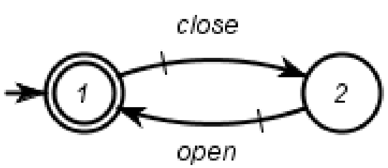
Techniques for the Parametrization of DES Templates

Lenko Grigorov and Karen Rudie

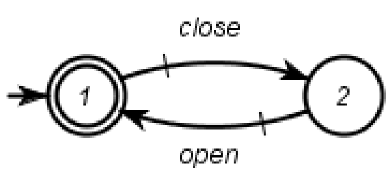
Dept. Electrical and Computer Engineering
Queen's University
2010

Motivation (templates)

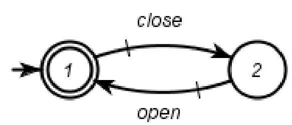
Press 1



Press 2



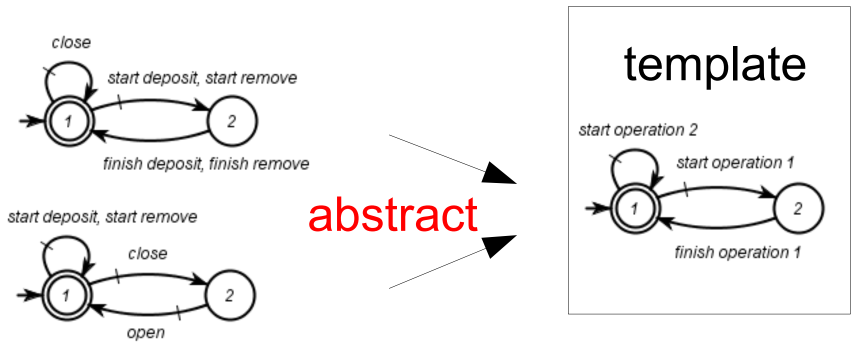
Press 3



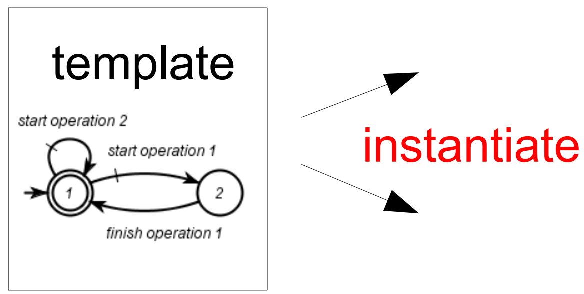
…

