

Verification of Simulation Models

—

Experiences and Challenges

Peter Kemper
LS Informatik IV,
Universität Dortmund,
Germany

Overview

- Simulation Modeling – Verification vs Validation
- Particular example, particular solution
- Modeling Logistic Networks
- Notation: ProC/B Formalism
 - process-oriented, hierarchically structured based on service calls,
 - for open systems
- Challenges for Validation, Verification, Debugging
 - Interacting processes & simultaneous resource allocation
- Can Petri Net Theory help ?
 - ProC/B model -> Petri Nets
 - Make use of Invariant Analysis
 - T-invariants to check individual entities, services without resources
 - T-invariants to distinguish reusable & consumable resources
 - P-invariants to detect parts of simultaneous resource allocation
 - State-based analysis
 - Check liveness of associated state space (finite by short-circuit)
 - Reduce Petri Net to reduce analysis effort
- More Challenges

Validation vs Verification

■ Validation:

- concerned with building the *right model*.
- utilized to determine that a model is an accurate representation of the real system.

■ Verification:

- concerned with building the *model right*.
- utilized in comparison of the conceptual model to the computer representation that implements that conception.

■ Verification asks questions like:

Is the model implemented correctly in the computer? Are the input parameters and logical structure of the model correctly represented?

Are measures appropriately defined?

■ Occasionally one also reads “verify (debug) the computer program”, “...the model has been validated and verified (i.e., that the model computer program has been debugged)”

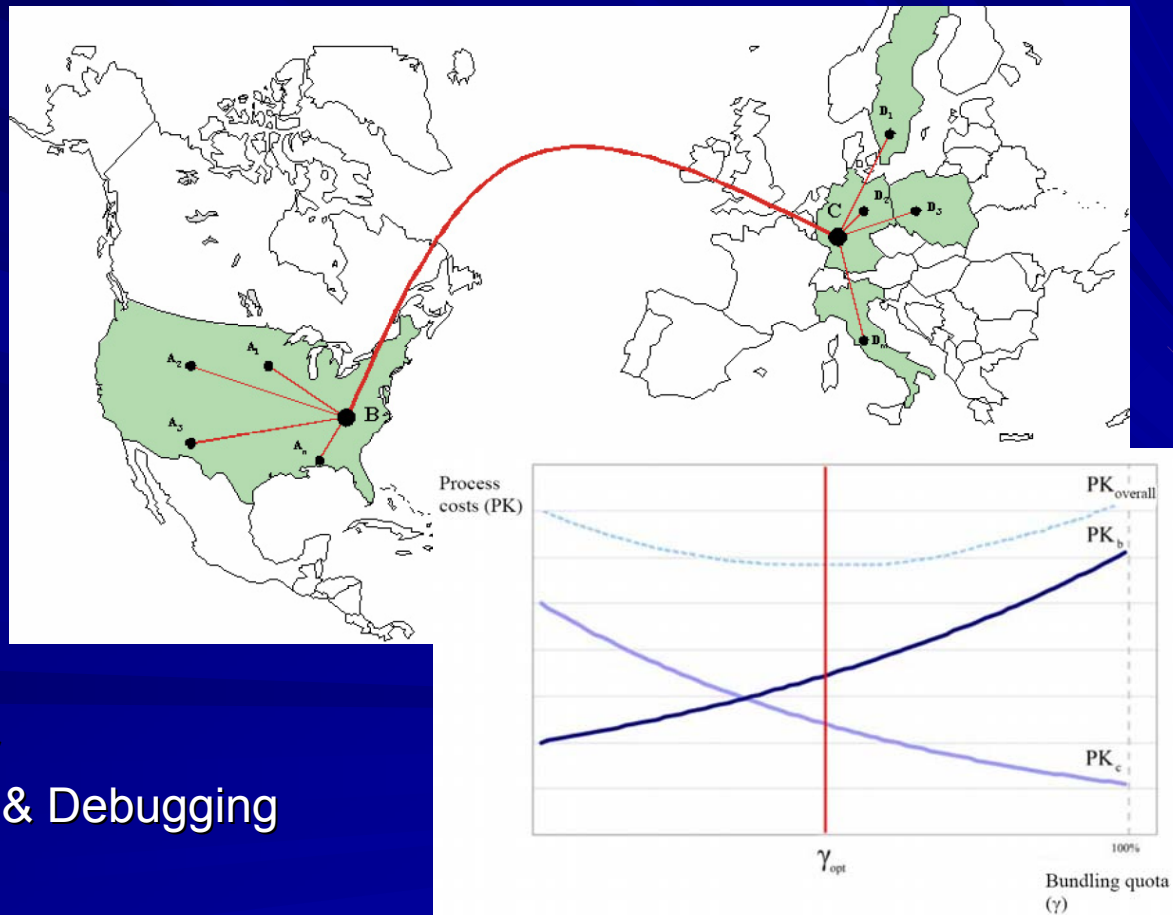
■ Here:

