Concurrent Marked Graph Execution on Single-Thread Processors

Sam Spekreijse

02/10/2020

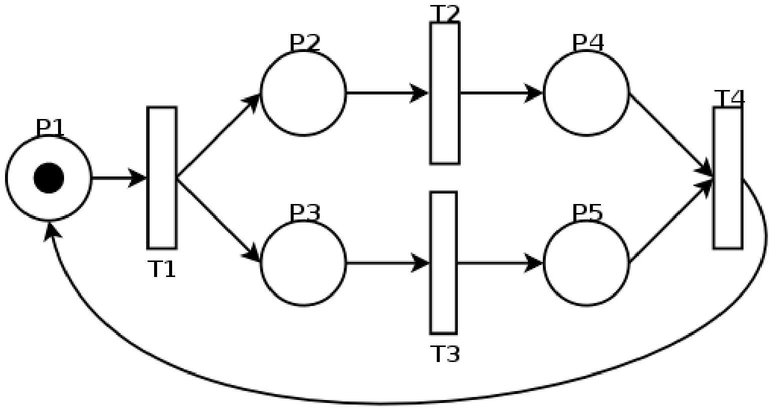
Marked Graphs

Marked graphs are directed graphs with alternating nodes of two types. These two types represent states and transitions.

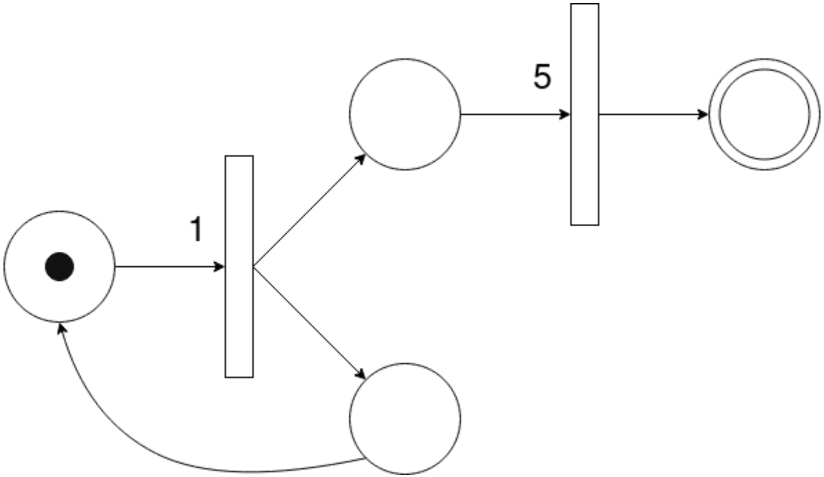
Similar to normal directed graphs, marked graphs can be stored in several ways. The two most common are:

- Adjacency List: Each vertex stores a list of edges.
- Adjacency Matrix: A lookup table for a given start and end node.

Wikipedia’s Basic Example



A More Interesting Example



Priority Properties

Priorities should be inherited from the ‘parent’ process every time a process forks.

How to store the priorities is therefore not straightforward. Both string-based and float-based systems were considered. At present, a linked list is used, and the priority of each process is not explicitly known.

Implementation

```
typedef struct Fsm_s Fsm_t;
typedef struct FsmState_s FsmState_t;
typedef int32_t FsmTransId_t;
typedef FsmTransId_t *(FsmFunc_t)(void);


typedef struct {
          bool isLockstep;
          size_t numProcesses;

          size_t numInProcesses;
          Process_t **processes;
          size_t numOutProcesses;
          FsmState_t **outStates;
} FsmTrans_t;
```

```
struct FsmState_s{
        //One of the following must be NULL.
        Fsm_t *subFsm;
        FsmFunc_t *func;

        size_t numTransitions;
        FsmTrans_t **transitions;
};


struct Fsm_s {
        Fsm_t *parent;
        FsmState_t *end;
        size_t numStates;
        FsmState_t *states[];
};
```

```
typedef struct process_s Process_t;
typedef struct process_s {
        bool isActive;
        FsmState_t *state;

        Process_t *prev;
        Process_t *next;
};
```

Additional Developments

Recursive UML statecharts support, where states can themselves also be FSMs, was also part of this project. This functionality can be implemented via FSM expansion at compile time, or by changing the runtime. The latter option has been implemented for debugging purposes.

The nature of marked graphs is that each transition can be used not only to synchronise processes, but to synchronise processors operating in lockstep. Each transition has the option of being a synchronising transition. This enables modular redundancy to be designed into the application from the first planning stages.