Lecture 9: Petri-Nets (Continued)

Kenneth M. Anderson
Foundations of Software Engineering
CSCI 5828 - Spring Semester, 2000

Today's Lecture

- Finish the Filling Station Example
- Look at analysis techniques using Petri Nets
- Look at extensions to the basic Petri Net formalism
  - add "data" to tokens
  - add "conditionals" to transitions

Filling Station Example

- Lets model the following situation
  - Fuel Pumps
  - Spaces next to Pumps
  - A cashier that takes payment
- Questions
  - What is the concurrency that we want modeled?
  - How do we handle the parameterization of the Petri net? (e.g. lets say I want to add a pump)

Concurrency Problems

- Starvation

  Enabled transition never fired
- Deadlock

  Unintended lack of enabled transitions
- V&V Tries to Detect These Problems

  Static and dynamic analysis techniques

Analysis of Specifications

- Design is a Human Activity

  Can be wrong; can change
- Verification and Validation
- V&V are "W.R.T." Activities
- A Confidence Game

  V&V can only be used to raise confidence in the quality of a specification

Approaches to Analysis

- Dynamic Analysis
  - Executes specification text to reveal properties
  - Requires executable specifications
  - Example: testing
- Static Analysis
  - Examines specification text to reveal properties
  - Useful in the absence of execution semantics, but also where execution would be impractical
  - Example: proof of correctness

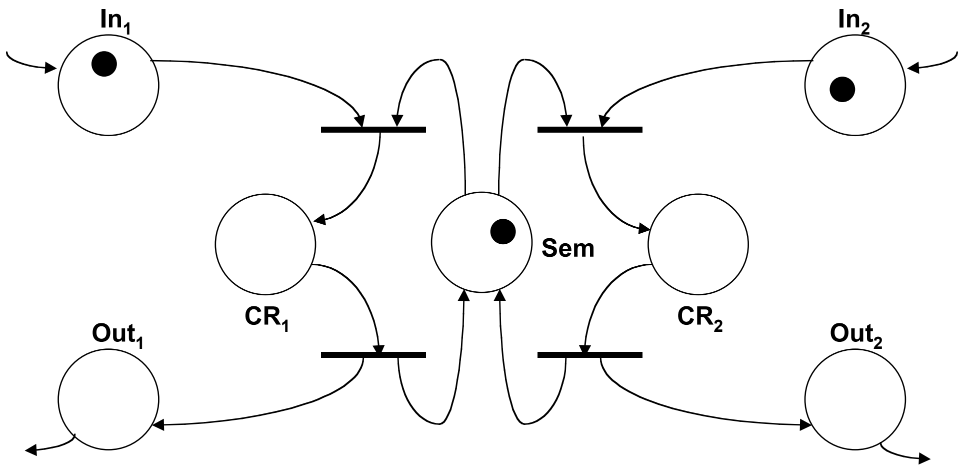
Dynamic Analysis

- An Experimentation Activity
- Goal: Demonstrate (In)correct Behavior
- An Experiment Characterizes a Single Behavior
- Applied to the Artifact Itself
- Can Miss Critical Behaviors
- In General, Impossible to Demonstrate Absence of Error

Petri Net Dynamic Analysis

- Reachability Graph
  - The *reachability graph* of a Petri net is a graph representation of its possible firing sequences
- Analysis Cast as Search for Node in Reachability Graph
  - Found, means behavior possible, not found means behavior impossible

Two-process Semaphore



Petri Net Dynamic Analysis

- Example: Two-process Semaphore

  Is it possible for both processes to be in their critical regions at the same time in the same marking? That is, is the following a valid marking?

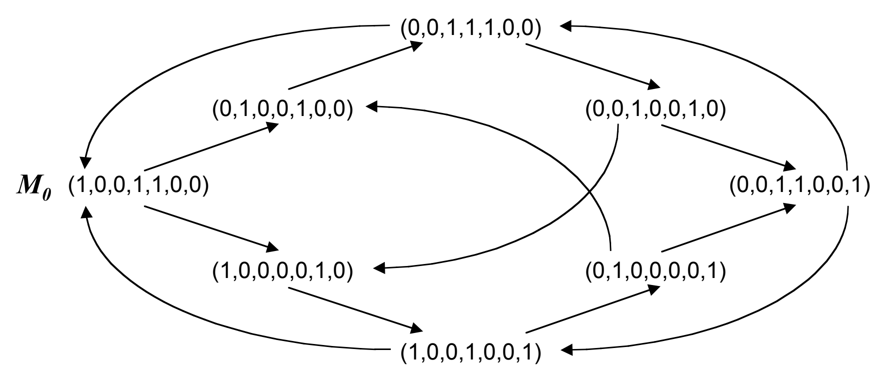
$$M = (|In_1|, |CR_1|, |Out_1|, |Sem|, |In_2|, |CR_2|, |Out_2|)$$
$$= (0,1,0,0,0,1,0)$$

