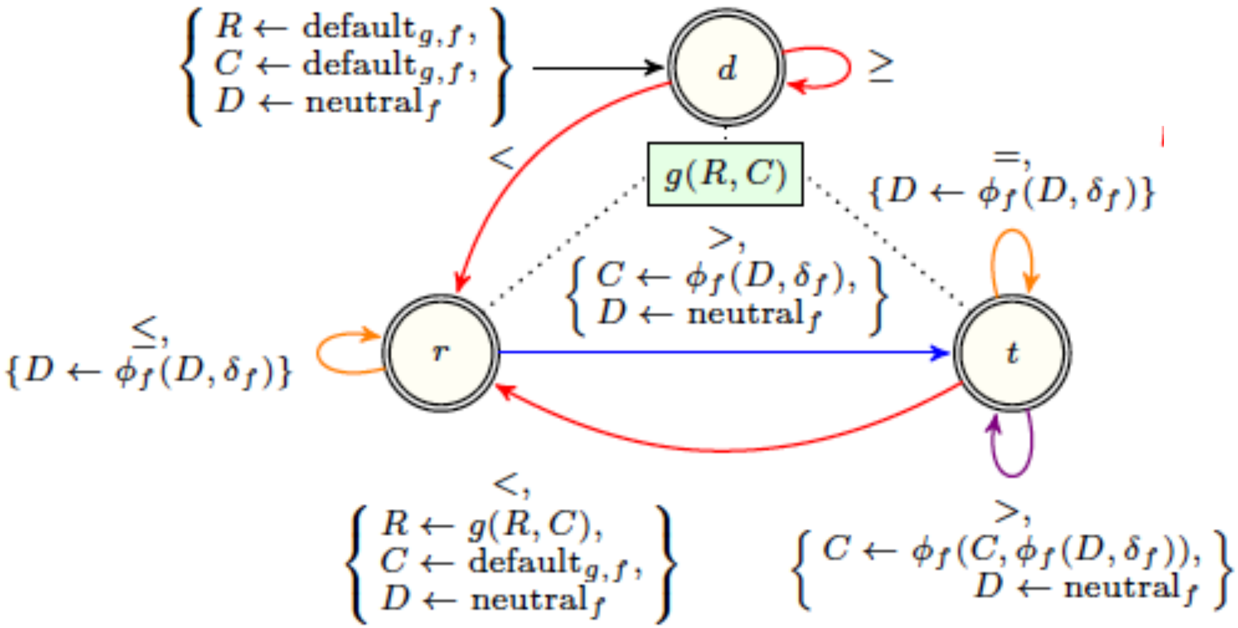


seed transducer

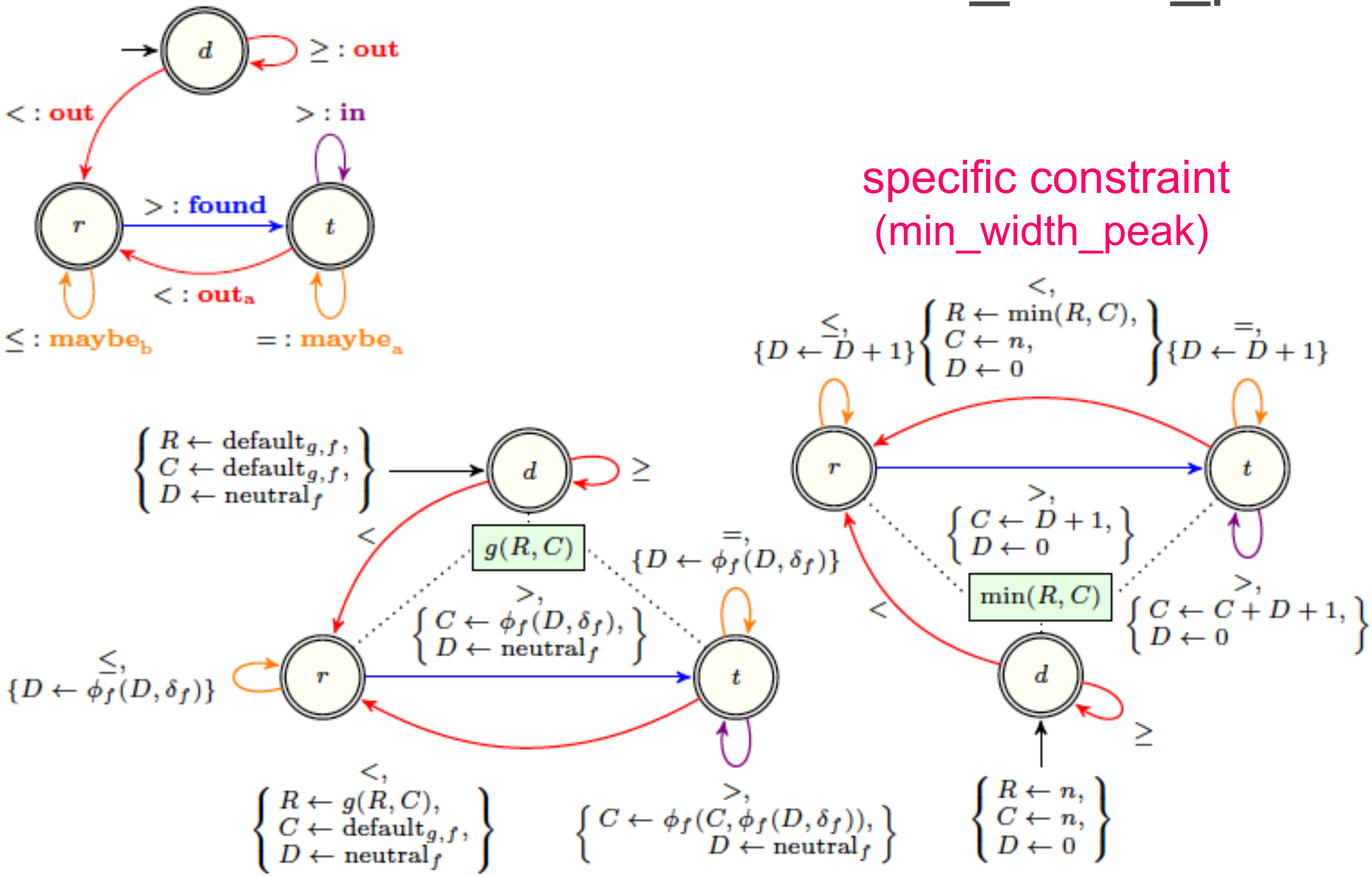
Example: synthesizing the automaton for min_width_peak