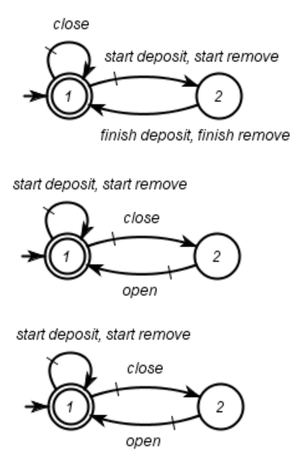
What are templates

- DES models which
  - Abstract common behavior

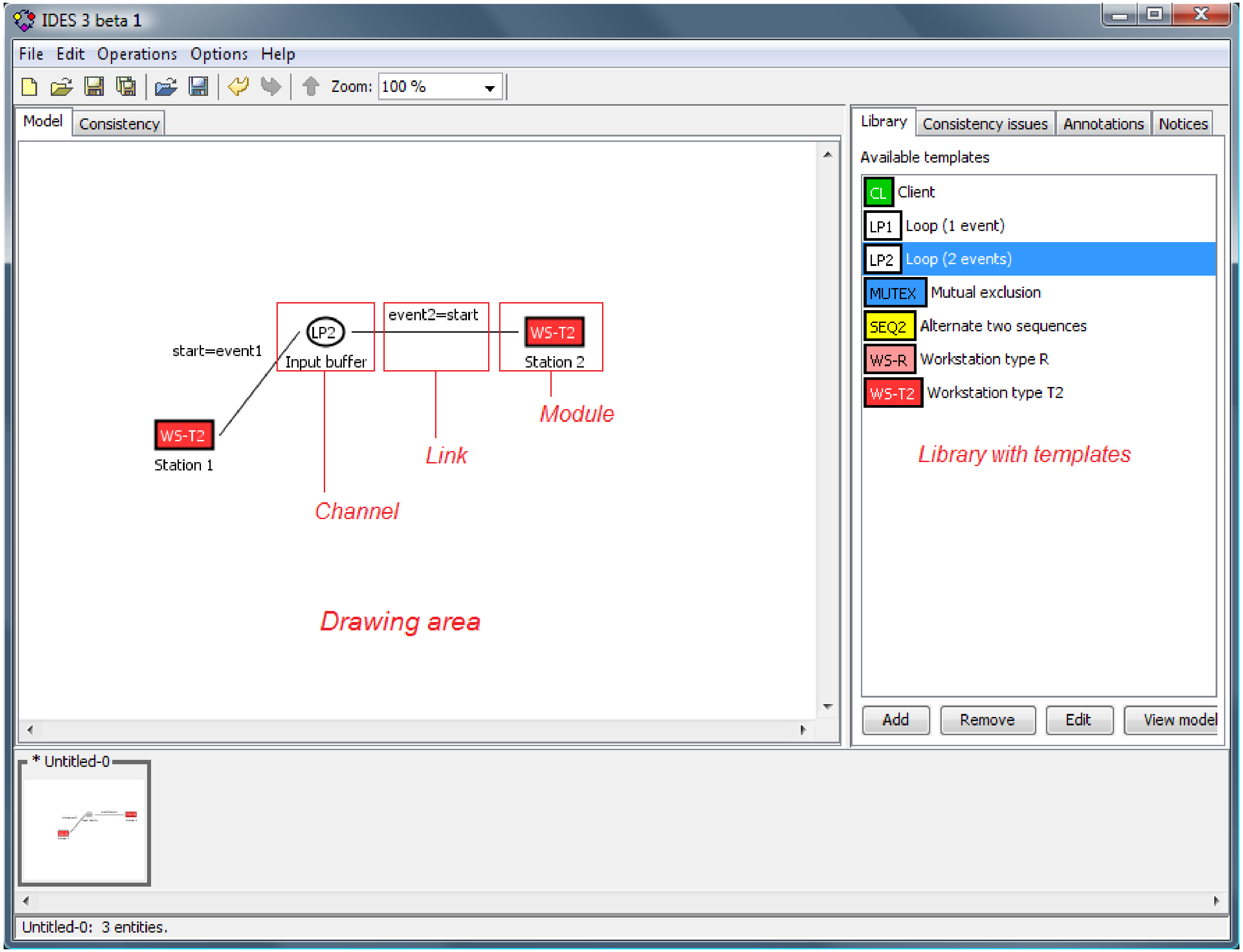


  - Can be instantiated to create new models



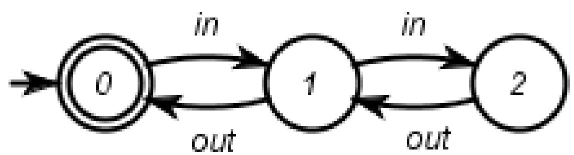


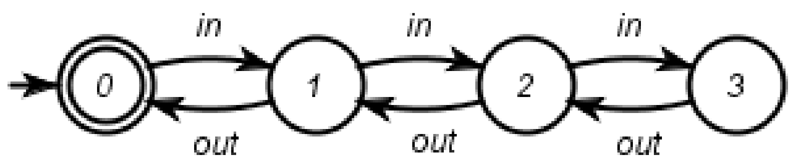
Template design environment



Motivation (parametrization)

- A template is a fixed model
  - So... a separate template for each buffer capacity?