Reachability Graph

Each node in the graph is a marking
$(|In_1|, |CR_1|, |Out_1|, |Sem|, |In_2|, |CR_2|, |Out_2|)$

Reachability Graph

Each node in the graph is a marking
$(|In_1|, |CR_1|, |Out_1|, |Sem|, |In_2|, |CR_2|, |Out_2|)$



Petri Net Dynamic Analysis

- Example: Two-process Semaphore

  Is it possible for both processes to be in their critical regions at the same time in the same marking? That is, is the following a valid marking?

$$M = (|\mathbf{In_1}|, |\mathbf{CR_1}|, |\mathbf{Out_1}|, |\mathbf{Sem}|, |\mathbf{In_2}|, |\mathbf{CR_2}|, |\mathbf{Out_2}|) = (0,1,0,0,0,1,0)$$

Static Analysis

- Goal: Prove Theorems About Properties
- An Analysis Characterizes a Class of Behaviors
- Applied to a (Static) Model
- Can Abstract Away Critical Apsects
- In General, Impossible to Prove Absence of Error

Petri Net Static Analysis

- The Method of Invariants

  Invariants are properties of a Petri net that hold in all markings
- Analysis Cast as Proof of Invariance

Petri Net Static Analysis

- Example: Two-process Semaphore

  Is the sum of the tokens in $\mathbf{CR_1}$, $\mathbf{CR_2}$, and **Sem** equal to 1 in all reachable markings? That is, forAll(m ε [all possible markings]) does:

$$|\mathbf{CR_1}| + |\mathbf{CR_2}| + |\mathbf{Sem}| = 1$$

Shortcoming of Basic Petri Nets

Simplicity of building blocks leads to complexity in nets

Example: Semaphore for *n* processes requires *2n* transitions and *3n+1* places

Would Like…

– *Enable* and *fire* as computations

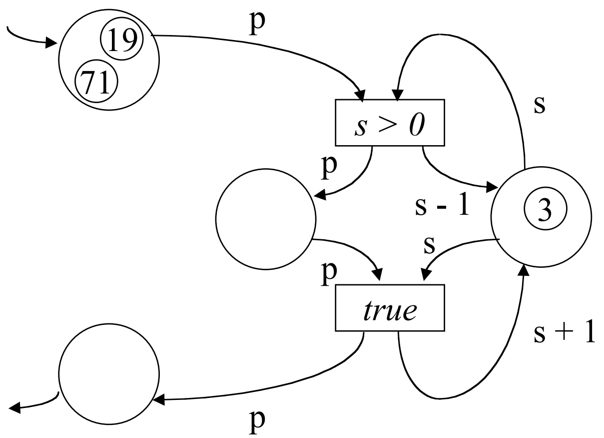
– Tokens as data, not just control

Higher-Level Petri Nets

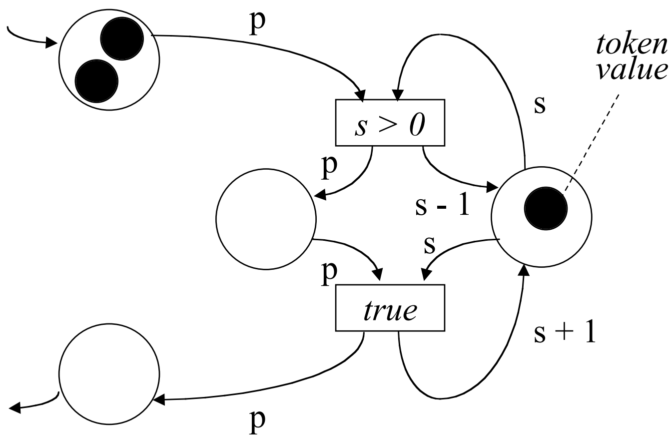
- Some Enhancements to Basic Petri Nets
  - Typed places and information-bearing tokens
  - Predicate transitions
  - Hierarchical decomposition of places and transitions

Requirement for analysis of higher-level nets: reducible to basic nets for analysis

