Automata with Parameterized Arrays and Parameterized Networks of Automata

Tomáš Vojnar, Ahmed Bouajjani

LIAFA

1. Automata with Arrays

- **generalized arrays** — combining classical indexing in some dimensions with path-based, tree-like addressing in other dimensions

- dimensions/arity fixed, bounds/height may be parametric

1. Automata with Arrays

- **generalized arrays — combining classical indexing in some dimensions with path-based, tree-like addressing in other dimensions**

- **dimensions/arity fixed, bounds/height may be parametric**

- **an automaton with arrays** $M_{\mathcal{D},I} = (P, A, S, Q, T, C_0)$ **where:**
  - P, A, Q **— finite, disjoint sets of parameters, arrays, and states**

1. Automata with Arrays

- **generalized arrays — combining classical indexing in some dimensions with path-based, tree-like addressing in other dimensions**

- **dimensions/arity fixed, bounds/height may be parametric**

- **an automaton with arrays** $M_{\mathcal{D},I} = (P, A, S, Q, T, C_0)$ **where:**
  - P, A, Q **— finite, disjoint sets of parameters, arrays, and states**
  - **for each** $a \in A$**,** $s(a) = (D^a, (K_1^a, ..., K_{n^a}^a))$ **where** $D \in \mathcal{D}$ **and** $\forall_{n^a} i : K_i^a = b_i^a \in \mathbb{N}^+ \cup P$ **or** $K_i^a = (r_i^a, h_i^a) \in \mathbb{N}^+ \times (\mathbb{N}^+ \cup P)$

1. Automata with Arrays

- ❖ **generalized arrays — combining classical indexing in some dimensions with path-based, tree-like addressing in other dimensions**

- ❖ **dimensions/arity fixed, bounds/height may be parametric**

- ❖ **an automaton with arrays** $M_{\mathcal{D},I} = (P, A, S, Q, T, C_0)$ **where:**
  - P, A, Q **— finite, disjoint sets of parameters, arrays, and states**
  - **for each** $a \in A$**,** $s(a) = (D^a, (K^a_1, ..., K^a_{n^a}))$ **where** $D \in \mathcal{D}$ **and** $\forall_{n^a} i : K^a_i = b^a_i \in \mathbb{N}^+ \cup P$ **or** $K^a_i = (r^a_i, h^a_i) \in \mathbb{N}^+ \times (\mathbb{N}^+ \cup P)$
  - T **is a finite set of transitions** $t = (q_1, q_2, \tau)$ **where**
    $\tau ::= [(\forall|\exists)_b x \mid (\forall|\exists)_{r,h} x]^* :$
    $<$**guard on** $P, A, \{x, ...\} > \to <$**assignment on** $P, A, \{x, ...\} >$

1. Automata with Arrays

❖ **generalized arrays — combining classical indexing in some dimensions with path-based, tree-like addressing in other dimensions**

❖ **dimensions/arity fixed, bounds/height may be parametric**

❖ **an automaton with arrays** $M_{\mathcal{D},I} = (P, A, S, Q, T, C_0)$ **where:**

- P, A, Q **— finite, disjoint sets of parameters, arrays, and states**
- **for each** $a \in A$**,** $s(a) = (D^a, (K_1^a, ..., K_{n^a}^a))$ **where** $D \in \mathcal{D}$ **and** $\forall_{n^a} i : K_i^a = b_i^a \in \mathbb{N}^+ \cup P$ **or** $K_i^a = (r_i^a, h_i^a) \in \mathbb{N}^+ \times (\mathbb{N}^+ \cup P)$
- T **is a finite set of transitions** $t = (q_1, q_2, \tau)$ **where**
  $\tau ::= [(\forall|\exists)_b x \mid (\forall|\exists)_{r,h} x]^* :$
  $<$**guard on** $P, A, \{x, ...\} > \to <$**assignment on** $P, A, \{x, ...\} >$
  guards and assignments built on terms and atoms w.r.t. $\mathcal{D}$, I and
  $t_p ::= \varepsilon \mid x \mid t_P.t_P \mid t_I$ and $af_P ::= t_P \leq t_P \mid |t_P| \leq t_I \mid |t_P| \leq |t_P|$

1. Automata with Arrays

- **generalized arrays — combining classical indexing in some dimensions with path-based, tree-like addressing in other dimensions**