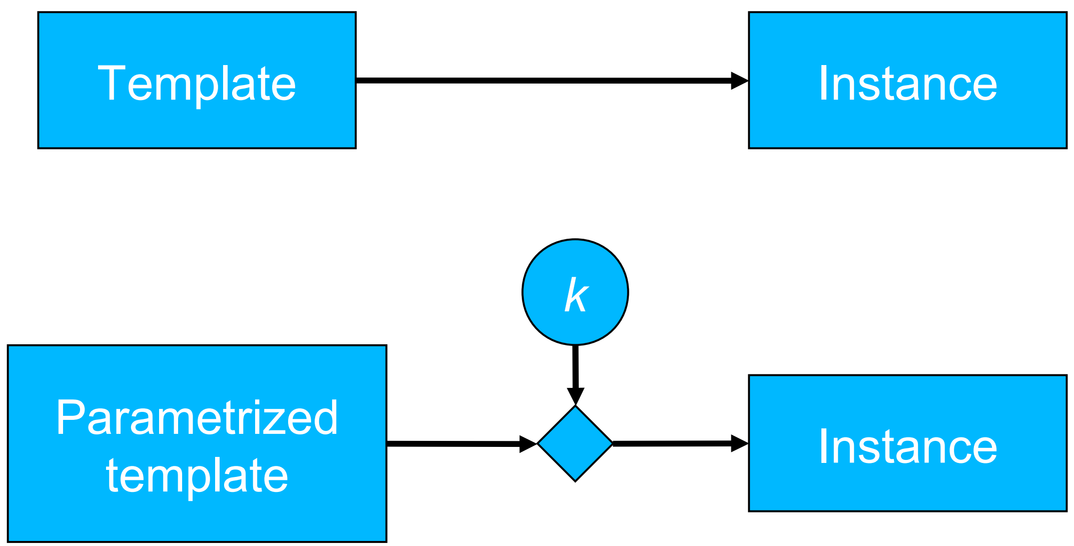


etc...

  - Why not just parametrize?

Goal

Develop parametrization techniques to produce model instances from templates, given a parameter



Two techniques

- Parametrization using variables
  - For example, use Extended Finite Automata from

  Sköldstam, M., Åkesson, K., and Fabian, M. (2007). Modeling of discrete event systems using finite automata with variables. In *Proceedings of the 46th IEEE Conference on Decision and Control*, 3387–3392. New Orleans, LA, USA
- Compositional parametrization
  - Similar in spirit to

  Bherer, H., Desharnais, J., and St-Denis, R. (2009). Control of parameterized discrete event systems. Discrete Event Dynamic Systems: Theory and Applications, 19(2), 213–265

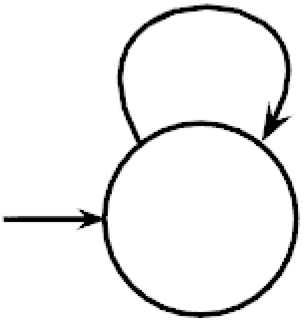
Parametrization using variables

- Can be used to parametrize the *occurrences* of events
- The finite automaton is extended with
  - (finite-interval) variables
  - update functions
  - transition guards
- An algorithm exists to convert extended finite automata to “flat” finite automata

Example 1
Buffer

- Parameter *k*
- Variable *n* from [0, *k*]
  - initial value *n*=0
  - accepting value *n*=0

$$\mathbf{in} \wedge n < k \;/\; n := n + 1,\ \mathbf{out} \wedge n > 0 \;/\; n := n - 1$$

