RGD

Jumping Finite Automata: Application in Embedded Systems

Malaník Petr <imalanik@fit.vutbr.cz>

Abstract

Finite State Machines (FSM), deterministic, are basic tool in embedded world and are used widely because offers simple and clear approach to implement sequential process. Also FSM are considered to be safe and secure, because can be statically checked and simulated. Due to this properties are widely used also in automotive industry.

One of their weakness is dependence on sequential inputs. This is because there is an assumption that input action will be arriving with time delays. This also determines that they can only work with sequential static input (input word in this meaning) and in one direction. With assumption of time delay, is it not possible to do otherwise. Alternative use of FSM is to solve some initial configuration or find if there is a way how to solve this configuration. This use case is rather problematic if there are some dependencies between actions (state transitions). Result of these dependencies is extreme increase in the number of states of machine.

But this problem can be solved by using of Jumping Finite Automata (JFA) because they are not depending on sequential input but can jump over it and processed it with same efficiency like FSM on sequential input. But with increasing dependencies there is not enormous increase in number of states. Another advantage is that is much easier to design JFA for this purpose, then FSM with dependencies. There are also downsides, one of the biggest is that those are not as simple to validate as FSM. Statically check all configurations of machine and find invalid is much harder.

To create Jumping Finite Automata for embedded system the states, actions and dependencies must be defined. Actions are depended on states of the machine. Transitions can be generated from dependent actions. Independent actions are used to transition from initial state. So there must be at least one independent action in whole system. Input word is then processed by automata, until finite state is achieved.

From practical side of view JFA have an advantage that in comparison with FSM, they can minimize number of states. This is very helpful in embedded systems, because there are limitations for memory and performance. Elimination of states reduce memory requirements, which directly lower number of steps which are needed to find next available state (lower performance requirements).

Example of practical usage in embedded systems can be homing of device with higher number degrees of freedom. Those devices often have dependent axis homing. This means: before homing some axis, another must be in predefined (known) position. But with increasing number of axis and dependencies between them, grows also complexity of homing process. JFA can validate homming plan by processing it. If finish state is reached, plan is valid. Also by processing an initial configuration, sequence of transitions between states can be used as homing plan for device.