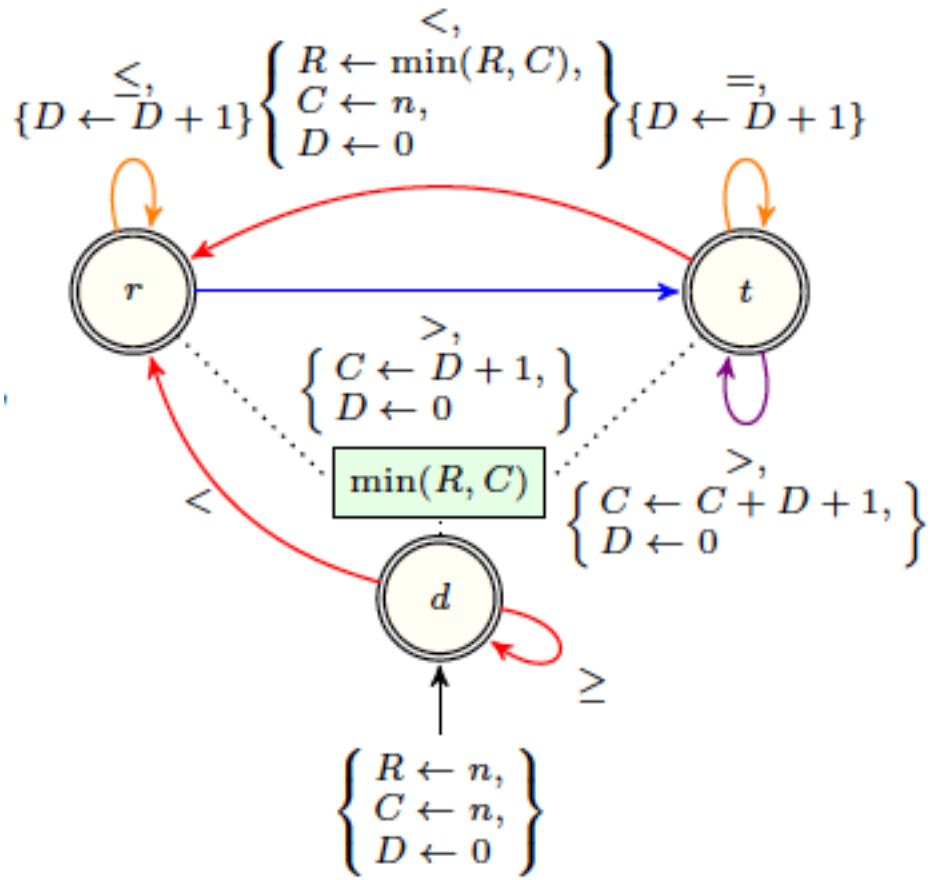
parametrized constraint
(feature f, aggregator g)