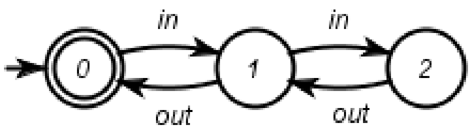


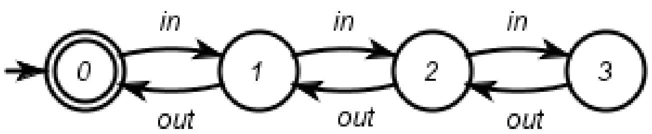
Example 1
Buffer

Parameter *k*

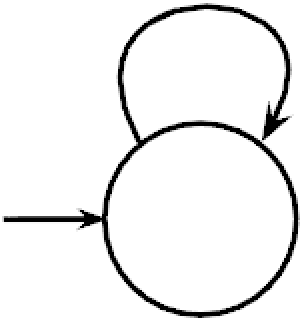
Variable *n* from [0, *k*]

- initial value *n*=0
- accepting value *n*=0



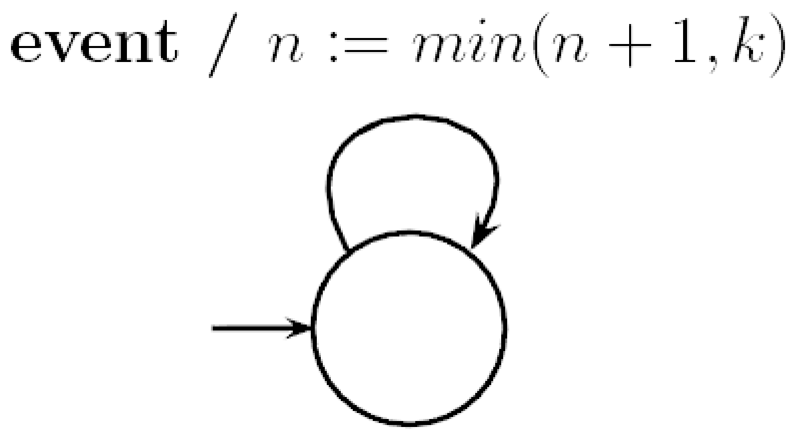


$$\mathbf{in} \wedge n < k \ / \ n := n + 1, \ \mathbf{out} \wedge n > 0 \ / \ n := n - 1$$



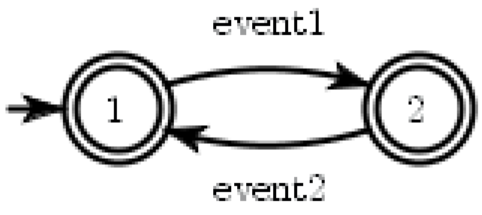
Example 2
Event occurs *k* times

- Parameter *k*
- Variable *n* from [0, *k*]
  - initial value *n*=0
  - accepting value *n*=*k*

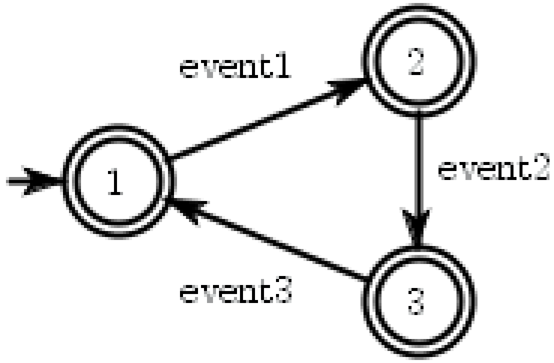


How to parametrize events?

- Sometimes it is necessary to parametrize the *events* of a model rather than the event occurrences.
- Example: *k* events have to occur in sequence.
  - Cannot be done with variables



k=2



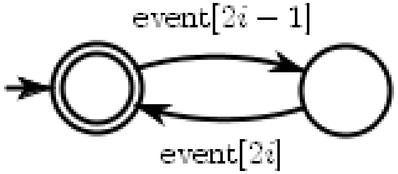
k=3

Compositional parametrization

- Define a prototype model with indexed events
  - Indices are functions of *i*
- Choose a parameter *k* and get individual components
  - For each *i* from lo(*k*) to hi(*k*), create models where events are indexed accordingly
- Compose the components using synchronous product

