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
Key Ideas

- Describe mechanism to define many global constraints declaratively
- Seen as functional constraints on a sequence \( c(R, [X_1, X_2, \ldots, X_n]) \)
- 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

\[ X_1, \ldots, X_n \] → Integer Sequence

\[ S_1, \ldots, S_{n-a+1} \] ← Signature Sequence

Pattern Recognition on Signature Sequence

Adjustment with Trimming Parameters

Feature Computation Based on \( x_1, \ldots, x_n \)

Feature Values (\( p \) Features)

Aggregation Level 1

First Level Aggregate

Aggregation Level 2

Result Value
Signature Definition

- Time-Series: binary $<, =, >$
- Unary example: $\in, \notin$ for value set $V, \max_{width\_group}$
- Other binary example: $=, \neq, \text{nb\_stretch}$
- 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
- Generator
  - Automaton
    - Rewriting
      - CP/MIP models
  - Implementation
    - Global Constraints
  - Analysis Tools
    - Properties

- Instruction Set
- Data
- Tools
- Background
- Generated

Insight Centre for Data Analytics
ModRef 18, August 26, 2018
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
bump_on_decreasing_sequence, Reg. expr’ >>><<>>‘, Signature ‘<==>’
Example Recognition

> > > > < > > >

out  out  out  \( m_b \)  \( m_b \)  \( f_e \)

\( m_r^1 \)  \( m_b \)  \( m_r^2 \)  \( m_b \)  \( m_b \)  \( f, e, m_r^2 \)

Old (Only for \( b=2 \))

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

<table>
<thead>
<tr>
<th>micro instruction</th>
<th>register updates</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>compute(\ell, b, v)</code></td>
<td><strong>if</strong> $b = 0$ <strong>then</strong> $V_\ell \leftarrow \phi_\ell(V_\ell, v)$ <strong>else</strong> $V_\ell \leftarrow \phi_\ell(V_\ell, -v)$</td>
</tr>
<tr>
<td><code>reset(\ell)</code></td>
<td><strong>for</strong> $k \in [0, \ell]$ <strong>do</strong> $V_k \leftarrow \text{id}_k$</td>
</tr>
</tbody>
</table>
| `transmit(c, b, \ell)` | **if** $c = 1$ **then** $V_{\ell+1} \leftarrow V_\ell$  
**else if** $b = 1$ **then** $V_{\ell+1} \leftarrow \phi_{\ell+1}(V_{\ell+1}, |V_\ell|)$  
**else** $V_{\ell+1} \leftarrow \phi_{\ell+1}(V_{\ell+1}, V_\ell)$ |
| `set(\ell, k)` | **if** $\text{before} + 1 - k > 0$ **then** $V_\ell \leftarrow \text{id}_\ell$  
**else if** $\text{before} + 1 - k = 0$ **then** $V_\ell \leftarrow \delta_f^i$  
**else** $V_\ell \leftarrow \phi_\ell(\delta_f^{i-k+1+\text{before}}, \ldots, \delta_f^i)$ |
### The Macro-Instructions

<table>
<thead>
<tr>
<th>letter</th>
<th>precondition</th>
<th>macro instruction code</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{maybe}_b$</td>
<td>$(s \notin \text{skip} \land \ (d &gt; \text{before})$</td>
<td>compute $(1, 0, \delta_f^i)$, transmit $(0, 0, 0)$, reset $(0)$</td>
</tr>
<tr>
<td></td>
<td>$(s \in \text{skip} \land \ (d &gt; \text{before})$</td>
<td>compute $(0, 0, \delta_f^i)$</td>
</tr>
<tr>
<td>$\text{maybe}_r^k$</td>
<td></td>
<td>reset $(1)$, set $(1, k)$</td>
</tr>
<tr>
<td>$\text{out}_r$</td>
<td></td>
<td>reset $(1)$</td>
</tr>
<tr>
<td>$\text{found}$</td>
<td></td>
<td>compute $(1, \text{balance}, \kappa)$, transmit $(1, 0, 1)$, reset $(1)$</td>
</tr>
<tr>
<td>$\text{maybe}_a$</td>
<td></td>
<td>compute $(1, \text{balance}, \delta_f^{i+a-1-\text{after}})$</td>
</tr>
<tr>
<td>$\text{in}$</td>
<td></td>
<td>compute $(1, \text{balance}, \delta_f^{i+a-1-\text{after}})$, transmit $(0, 0, 1)$, reset $(1)$</td>
</tr>
<tr>
<td>$\text{end}$</td>
<td></td>
<td>transmit $(0, \text{balance}, 2)$, transmit $(0, 0, 3)$, reset $(2)$</td>
</tr>
</tbody>
</table>
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