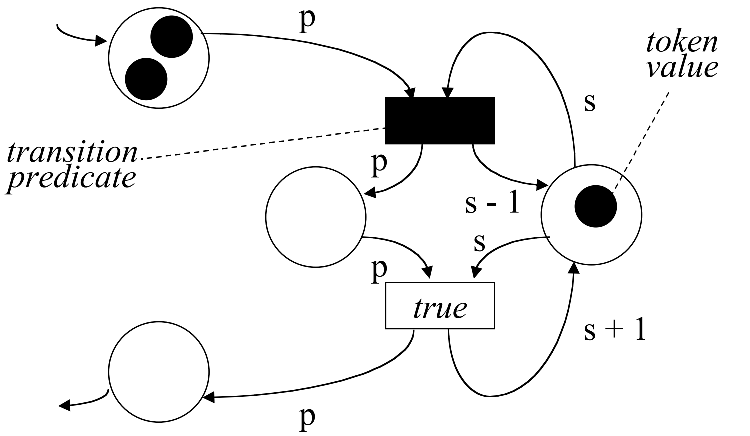
Higher-Level Net



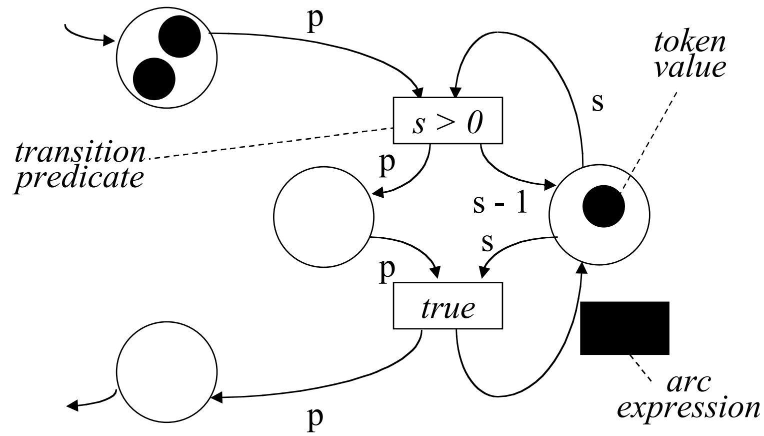
Higher-Level Net



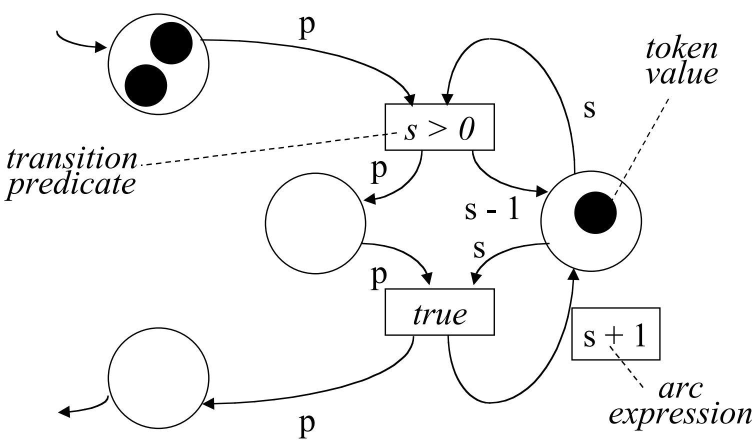
Higher-Level Net



Higher-Level Net



Higher-Level Net



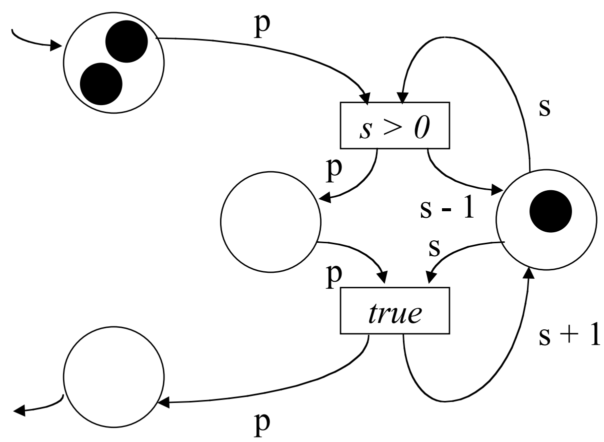
Execution Model

- "Enable" is a Predicate on Input Tokens
  - Transition with *k* input places is enabled if there exists a *k*-tuple of tokens, one at each input place, that satisfy the predicate; called a *ready tuple*
  - Enabled transition and ready tuple are nondeterministically selected
  - Tokens of selected ready tuple removed at firing