Illustration

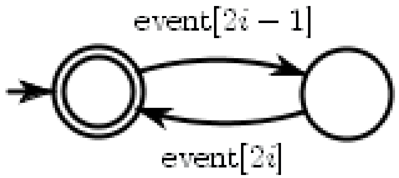
Prototype



lo(k) = 1
hi(k) = k

Illustration

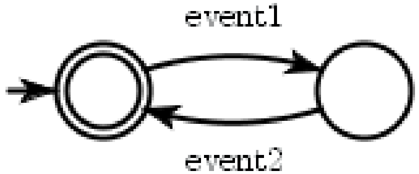
Prototype



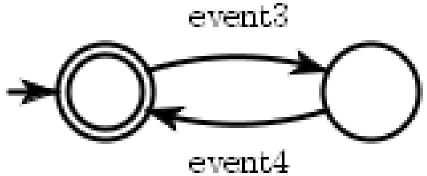
lo(*k*) = 1
hi(*k*) = *k*

Components for *k = 2*

i = 1

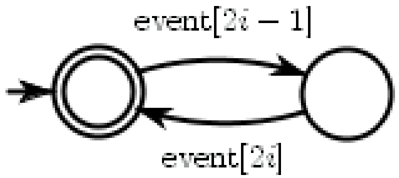


i = 2



Illustration

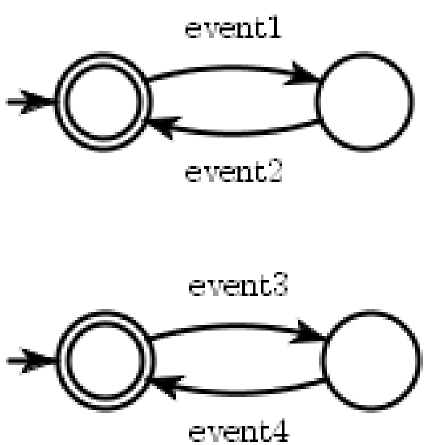
Prototype



lo(*k*) = 1
hi(*k*) = *k*

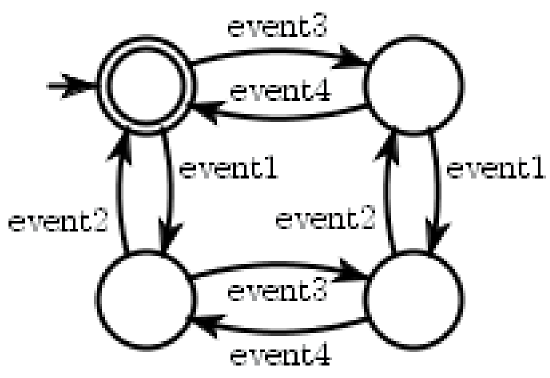
Components for *k = 2*

i = 1



i = 2

Final model for *k = 2*



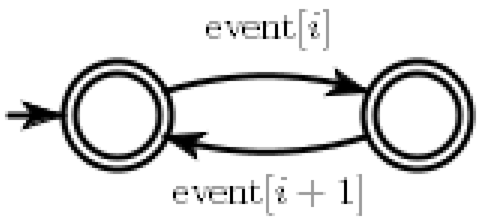
Compositional parametrization (definition)

- Let $G^p = (\Sigma_F, Q, \delta, q_0, Q_m)$
  - $\Sigma_F = \{ \sigma_f \mid f$ is some function **N** to **N** $\}$
- Let $G^p(i) = (\Sigma(i), Q, \delta(i), q_0, Q_m)$
  - Where σ_f have been replaced by $\sigma_{f(i)}$
- Let $\langle G^p,lo,hi\rangle = ||_{i\in [lo,hi]}\ G^p(i)$
- Compositional parametrization

