Context Migration Machine

Cam Feenstra

May 2020

Define a **contextual finite state machine** to be the following (using the FSM notation from wikipedia):

1. Σ - A set of input symbols
2. S - A finite non-empty set of states
3. $s_0 \in S$ - An initial state
4. $F \subseteq S$ - A possibly empty set of final states
5. G - The dependency scope, a set of contextual state machines that this one might depend on.
6. $d : \Sigma \to G$ - A dependency function
7. $t : \{(\varepsilon, \{\prod_{m \in d(\varepsilon)} S_m\}) : \varepsilon \in \Sigma\} \times S \to S$ - The transition function. Note that this function takes as input an input and symbol and the a state, just like a typical FSM. However, it also takes the state of machines that are defined by its dependency function depending on the input symbol.

This definition implies that the finite state machine is a non-deterministic finite state, as for any state an input symbol the transition function δ can be defined as:

$$\delta(\varepsilon, s) = \{t(\varepsilon, (s_1, ..., s_k), s) : s_i \in S_i : 1 \leq i \leq k\} \subseteq \mathcal{P}(S) \tag{1}$$

Since all deterministic finite state machines also meeting the criteria for non-deterministic finite state machines, this does not preclude it from also being a deterministic finite state machine. For example, if $d(\varepsilon) = \emptyset \ \forall \varepsilon \in \Sigma$ the function δ can be defined as $\delta(\varepsilon, s) = t(\varepsilon, \emptyset, s) \ \forall s \in S, \varepsilon \in \Sigma$ and by definition it is a deterministic FSM.

Now define an **finite state machine context** to be:

1. $G = \{m_1, ..., m_k\}$ - A non-empty set of contextual finite state machines whose dependency scopes are subsets of G. Define the following properties of G:
   (a) S_g - The product of all possible states of the machines in G, or $\prod_{1 \leq i \leq k} S_i$.
   (b) Σ_g - The product of all possible input symbols of the machines in G or $\prod_{1 \leq i \leq k} \Sigma_i$.
   (c) $D(\varepsilon)$ - The edges defined by $\{(m_x, m_y) : m_x \in d_y(\varepsilon_y) : m_y \in G\}$ for an input ε.
2. For all $\varepsilon \in \Sigma_g$ the graph defined with the elements of G as its nodes and the elements of $D(\varepsilon)$ as its edges is a directed acyclic graph. In other words, for a given set of input symbols none of the machines may depend on one another or any of one another's dependencies.

From this definition, we can define a deterministic finite state machine for any finite state machine context that will migrate each of the component contextual state machines in a deterministic manner.

Define a **context migration machine** as:

1. A finite state machine context $G = \{m_1, ..., m_k\}$
2. $\Sigma = \prod_{1 \leq i \leq k} \Sigma_i$
3. $S = \prod_{1 \leq i \leq k} S_i$
4. $s_0 = (s_{01}, ..., s_{0k})$
5. $F = (F_1, ..., F_k)$
6. $T : S \times \{(m, \varepsilon) : \varepsilon \in \Sigma_m : m \in G\} \to S$ - The *step transition function*, to be described below.
7. $\delta : \Sigma \times S \to S$ - The transition function, to be described below.

First, we will define the step transition function T. T can be defined quite simply: for some initial state s, some input machine m, and some input symbol ε, $T(s, m, \varepsilon)$ returns the same value as s except that the state corresponding to the machine m is updated to be:

$$t_m(\varepsilon, (s_1, ..., s_k), s_m) \tag{2}$$

where $(s_1, ..., s_k)$ are the states in s of the machines returned by $d_m(\varepsilon)$ and s_m is the state of m.

Now our transition function δ can be defined. Assume we have a tuple of input symbols $\varepsilon = (\varepsilon_1, ..., \varepsilon_k)$ and an existing state $s = (s_1, ..., s_k)$.

Note that the state of the context migration machine consists of two parts: it contains a state for each of its component machines, and it contains a set of edges between machines.

Define a graph V whose nodes are the elements of G and whose edges are $D(\varepsilon)$, and define a sequence of time steps $(c_1, ..., c_k) = (1, ..., k)$. Since the fact that G is a finite state machine context implies V is a DAG, we can order the machines according to some topological ordering of V and assign each machine $m_1, ..., m_k$ to a time step $c_1, ..., c_k$. Denote c_m to be the time step assigned to each machine m and m_c to be the machine assigned to each time step.

Define the state s'_c to be the current state of the migration machine at some time step c. Also define ε_c as the input symbol in ε corresponding to the machine m_c. The state s_c is defined recursively, with the output state being defined as s'_k:

$$s'_c = T(s'_{c-1}, \varepsilon_c, m_c) \tag{3}$$

Now we will show by induction that there is only one possible output state for a given input state and tuple of input symbols. We specifically will show that for all $1 \leq i \leq k$, the output state of m_i has only one possible value, and since the output state of the migration machine a set of all of these states that can only then one have possible value as well.

Let $i = 1$. By the ordering declared above, this corresponds to some time step c and thus some machine m_c. By definition of topological ordering, since this machine is first in the ordering it implies that $d_{m_c}(\varepsilon_{m_c}) = \emptyset$. This means that for this input symbol, the machine does not depend on any states other than its own, and thus the output state can only be one possible value. Therefore, since the definition of T implies that the only machine whose state is different in s'_1 from s'_0 is m_c, we can conclude that there is only one possible value of s'_i.

Now assume that s'_n has only one possible value for some $1 \leq n \leq k$, and define $i = n + 1$. This corresponds to some time step c and some machine m_c. The only machine whose state can differ in s'_i from s'_{i-1} is m_c. The state of m_c in s'_i is defined by:

$$t_c(\varepsilon_c, (s'_{i-1,m} : m \in d_c(\varepsilon_c)), s'_{i-1,c}) \tag{4}$$

Because this only depends on the inputs and the previous state which only has one possible value, we can conclude that there is only one possible state of m_c in s'_i. Therefore, since it is the only machine whose state can differ from s'_{i-1}, we can conclude there is only one possible value of s'_1.