

# COMS 556      HOMEWORK 10

**Due 16 April 2007**

For this assignment, you will write code to generate a CTMC from a high-level model (specifically, a stochastic Petri net) to be read from standard input. The output of the program should be a CTMC, plus measures, in the format that is readable by your CTMC solver. You will be given code that parses the input file of the high-level model (using the `flex` utility), and builds the internal representation of the model. You will also be given a makefile; this can all be found under `/home/staff/asminer/public/Hw10/`. You will need to implement two functions (found in the file `gen.cc`):

```
int ShowCTMCForModel(const model* m, bool show, std::ostream &out);
```

This function is responsible for building the CTMC for model `m`, and writing it to the stream `out` in the format for your CTMC solver, from the keyword `CTMC` to the keyword `END`. If `show` is `true`, you should also display the reachable states as “comments”, e.g.:

```
    # State 15: [ p1: 5, p4: 2]
```

(this occurs if you call the main program with the switch `-s`). The format of a model can be found in the file `model.h`. The function should return 0 if it was successful, and non-zero otherwise.

```
void ShowRewardFunction(const measure* m, std::ostream &out);
```

This function cycles through all the reachable states and writes, to stream `out`, the state index and measure value for each state, but only if the value is not zero. The format of a measure can also be found in the file `model.h`.

## Input format

Blank lines, and comment lines (following `#`) are ignored. The format of the input file is: keyword `SPN`, followed by declaration of places, followed by initial number of tokens in places. Then, optionally, declaration of “immediate” transitions, followed by declaration of “timed” transitions. Finally, there can be any number of `GUARDS`, `ARCS`, and `INHIBITORS` declarations, in any order, until the keyword `END`. Following the `END` keyword are reward measures.

### PLACES

Places are declared as

```
PLACES
  <ident>
  <ident>
  ...
  <ident>
```

with any whitespace between place identifiers. Identifiers must be unique, and cannot be one of the keywords.

### INIT

The initial tokens for places are declared as

```
INIT
  <place> : <#tokens>
  ...
  <place> : <#tokens>
```

with any place not listed having a default of zero tokens initially.

## IMMEDIATE

Immediate transitions are declared as

```
IMMEDIATE
  <ident> : <weight expression> ;
  ...
  <ident> : <weight expression> ;
```

where the identifiers must be unique. Weight expressions are written in *postfix* notation, and may contain place identifiers to signify the current number of tokens in that place.

## TIMED

Timed transitions are declared as

```
TIMED
  <ident> : <rate expression> ;
  ...
  <ident> : <rate expression> ;
```

where, again, the rate expressions are in postfix notation and may contain place identifiers.

## GUARDS

Transition guards are declared as

```
GUARDS
  <trans> : <guard expression> ;
  ...
  <trans> : <guard expression> ;
```

where each transition must have been declared, and the guard expression evaluates to zero when the transition should be disabled (in addition to the usual enabling rule). Guard expressions are in postfix notation and may contain place identifiers.

## ARCS

Input and output arcs are declared as

```
ARCS
  <place> : <trans> : <cardinality expression> ;
  ...OR...
  <trans> : <place> : <cardinality expression> ;
```

where the cardinality expression is written in postfix notation and may contain place identifiers.

## INHIBITORS

Inhibitor arcs are declared similar to input arcs, as

```
INHIBITORS
  <place> : <trans> : <cardinality expression> ;
  ...
  <place> : <trans> : <cardinality expression> ;
```

where the cardinality expression is written in postfix notation and may contain place identifiers.

## Reward measures

Reward measures are defined as in your CTMC solver, except an expression is used instead of a list of state, value pairs. Specifically, these are declared as

```
TYPE
  <ident> : <reward expression> ;
  ...
  <ident> : <reward expression> ;
```

where TYPE is one of STEADY, ACCUMULATED, TIME  $t$ . The reward expression is written in postfix notation, and may contain place identifiers to signify the number of tokens in the place, or transition identifiers (for timed transitions) to signify the current rate of the transition (which will be zero if the transition is disabled).

## Ordinary credit

You must handle TIMED transitions and be able to produce CTMCs with hundreds of thousands of states. Print an error message and terminate if the model contains immediate transitions (unless you do the extra credit, below).

## Extra credit

### Size bonus

You will receive extra points if you can generate more than 1 million states. You will receive additional extra points if you can generate more than 10 million states. To receive this bonus, your program must

1. generate a correct CTMC.
2. use less than 3Gb of RAM.
3. complete in less than 30 minutes.

### Immediate transitions

You will receive (significant) extra credit if you can handle immediate transitions. You will receive points based on the level of difficulty of models that you can handle. In increasing order of difficulty, these are:

1. no vanishing loops.
2. no vanishing loops; detect vanishing loops, give an error message, and terminate.
3. handle vanishing loops as long as each vanishing state can eventually reach a tangible state.
4. handle vanishing loops; detect “absorbing” vanishing loops, give an error message, and terminate.