Example: synthesizing the automaton for min_width_peak



Running min_width_peak

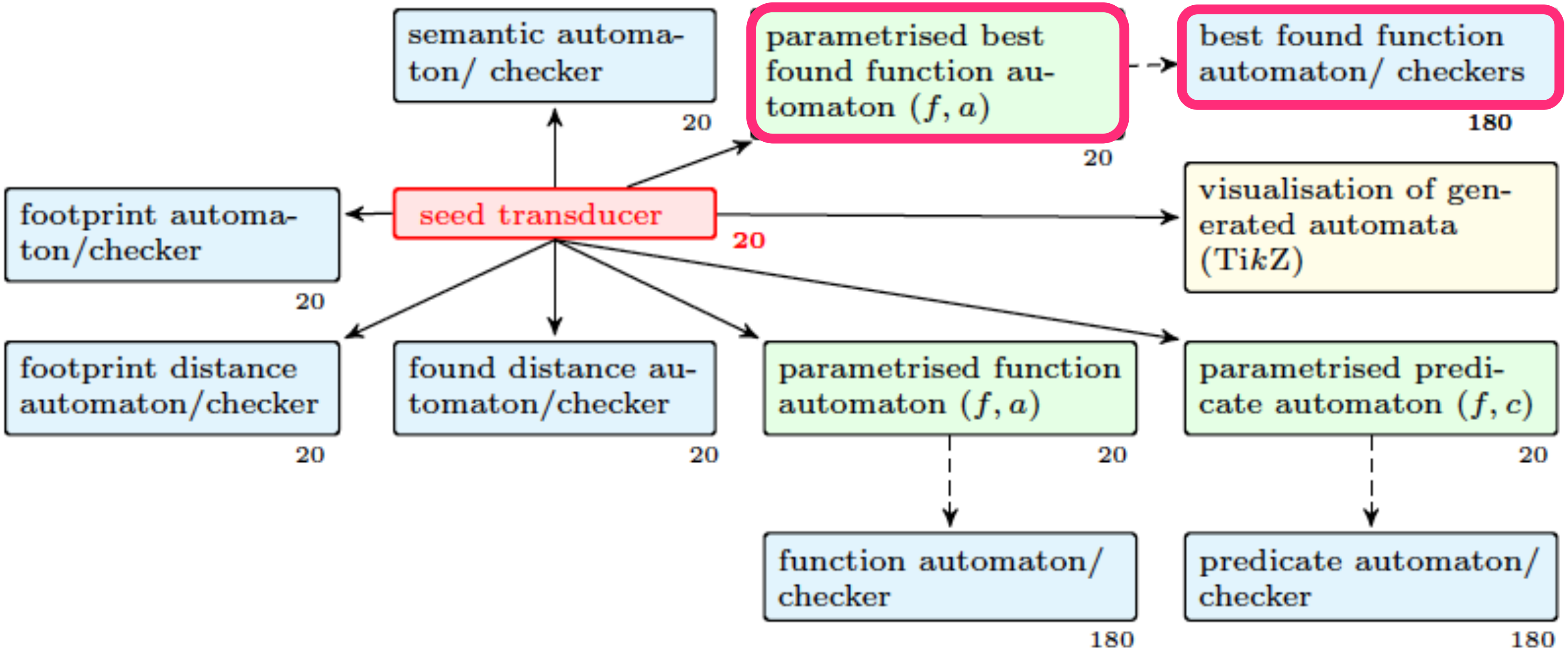


aggregate R	18	18	18	18	18	18	18	18	18	18	18	5	5	5	5	5	5	5	min(5, 6)
current C	18	18	18	18	18	18	18	18	4	5	5	18	18	18	18	18	18	6	
potential D	0	0	0	0	0	1	2	3	0	0	1	0	1	2	3	4	5	0	
semantic action		o	o	o	o	m_b	m_b	m_b	f	in	m_a	o_a	m_b	m_b	m_b	m_b	m_b	f	
signatures		=	>	=	<	<	=	<	>	>	=	<	=	=	=	=	=	>	
states	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	

Generated constraint for min_width_peak

```
min_width_peak_a(Value, VARIABLES):-
        collection(VARIABLES, [dvar]),
        get_attr1(VARIABLES, Xs), length(Xs, N),
        signature(Xs, Signature), gen_pairs(Xs, XPairs),
        Xs = [First|_], LT = 0, EQ = 1, GT = 2,
        automaton(XPairs, Xi-Xj, Signature,
        [source(s),sink(s),sink(r),sink(t)],
        [arc(s,GT,s,([C, D, R])),
        arc(s,EQ,s,([C, D, R])),
        arc(s,LT,r,([C, D, R])),
        arc(r,GT,t,([D + 1, 0, R])),
        arc(r,LT,r,([C, D + 1, R])),
        arc(r,EQ,r,([C, D + 1, R])),
        arc(t,GT,t,([C + D + 1, 0, R])),
        arc(t,EQ,t,([C, D + 1, R])),
        arc(t,LT,r,([N, 0, min(R, C)]))],
        [C,D,R], [N,0,N], [CLast,DLast,RLast]),
        Value #= min(RLast, CLast).
```

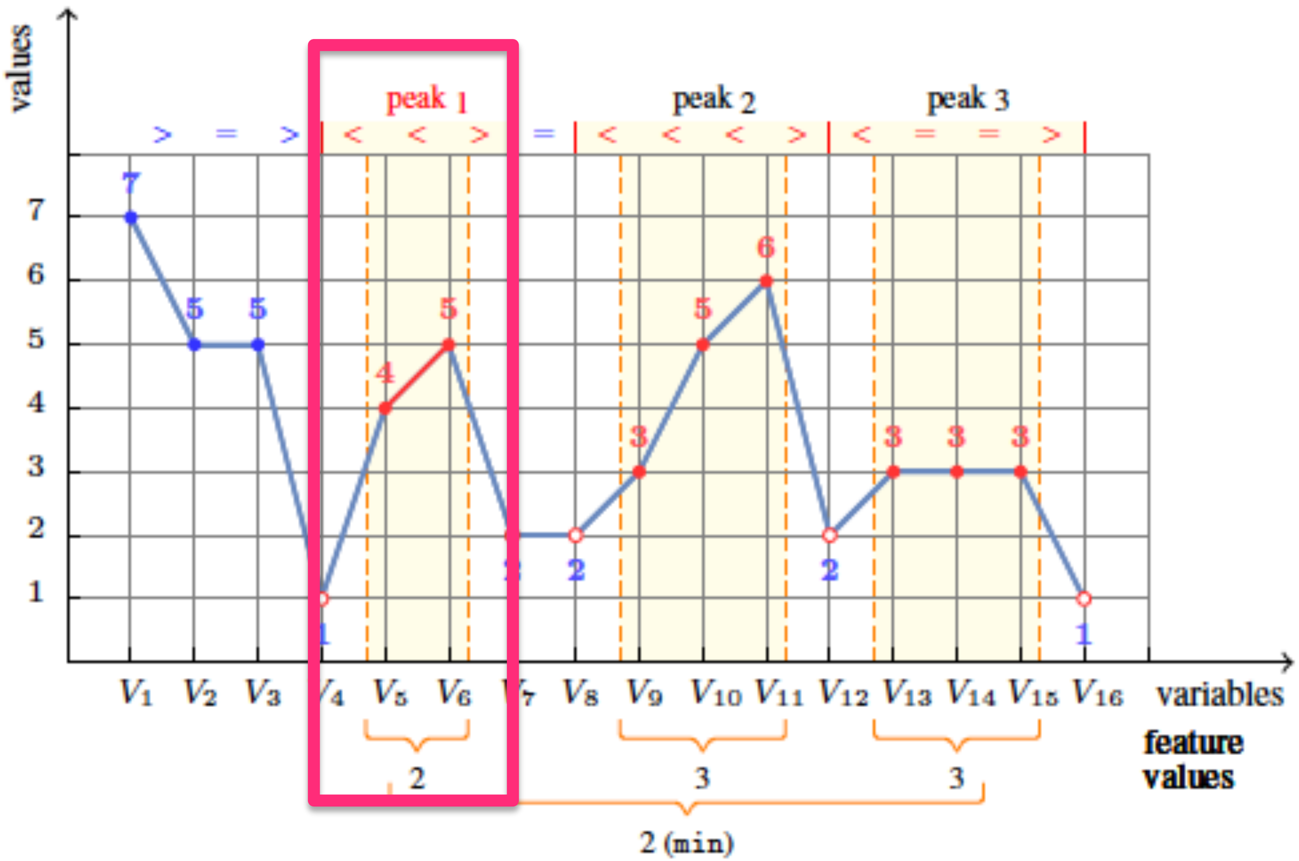
Identifying occurrence of pattern with extreme value