„To what extent can we support a modeller in checking if the coded model is what was originally intended?“

Modeling Logistic Networks

Modeling an Aircargo Network

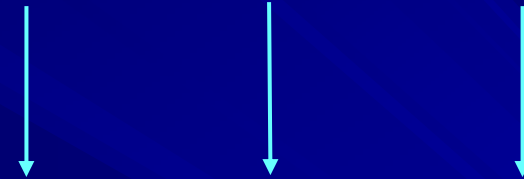
- Universität Dortmund: „Modeling large networks of logistics“, SFB 559, a large, long-term, interdisciplinary research project (40 scientists)
- Example: Air Cargo Network – an Evaluation of Bundling Strategies
- Work by Völker and Sieke (ASIM 2005)
- Objective: Quantify impact of different bundling strategies at 2 hubs
- Achieved with a ProC/B simulation model
 - Process-oriented
 - Large, complex
 - Hierarchically structured
- Required some effort for Verification & Validation & Debugging



Process Chains: General Idea

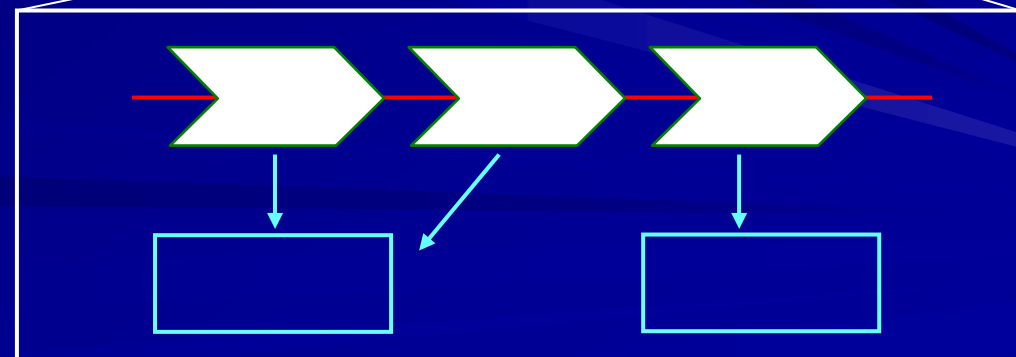
■ Process Chains describe behavior:
What happens and when?

- activities: **PC elements**
- sequencing by:
sequ. **concatenation** + connectors
- process incarnation + termination:
sources + sinks



ProC/B, general idea

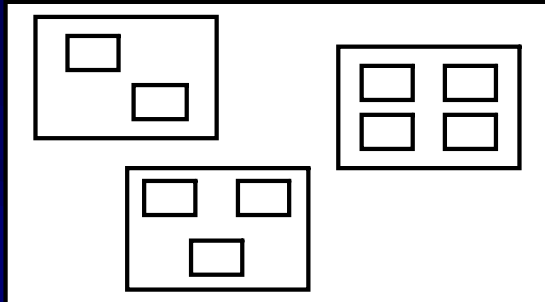
- Who performs activities?
- „resources“ / functional units
capture system structure