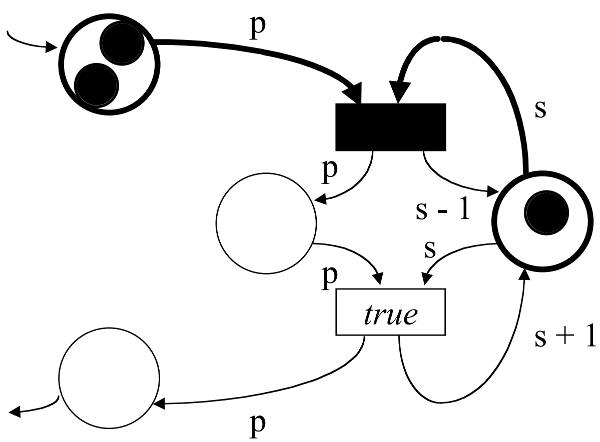
Execution Model

- Function Computes Output Token Values
  - Transition with *h* output places uses the function to compute *h* values, one for each output token

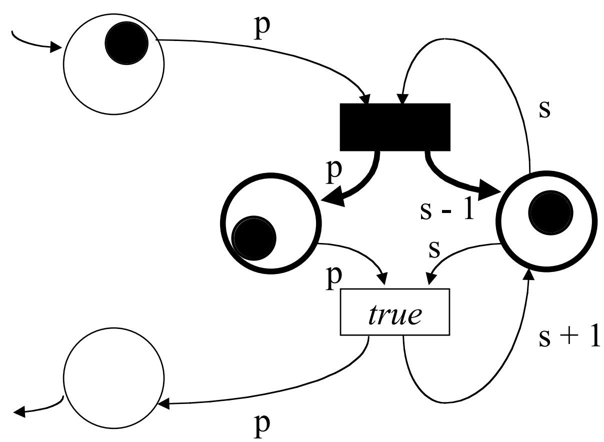
Higher-Level Net Semaphore



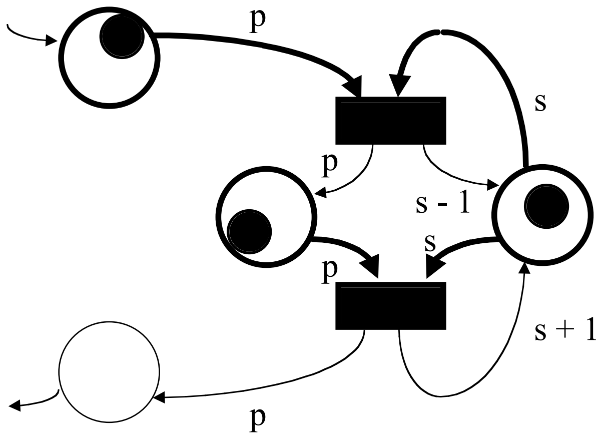
Enabled Transition



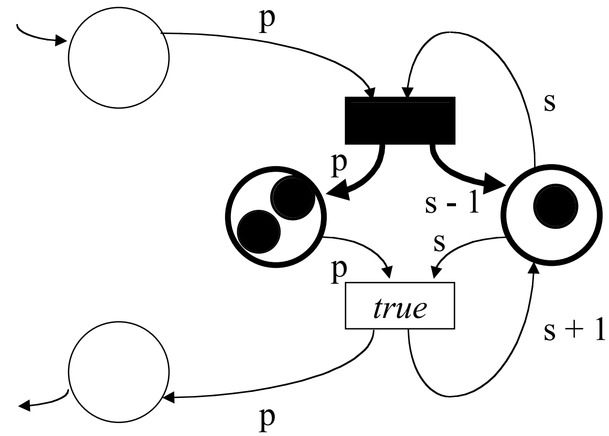
After Firing



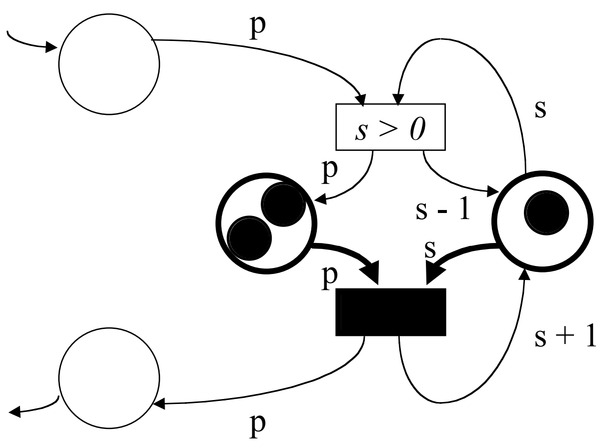
Enabled Transitions



After Firing



Enabled Transition



After Firing