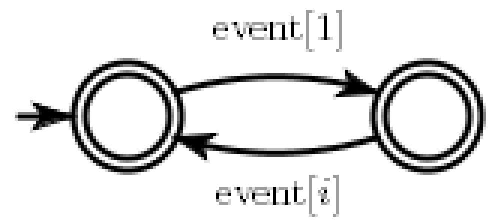
$$G[k] = A||\langle G^p_1,lo_1(k),hi_1(k)\rangle||\ldots||\langle G^p_n,lo_n(k),hi_n(k)\rangle$$

Example 3
k events occur in sequence

Prototypes



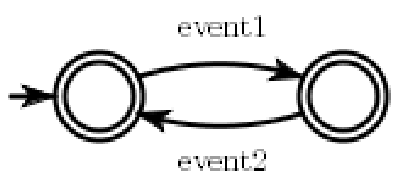
lo(k) = 1
hi(k) = k-1



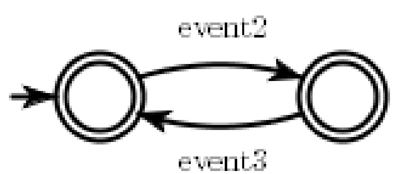
lo(k) = k
hi(k) = k

Components for $k = 3$

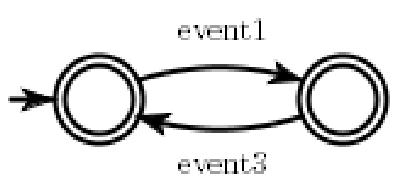
i = 1



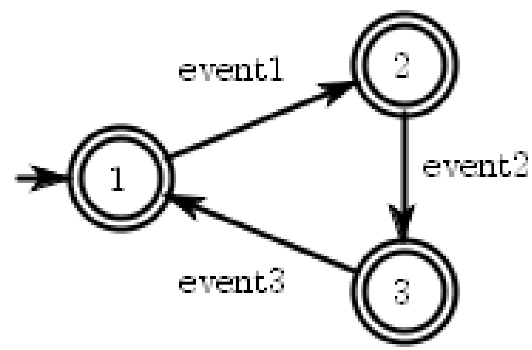
i = 2



i = 3

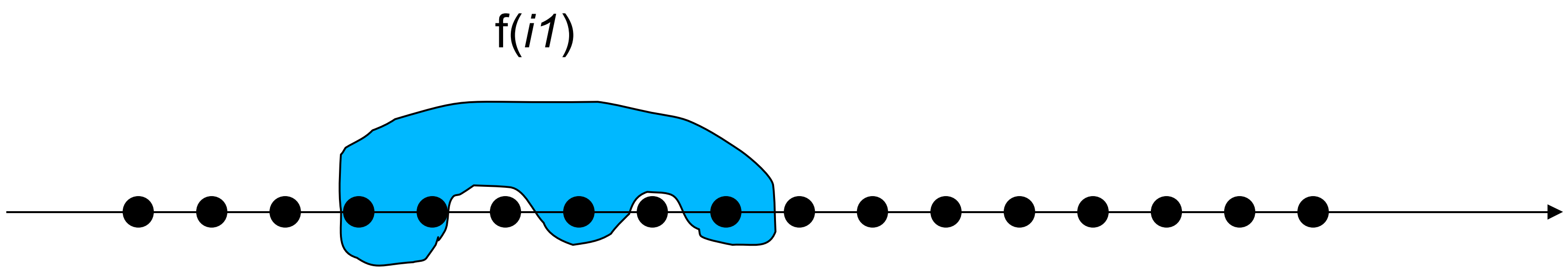


Final model for $k = 3$



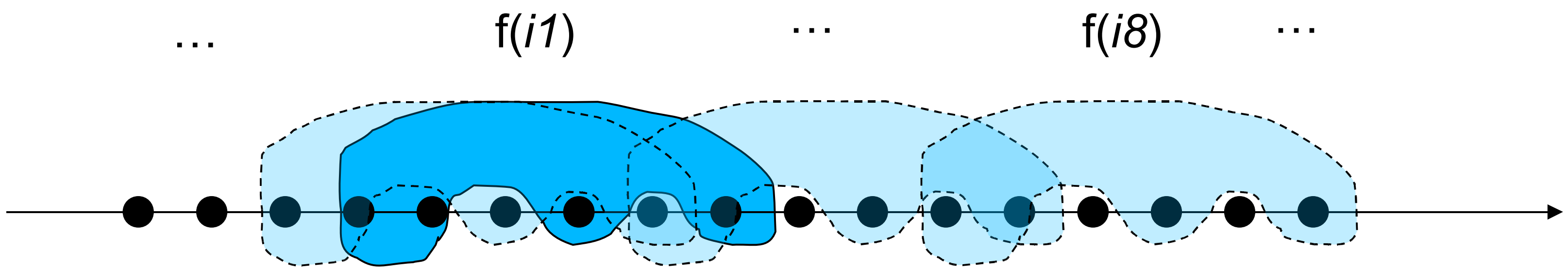
Synchronization scope (local)