- **dimensions/arity fixed, bounds/height may be parametric**

- **an automaton with arrays** $M_{\mathcal{D},I} = (P, A, S, Q, T, C_0)$ **where:**
  - P, A, Q **— finite, disjoint sets of parameters, arrays, and states**
  - **for each** $a \in A$**,** $s(a) = (D^a, (K_1^a, ..., K_{n^a}^a))$ **where** $D \in \mathcal{D}$ **and** $\forall_{n^a} i : K_i^a = b_i^a \in \mathbb{N}^+ \cup P$ **or** $K_i^a = (r_i^a, h_i^a) \in \mathbb{N}^+ \times (\mathbb{N}^+ \cup P)$
  - T **is a finite set of transitions** $t = (q_1, q_2, \tau)$ **where**
    $\tau ::= [(\forall|\exists)_b x \mid (\forall|\exists)_{r,h} x]^* :$
    <**guard on** $P, A, \{x, ...\}$ > $\rightarrow$ <**assignment on** $P, A, \{x, ...\}$ >
    guards and assignments built on terms and atoms w.r.t. $\mathcal{D}$, I and $t_p ::= \varepsilon \mid x \mid t_P.t_P \mid t_I$ and $af_P ::= t_P \leq t_P \mid |t_P| \leq t_I \mid |t_P| \leq |t_P|$
  - C_0 **is a finite set of initial configurations** (q, ι)

- the associated **modelling language constructs**:
  - `type ::= tree [r,h] of type-id`
  - `forall(b):|exists(b):|forall(r,h):|exists(r,h):`
  - ```
    tp  ::= eps|x|tp.tp|ti
    afp ::= tp<=tp|#(tp)<=ti|#(tp)<=#(tp)

            root(x)|leaf(r,h,x)|inner(r,h,x)|son(x,y)
    ```
    ```

2. Networks of Automata

- $\mathcal{D}$, I **as a basis**
- **a set of global variables** G **(+ initial constraints)**
- **a process infrastructure** $a = (K_1^a, ..., K_{n^a}^a)$ **with** K_i^a **as in arrays**

2. Networks of Automata

- $\mathcal{D}$, I as a basis
- a set of global variables G (+ initial constraints)
- a **process infrastructure** $a = (K_1^a, ..., K_{n^a}^a)$ with K_i^a as in arrays
- for each particular automaton (process):
  - a constraint on the **identifiers** of the instances of the process corresponding to addresses in a

2. Networks of Automata

- **$\mathcal{D}$, I as a basis**
- **a set of global variables G (+ initial constraints)**
- **a process infrastructure $a = (K_1^a, ..., K_{n^a}^a)$ with K_i^a as in arrays**
- **for each particular automaton (process):**
  - **a constraint on the identifiers of the instances of the process corresponding to addresses in a**
    - **a variable-based address with a constraint on the variables, i.e., $((x_1, ..., x_{n^a}), \varphi)$, e.g., $((p), \exists_{2,h} q : 0 < |q| < h \wedge p = q)$**
    - **a simplification — using just constants, parameters, and address constructors (**root(r,h), leaf(r,h), inner(r,h), in(l,h)**), e.g.,** process router(inner(2,h))

2. Networks of Automata

- **$\mathcal{D}$, I as a basis**
- **a set of global variables G (+ initial constraints)**
- **a process infrastructure $a = (K_1^a, ..., K_{n^a}^a)$ with K_i^a as in arrays**
- **for each particular automaton (process):**
  - **a constraint on the identifiers of the instances of the process corresponding to addresses in a**
    - **a variable-based address with a constraint on the variables, i.e., $((x_1, ..., x_{n^a}), \varphi)$, e.g., $((p), \exists_{2,h} q : 0 < |q| < h \wedge p = q)$**
    - **a simplification — using just constants, parameters, and address constructors (**root(r,h), leaf(r,h), inner(r,h), in(l,h)**), e.g.,** process router(inner(2,h))
  - **a set of local variables L, a set of states Q, and local initial constraints**
  - **transitions $(q1, q2, \tau)$ with guards and assignments over G, id, L, x, ...**

- **may be relatively easily mapped onto an automaton with arrays**
- **may be extended by:**
  - **global parameterized arrays**
  - **local parameterized arrays (yielding arrays whose structure is a "concatenation" of the process infrastructure and the local arrays)**
- **some limitations/open problems:**
  - **parameterized dimensions/arities**
  - **general graph architectures**
  - **dynamic instantiation**