Centre for
Data Analytics

Insight

A Transducer-Based Model for Representing Functional Constraints on Integer Sequences

Ekaterina Arafailova, Nicolas Beldiceanu, Mats Carlsson, Remi Douence, Maria Andreina Francisco Rodriguez, and **Helmut Simonis**

ModRef 18, August 26, 2018

DCU NUI Galway OÉ Gaillimh UCC UCD sfi

Key Ideas

- Describe mechanism to define many global constraints declaratively
- Seen as functional constraints on a sequence $c(R, [X_1, X_2, ..., X_n])$
- Automatically generate constraints from description
- Based on automata with registers
- Provides basis for much more (not covered here):
    - Bounds
    - Redundant constraints
    - Interaction of constraints

Examples

- `nb_strictly_decreasing_sequence`$(2, [1, 1, 0, 0, 1, 0, 0, 1])$
- `max_width_group`$(2, [0, 1, 0, 1, 1])$
- `nb_stretch`$(6, [0, 1, 1, 1, 0, 1, 0, 1])$

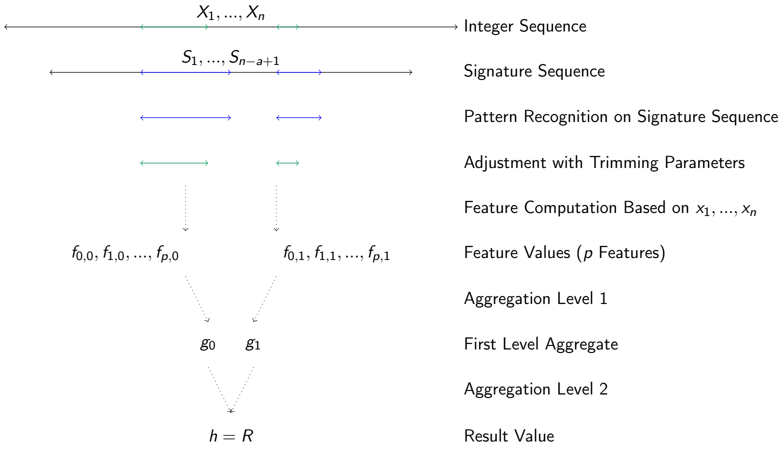
Why?

- Constraint solvers depend on global constraints for performance
- But each constraint is an algorithm on its own
- Too many to implement
- Impossible to be confident in correctness
- Too many to remember
- We need a more systematic way to describe and implement global constraints

Note

- We describe the process based on integer sequences
- The derived mechanism also works as a constraint on finite domain variables

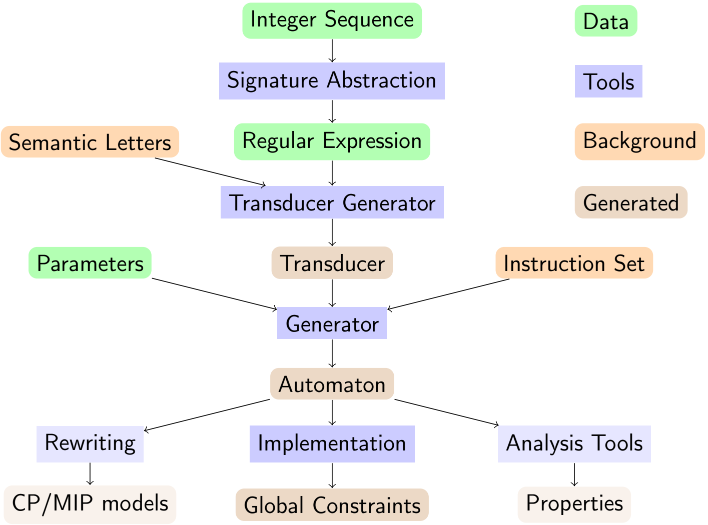
Computing with Integer Sequences



Signature Definition

- Time-Series: binary $<, =, >$
- Unary example: $\in, \notin$ for value set V, max_width_group
- Other binary example : $=, \neq$, nb_stretch
- k-ary: Compute signature value from sub-sequences of length k
- Unary classification
- Extended binary $\ll, <, =, >, \gg$

The Scheme

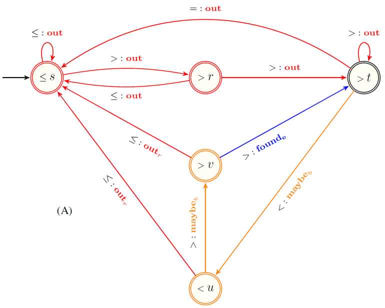

Integer Sequence
Signature Abstraction
Semantic Letters
Regular Expression
Transducer Generator
Parameters
Transducer
Instruction Set
Generator
Automaton
Rewriting
Implementation
Analysis Tools
CP/MIP models
Global Constraints
Properties
Data
Tools
Background
Generated

But you did all this already for time-series?