Best found feature constraints

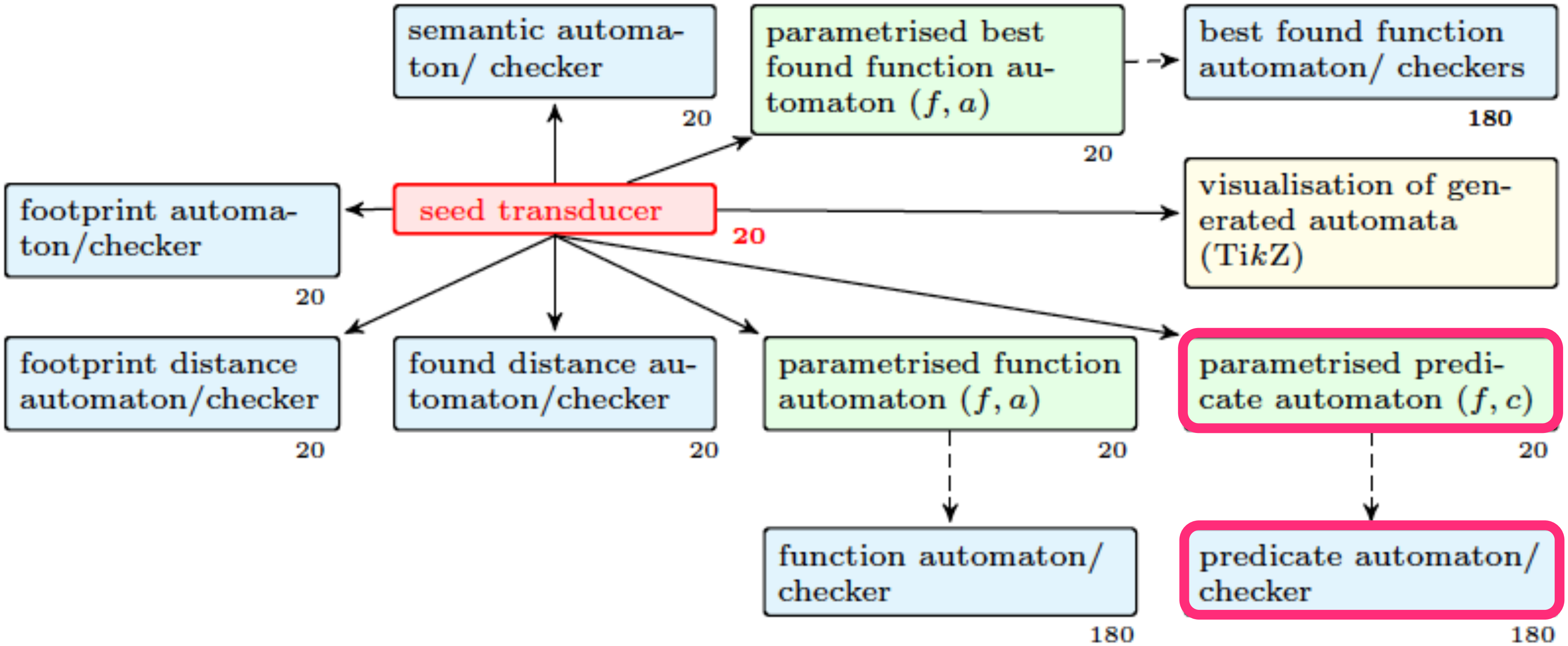
- Used only with min/max aggregators
- Smallest provide footprint of those i-occurrences for which the feature value corresponds to the minimum/maximum value
- Possible use: find most problematic patterns occurrences (e.g., longest zigzag)

Best found feature constraints (example)

