Colored Petri Net Evaluation Tool

Insert Token

Fire Transition

Send help message

Stephen Rojciewicz
CS 2310
Motivating Example (Colored Petri Nets)

- Consider a gesture-driven application interface. The system must detect three kinds of gestures and respond differently to each gesture.
- A Colored Petri Net provides an efficient and clean representation of the interface.
- Specific gestures are represented by token values, and the system’s corresponding responses are modeled using conditional transitions (see next slide)
Motivating Example (2)

- We specify input conditions on the three transitions as follows.
- Transition “Process Gesture A” can accept only tokens that have the value “Gesture A”.
- Transition “Process Gesture B” can accept only tokens with the value “Gesture B”, and “Process Gesture C” will only accept “Gesture C”.
- This way, we can use a very simple structure of one front-end place and three transitions to model this conditional system.
- To model this system using a non-Colored Petri net, we would either use a separate input place for each gesture, or give each transition a probability of firing according to the expected probability of detecting the corresponding gestures.
Motivating Example (3)

• The Colored Petri net representation is more concrete and can be easier to understand.
• In other words, Colored Petri nets allow for simpler representations by replacing cumbersome structural elements with token values and transition conditions.
• Colored Petri nets may also specify output functions, which allow transitions to perform computations on their input tokens.
• In this project, only input conditions have been implemented. Output functions are a topic of future work.
Features of the Colored Petri Net Evaluation Tool

- All tokens have a value, defined as a string
- Transitions may be equipped with conditions, which limit the kinds of tokens that they accept
- Simulation Mode for automatic execution
- Debug Mode for user-controlled execution
- Imports PNML files created by the PIPE editor.
- Since PIPE does not support colored Petri nets, the mechanism for defining values and conditions has been implemented in this project.
Specifying Conditions

In this example, the user is defining a condition on transition “Send help message”. This transition will accept only tokens that have the value “The Right Message”.
Specifying an Initial Marking

- Once the desired conditions have been specified, the Petri Net is ready for execution.
- The user begins by providing an initial marking.
Token Generator Tool (Simulation Mode)

- In simulation mode, transitions fire automatically.
- The user can assign token generators for specific places in the Petri Net. In the example below, the user is defining a token generator for the “Gesture received” place. One firing between generations specifies that a token will be generated after every other firing.
Breakpoints (Simulation Mode)

- Breakpoints can be defined in two ways. Marking breakpoints allow the user to specify certain place/token combinations that cause execution to stop. Transition breakpoints define sets of transitions that, when active, cause execution to stop.

In this example, the user has specified a marking breakpoint in order to stop execution any time the “Visited Patient” place contains a “Patient Y” token.
Below is an example of a Petri Net executing in Simulation Mode

Note that the user can stop (and resume) the simulation at any time
Debug Mode

• If a breakpoint is reached or execution is manually paused, the user can switch to debug mode. Simulation mode can be resumed later.
• In debug mode, the user initiates all transition firings.
• Debug mode allows the user to specify which transitions to fire when multiple transitions are active.
• The user can also insert tokens at any time.
Below is a screenshot of debug mode

The user is about to fire the “Call patient” transition.
Future Work

• More General conditions: In this project, all conditions are defined in terms of a set of tokens that a transition will accept. Conditions could be more powerful if they were specified separately for each input arc.

• Also, in this project, conditional tests are limited to equality and subset. Other tests, such as AND and OR, would be useful.
Future Work (2)

• User-specified output functions: The user should be able to specify how the value of a transition’s output token is computed. In the implementation of this project, one value is arbitrarily selected from the set of input tokens.

• With output functions, the user could model, at a much more concrete level, the actual computation performed by the system component that a transition represents.

• PIPE supports stochastic Petri nets. Future work could be done to integrate stochastic Petri nets with colored Petri nets.

• Statistics gathering capabilities, such as those found in program execution profiling tools, would also be a useful feature to include in a future version.
Conclusion

• Colored Petri nets provide an efficient and intuitive means of modeling conditional and computational systems.
• The excellent well-known PIPE editing and evaluation tool does not currently support colored Petri nets.
• This project provides a simple tool for augmenting PIPE Petri nets with certain elements of colored Petri nets, and it includes features for simulating their execution.
• This project may serve as a framework for a tool that supports more complex input conditions and output functions.