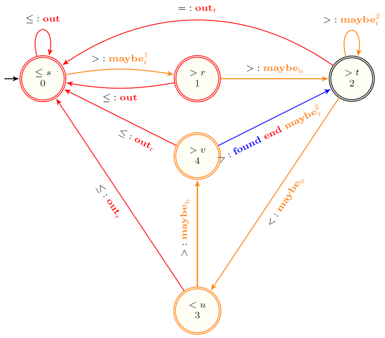
- This is a second-generation study
- Time-series were a special case
  - Specific, binary signature ($<$,$=$,$>$)
  - Initially hand-crafted transducers
  - Feature/Aggregator set limited
  - Only one feature considered per constraint
- Some issues were not dealt with elegantly
  - Mix-up of pattern recognized and feature value computed
  - Some pattern were not expressible

Compare: Old vs New

bump_on_decreasing_sequence, Reg. expr $'>><>>'$, Signature $'<=>'$



Old



New

Example Recognition $' >><>>'$

$>$	$>$	$>$	$<$	$>$	$>$	
out	out	out	m_b	m_b	f_e	Old (Only for b=2)
m_r^1	m_b	m_r^2	m_b	m_b	f, e, m_r^2	New

The Generated Automaton

- Based on the generated transducer
- Five registers
  - Level 4: aggregated value h
  - Level 3: aggregated value g
  - Level 2: feature value of current confirmed, but uncompleted pattern
  - Level 1: potential feature value of unconfirmed pattern
  - Level 0: potential feature value of unconfirmed pattern

The Micro-Instructions

micro instruction	register updates
$\texttt{compute}(\ell, b, v)$	: **if** $b = 0$ **then** $V_\ell \leftarrow \phi_\ell(V_\ell, v)$ **else** $V_\ell \leftarrow \phi_\ell(V_\ell, -v)$
$\texttt{reset}(\ell)$	: **for** $k \in [0, \ell]$ **do** $V_k \leftarrow \mathrm{id}_k$
$\texttt{transmit}(c, b, \ell)$	: **if** $c = 1$ **then** $V_{\ell+1} \leftarrow V_\ell$
	else if $b = 1$ **then** $V_{\ell+1} \leftarrow \phi_{\ell+1}(V_{\ell+1}, \lvert V_\ell \rvert)$
	else $V_{\ell+1} \leftarrow \phi_{\ell+1}(V_{\ell+1}, V_\ell)$
$\texttt{set}(\ell, k)$	: **if** $\mathit{before} + 1 - k > 0$ **then** $V_\ell \leftarrow \mathrm{id}_\ell$
	else if $\mathit{before} + 1 - k = 0$ **then** $V_\ell \leftarrow \delta_f^i$
	else $V_\ell \leftarrow \phi_\ell(\delta_f^{i-k+1+\mathit{before}}, \ldots, \delta_f^i)$

The Macro-Instructions

letter		precondition	macro instruction code
$\mathbf{maybe}_{\mathrm{b}}$	:	$\begin{pmatrix} s \notin skip \wedge \\ d > before \end{pmatrix}$	$\texttt{compute}\left(1, 0, \delta_f^i\right)$, $\texttt{transmit}(0,0,0)$, $\texttt{reset}(0)$
		$\begin{pmatrix} s \in skip \wedge \\ d > before \end{pmatrix}$	$\texttt{compute}\left(0, 0, \delta_f^i\right)$
$\mathbf{maybe}_{\mathrm{r}}^k$	:		$\texttt{reset}(1)$, $\texttt{set}(1, k)$
$\mathbf{out}_{\mathrm{r}}$	:		$\texttt{reset}(1)$
found	:		$\texttt{compute}(1, balance, \kappa)$, $\texttt{transmit}(1,0,1)$, $\texttt{reset}(1)$
$\mathbf{maybe}_{\mathrm{a}}$	:		$\texttt{compute}\left(1, balance, \delta_f^{i+a-1-after}\right)$
in	:		$\texttt{compute}\left(1, balance, \delta_f^{i+a-1-after}\right)$, $\texttt{transmit}(0,0,1)$, $\texttt{reset}(1)$
end	:		$\texttt{transmit}(0, balance, 2)$, $\texttt{transmit}(0,0,3)$, $\texttt{reset}(2)$

Limitations

- Complexity of scheme
  - Initial learning curve
  - Amortized over many constraints
- No guarantees about consistency levels achieved
  - More reasoning possible (similar to time-series results)
  - Better correct and slow than fast and wrong