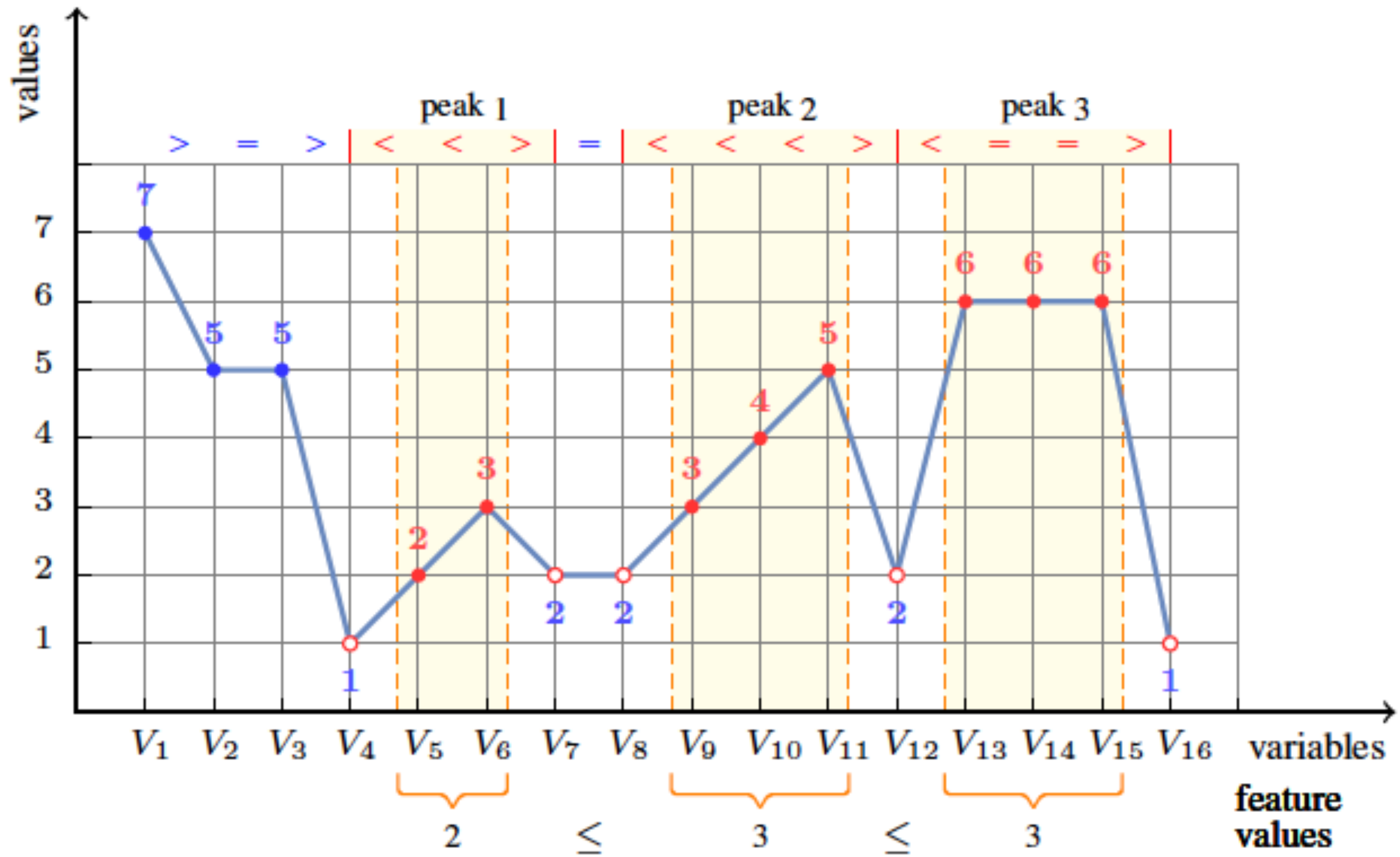


best_found(min_width_peak, 2,
[7,5,5,1,4,5,2,2,3,5,6,2,3,3,3,1],
[0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0])