- Event indices can be expressed as a function of *i* in some range
  - Create repeated patterns of synchronization



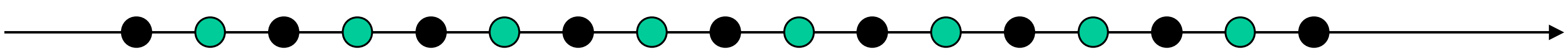
Synchronization scope (local)

- Event indices can be expressed as a function of *i* in some range
  - Create repeated patterns of synchronization



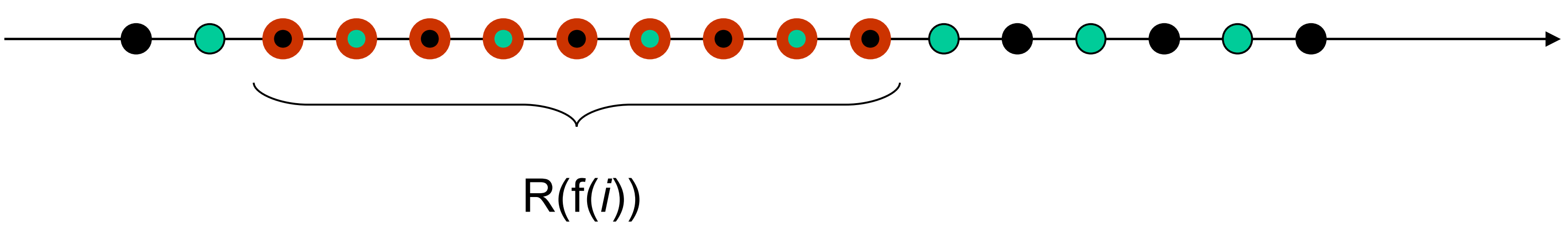
Synchronization scope (global)

- Sometimes we need to synchronize over *all* events
  - Mutual exclusion
  - **Depends on the specific value of *k* !**
- Selector transitions
  - Event indices are characteristic functions h(*i*)



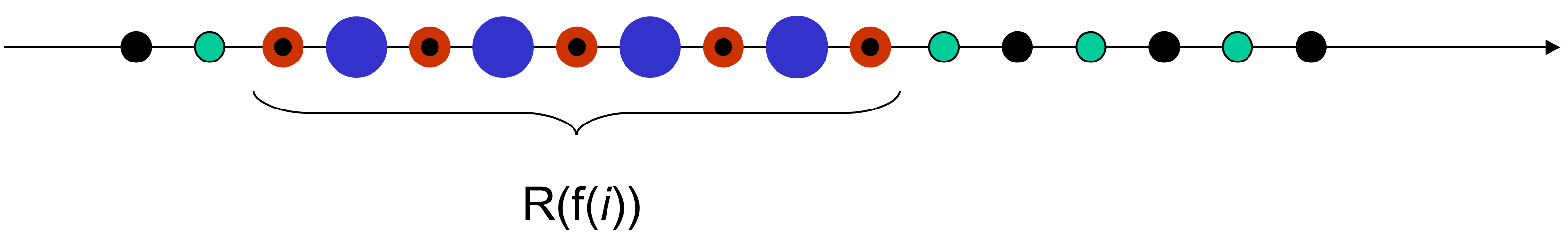
Synchronization scope (global)