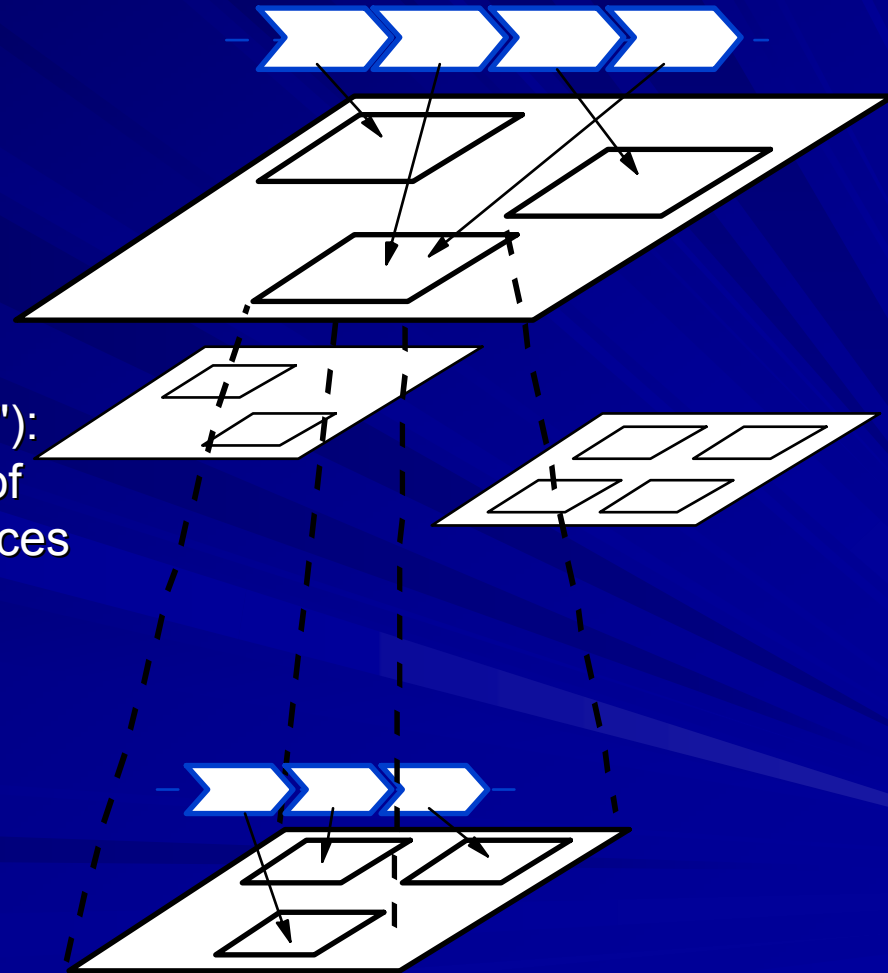
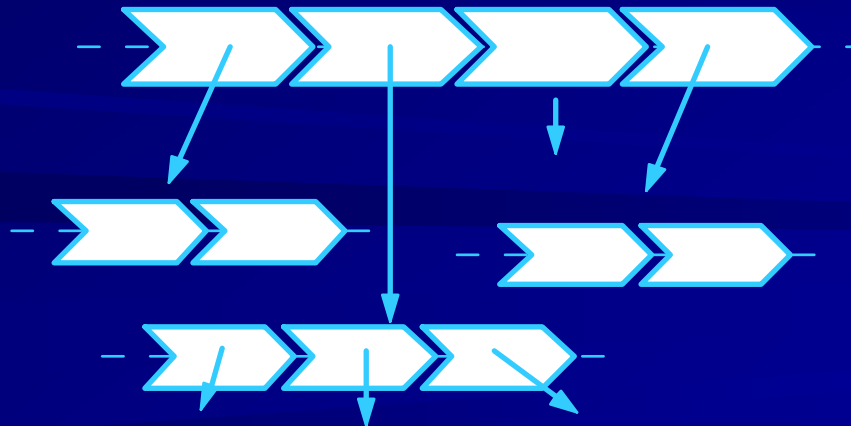
- How is it done?

Process Chains: Hierarchies

- Structural Hierarchy ("includes hierarchy"):
FUs contain FUs contain ...