Predicate constraints

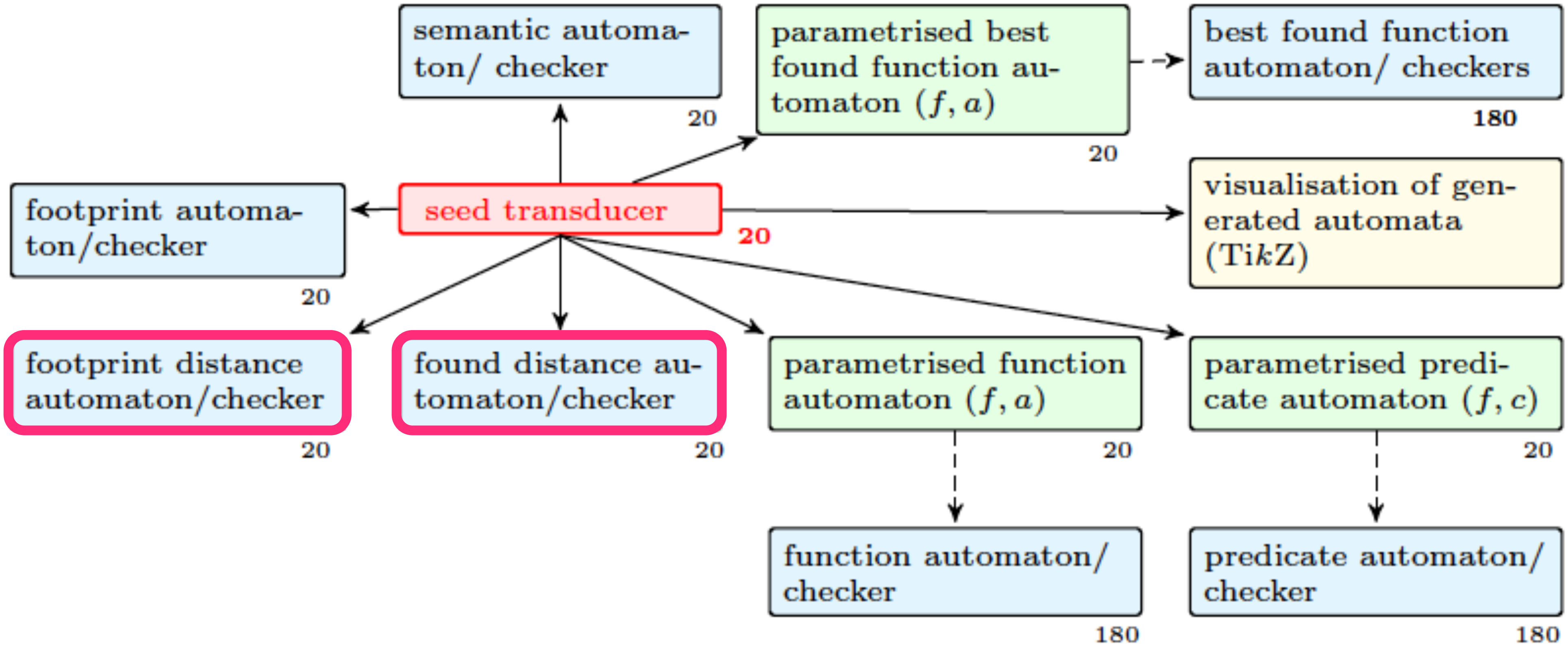


Predicate constraints (example)



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

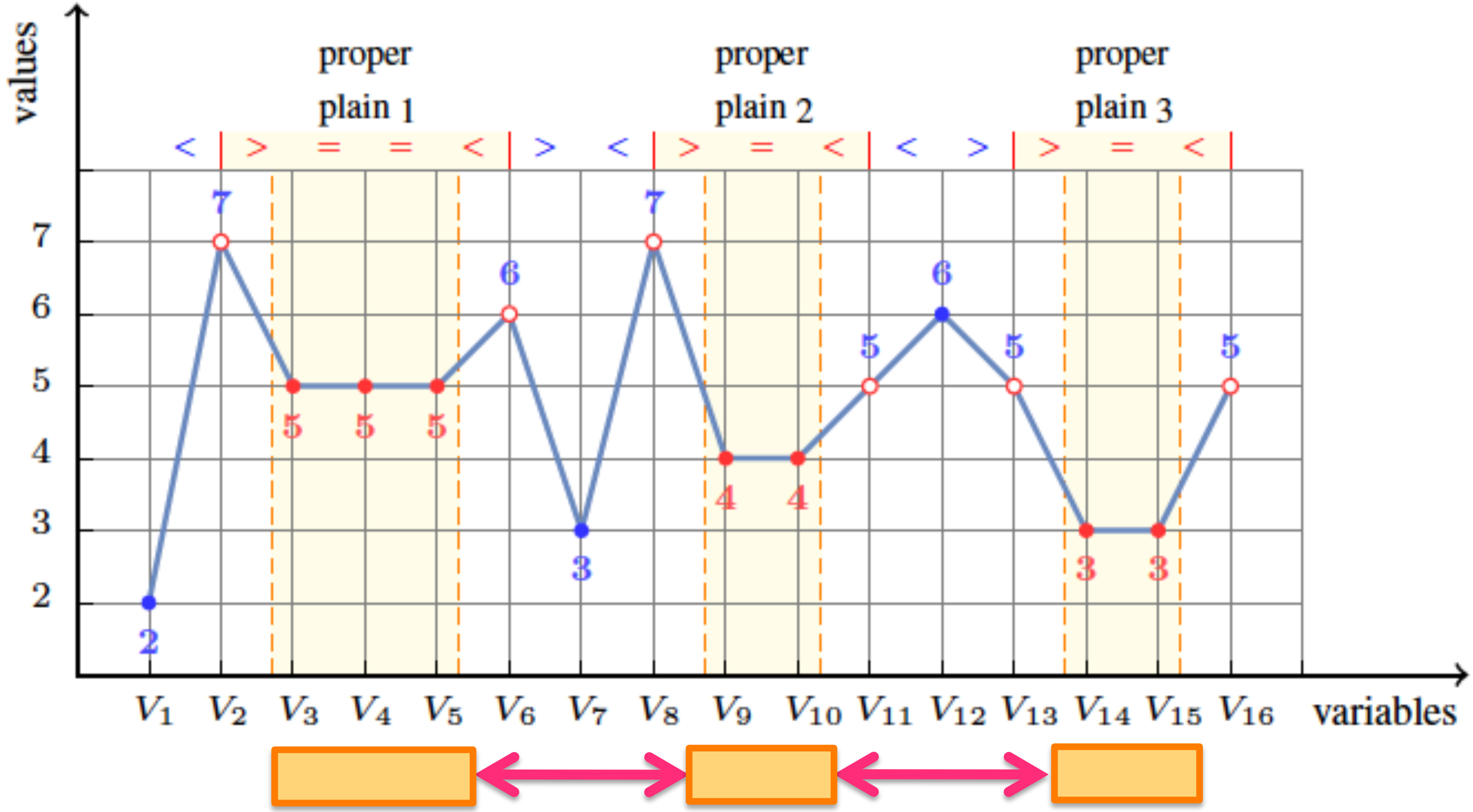
Min/max distance between pattern occurrences



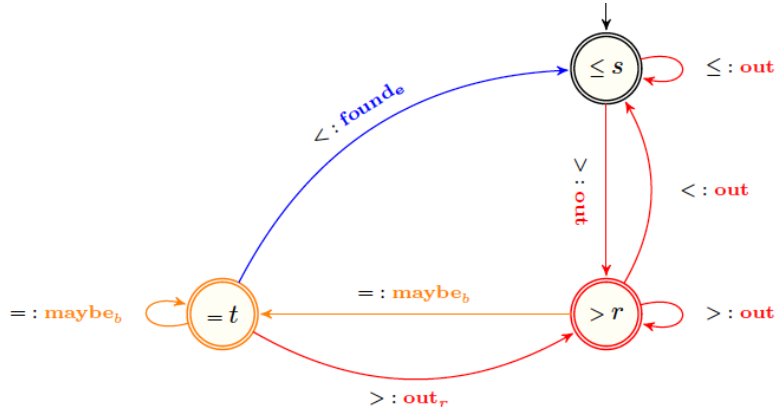
Min/max distance between occurrences of patterns