- Sometimes we need to synchronize over *all* events
  - Mutual exclusion
  - **Depends on the specific value of *k* !**
- Selector transitions
  - Event indices are characteristic functions h(*i*)



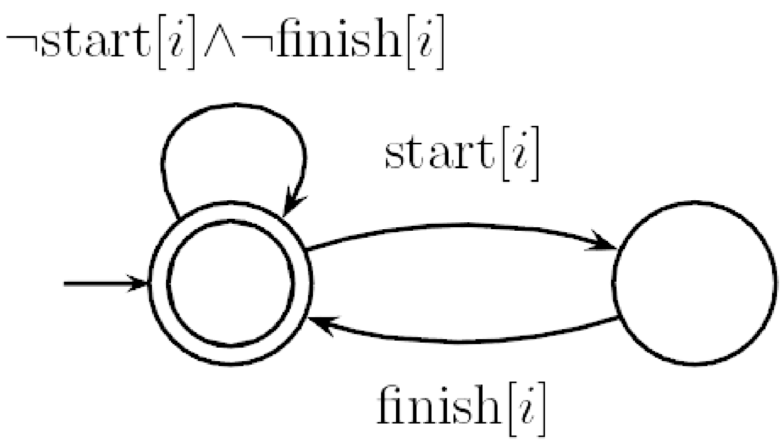
Synchronization scope (global)

- Sometimes we need to synchronize over *all* events
  - Mutual exclusion
  - **Depends on the specific value of *k* !**
- Selector transitions
  - Event indices are characteristic functions h(*i*)



Example 4
Mutual exclusion

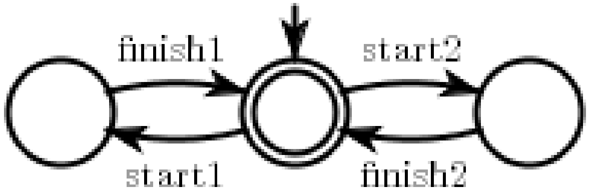
Prototype



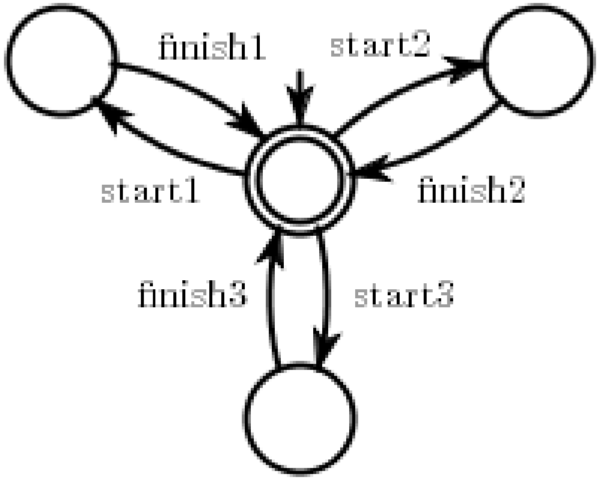
lo(k) = 1
hi(k) = k

Final models

k=2



k=3



Final remarks

- We proposed two techniques for the parametrization of templates
- Three types of patterns can be expressed
  - Occurrences of events
  - Local synchronization
  - Non-local synchronization
- Future work
  - Implementation
  - “In-use” parametrization
  - Direct controller synthesis for parametrized models similar to Bherer *et al.* (2009)