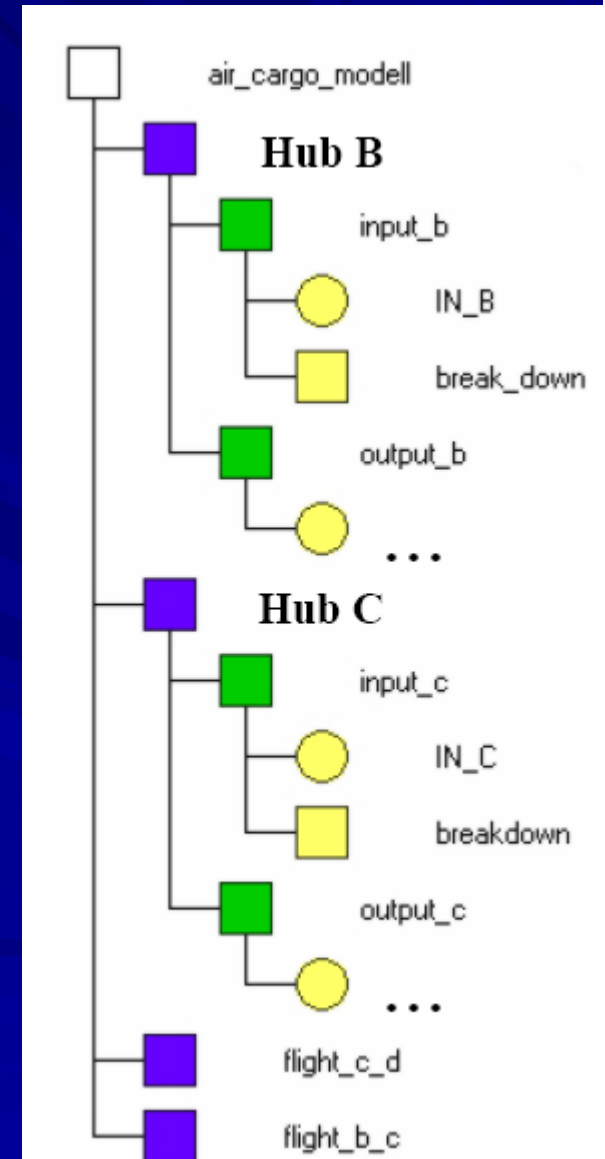


- Behavioural Hierarchy ("call hierarchy"):
processes are described by sequences of activities which are described by sequences of activities ...