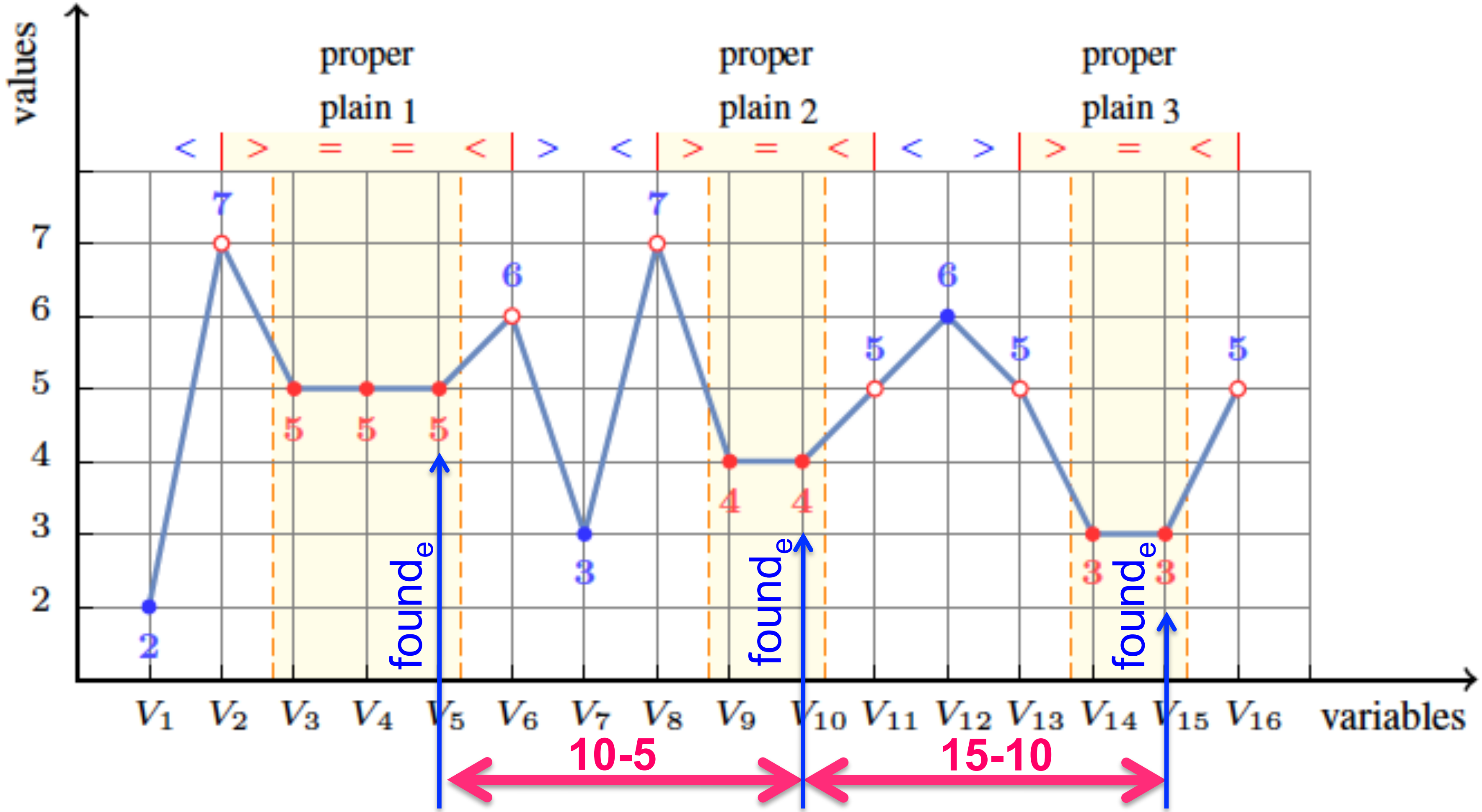
- **Found** distance:
  distance between two consecutive found events
- **Footprint** distance:
  distance between two consecutive *i*-occurrences

Footprint distance example (proper_plain)



Found distance example (proper_plain)





Existing constraints of the catalog

constraint
all_equal_peak
all_equal_valley
decreasing_peak
decreasing_valley
deepest_valley
highest_peak
increasing_peak
increasing_valley
inflexion
longest_decreasing_sequence
longest_increasing_sequence
max_decreasing_slope
max_increasing_slope
min_decreasing_slope
min_dist_between_inflexion
min_increasing_slope
min_surf_peak
min_width_peak
min_width_plateau
min_width_valley
peak
valley

Reformulation of automata constraints in LP

- Two initial papers on automata constraints

  G. Pesant (without counters) CP 2004
  N. Beldiceanu *et al.* (with counters) CP 2004

- Reformulation of regular in LP

  L.M. Rousseau *et al.* (without counters) CPAIOR 2007

Reformulation of times-series constraints (function) in LP

Main result

- Reformulation as LP of all time-series constraints
- Reformulation size (number of variables/number of linear constraints) is in O(n) where n is the sequence size

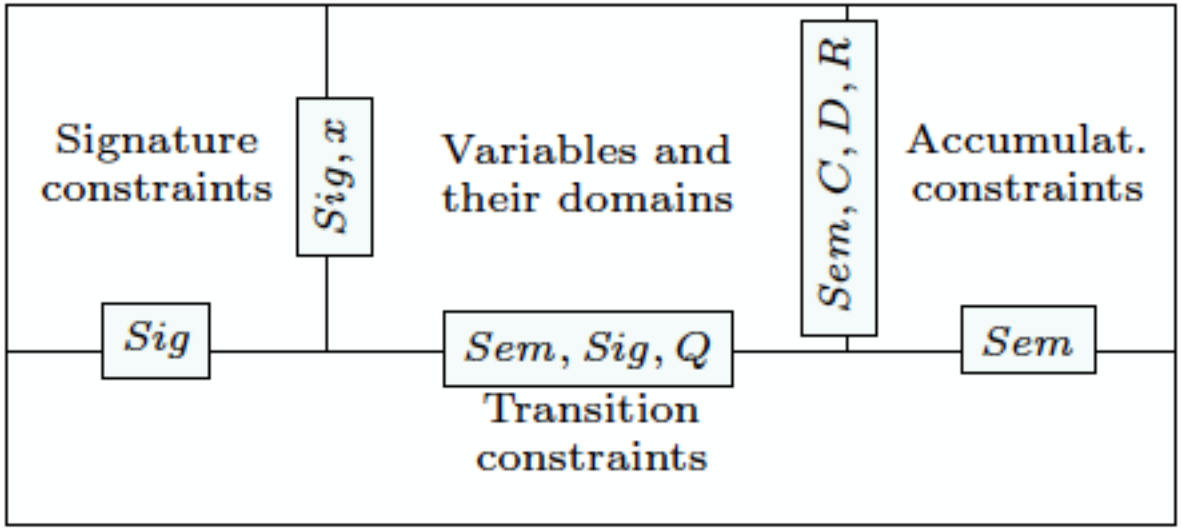
Reformulation of times-series constraints in LP

- Two step process
  - First have a logical model encoding the behaviour of the transducer and of the counter updates
  - Second linearize the logical model using standard modelling techniques

Variables of the logical model

- States variables Q_i (i in [0,n-1])
- Signature variables Sig_i (i in [0,n-2])
- Semantic variables Sem_i (i in [0,n-2])
- Accumulator variables C_i,D_i,R_i (i in [0,n-2])

Constraints of the logical model



example

Accumulator constraints

$$
\begin{aligned}
&Sem_i = \mathbf{found} && \Rightarrow R_{i+1} = R_i + 1,\\
&Sem_i = \sigma' && \Rightarrow R_{i+1} = R_i,\\
&\forall \sigma' \in \{\mathbf{out_a}, \mathbf{out}, \mathbf{in}\},\ \forall i \in [0, n-2].
\end{aligned}
$$

Transition constraints

$$
\begin{aligned}
&Q_i = s \wedge Sig_i = `>' \Rightarrow Q_{i+1} = s \wedge Sem_i = \mathbf{out},\\
&Q_i = s \wedge Sig_i = `=' \Rightarrow Q_{i+1} = s \wedge Sem_i = \mathbf{out},\\
&Q_i = s \wedge Sig_i = `<' \Rightarrow Q_{i+1} = t \wedge Sem_i = \mathbf{found},\\
&Q_i = t \wedge Sig_i = `<' \Rightarrow Q_{i+1} = t \wedge Sem_i = \mathbf{in},\\
&Q_i = t \wedge Sig_i = `>' \Rightarrow Q_{i+1} = s \wedge Sem_i = \mathbf{out_a},\\
&Q_i = t \wedge Sig_i = `=' \Rightarrow Q_{i+1} = s \wedge Sem_i = \mathbf{out_a},\\
&\forall i \in [0, n-2].
\end{aligned}
$$

Use

Context of use
time-series that **partly depends on some underlying structural constraints** that are linked to some infrastructure
(*e.g. temperature measurement in a building,*
energy production by some unit)

Building models

Learning models from structured time series data,
where model is a conjunction of time-series constraints.

➔ *Compute features and select the most relevant ones*
not just classifying things,
can be used as a constraint model to generate new solutions.

Identifying automatically problems in data

➔**Locate extreme feature values** for some pattern
(e.g. zizag, bump_on_decreasing_sequence,
dip_on_increasing_sequence)

➔**Repair identified problems**
(by solving a small constraint problem)

Conclusion

1. Everything synthesised from the seed transducer (*decoration tables independent from the transducer*)

2. Some new work (*oriented toward practice*) regarding automata constraints (*not just picking the next item in the language hierarchy*)

3. On going work to extend, reinforce things along different lines

4. Automata for which the generator is very compact/versus automata for which you need to write ad-hoc code to generate them

Conclusion

- Second volume of the global constraint catalog
devoted to time-series constraints
  - use exactly the **same format** as the current catalog,
  - with the difference that **everything is synthesized** (*text, figures, code*).

Global Constraint Catalog

Volume II

Time-Series Constraints

Nicolas Beldiceanu[1]
TASC (CNRS/INRIA) Mines Nantes, FR-44307 Nantes, France

Mats Carlsson
SICS, Box 1263, SE-16 429 Kista, Sweden

Helmut Simonis
Insight Centre for Data Analytics, University College Cork, Ireland

Abstract: This report first presents a restricted set of finite transducers used to synthesise structural time-series constraints described by means of a multi-layered functions composition scheme. Second it provides the corresponding synthesised catalogue of structural time-series constraints where each constraint is explicitly described in terms of automata with accumulators.