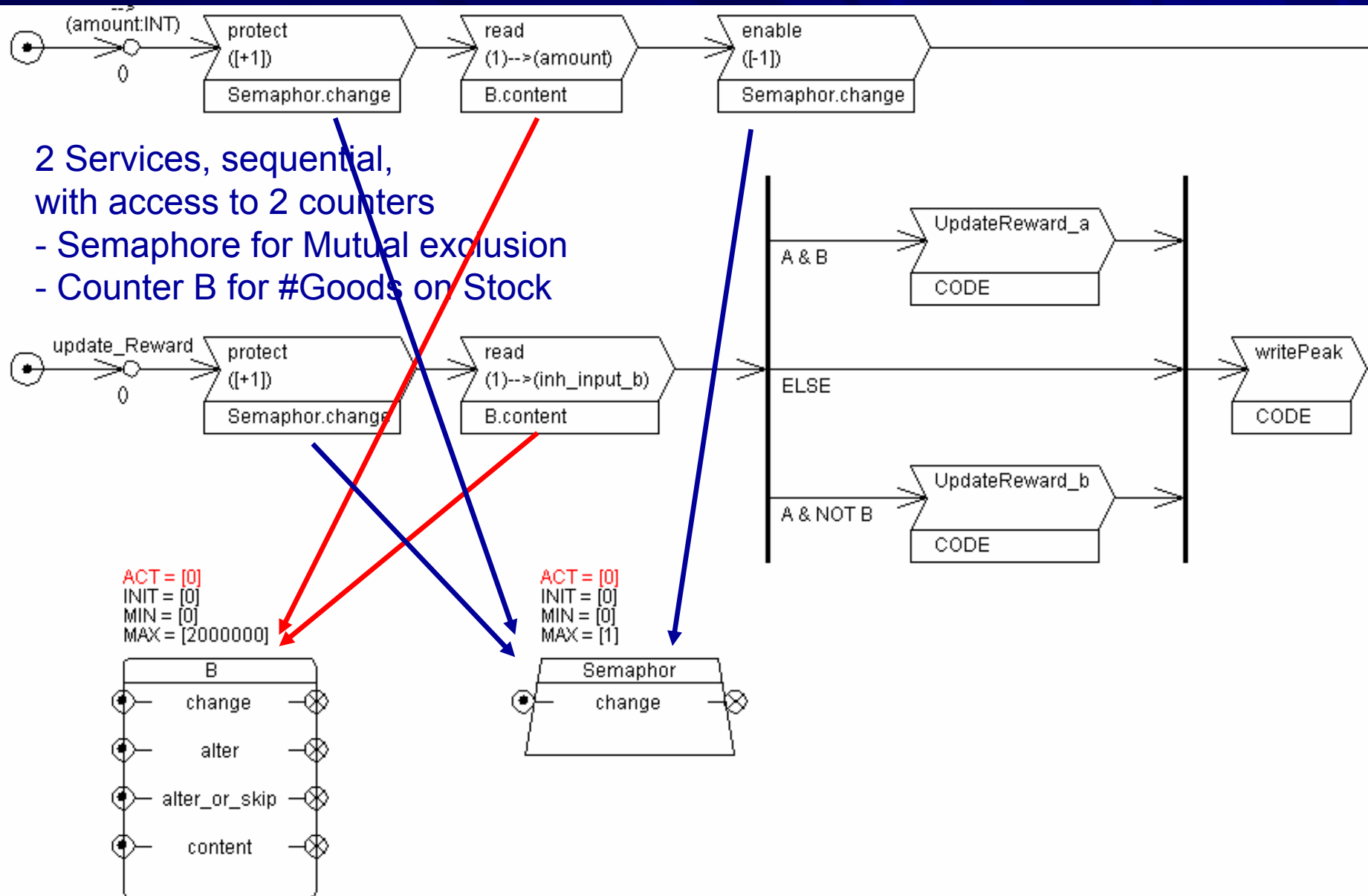


ProC/B Air Cargo Model

- Hierarchical, 3 levels, many details ... (Völker, Sieke '05)
- ProC/B elementary resources:
 - Active: Server, multi-class queue
 - Passive: Counter, n-dimensional Counters used for different purposes
 - Reusable resource:
 - Semaphore, Space, Staff, Fork lifts ...
 - Usage pattern: allocate, ..., release
 - Consumable resource:
 - Messages, Goods on stock
 - Usage pattern: ... put ... XOR ... get ...
- Problems:
 - inconsistent use, range limitations, ...



Fraction of particular, simplified submodel



2 Services, sequential,
with access to 2 counters

- Semaphore for Mutual exclusion
- Counter B for #Goods on Stock

Particular, still simplified submodel

4 services:

In:

- Delivers goods
- Write access to B

Out:

- Takes goods
- Write access to B

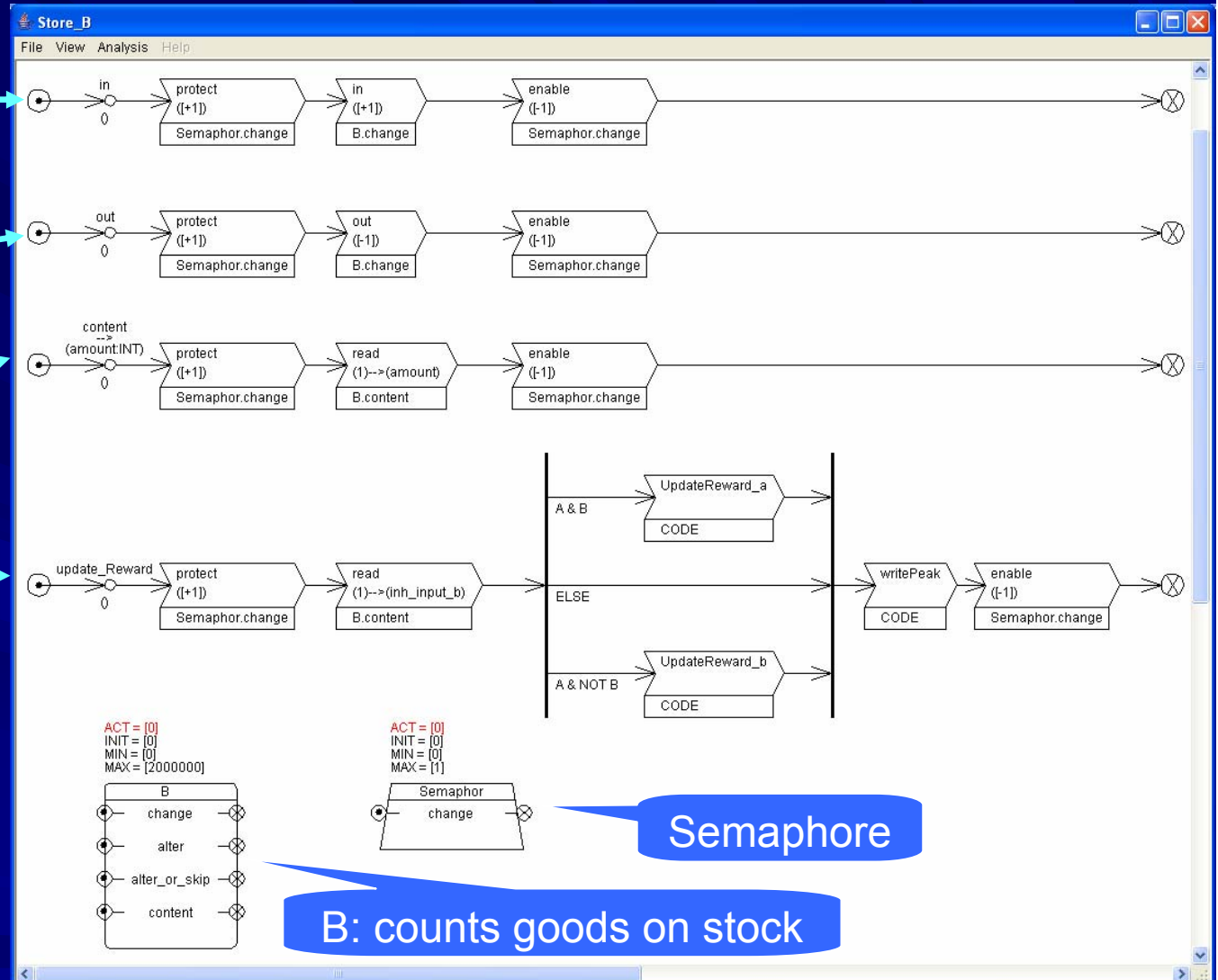
Content:

- get amount on stock
- Read access to B

UpdateReward:

- Simulation results
- Read access B

All access to B in
Mutex ensured by
Counter Semaphore.



Issues with this Example

Internal to submodel:

- Mutex: necessary? enforced ?
- No exception handling in case of blocking

External to submodel:

- Simulation model violates constraints for usage of services **in** and **out**

=> Deadlock

Constraints?
What kind of
constraints ?

In General

- Performance models quantify time for congestion, blocking, waiting situations
- To distinguish from permanent situations, deadlocks

In a ProC/B setting:

- Potential causes for blocking:
 - Change operations at counters that would give range violations
 - Join operations at AND connectors
 - PC connectors with several incoming arcs
- Goal: check sources of partial deadlocks
 - Identify role of counters (reusable vs consumable resources)
 - Check constraints for each process chain, each service of a FU,
 - Check access limitations for parts of a process chain or service
 - Check for simultaneous allocation of multiple resources

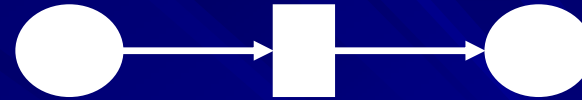
Potential treatment

- Restrict modeling formalism
 - Example for reusable resources:
 - All resources allocated at once
- Modeling guidelines
 - Patterns, Antipatterns for resource allocation
 - Wizard that supports a modeller
- Consistency checks
 - Static at syntax level, type checking etc.
 - Dynamic by assertions, formulas checked at runtime
- Visualization, Animation
 - Animation of modeling notation, 2D-3D animation
- ...
- In what follows: How to
 - Identify implicit constraints, ratios of service calls
 - Distinguish consumable and reusable resources

Based on invariant analysis of Petri nets

Petri Net & PN Analysis

■ Place/Transition nets



■ Invariant analysis

- Algorithm yields generating set of semi-positive P or T- invariants
- Positive: independent from state space

■ P-Invariant: $p C = 0$, p : non-negative integer vector

- Constraint: $p M = p M_0$ for all reachable states

■ T-Invariant: $C t = 0$, t : non-negative integer vector

- Constraint: firing sequence with Parikh-vector t describes cycle

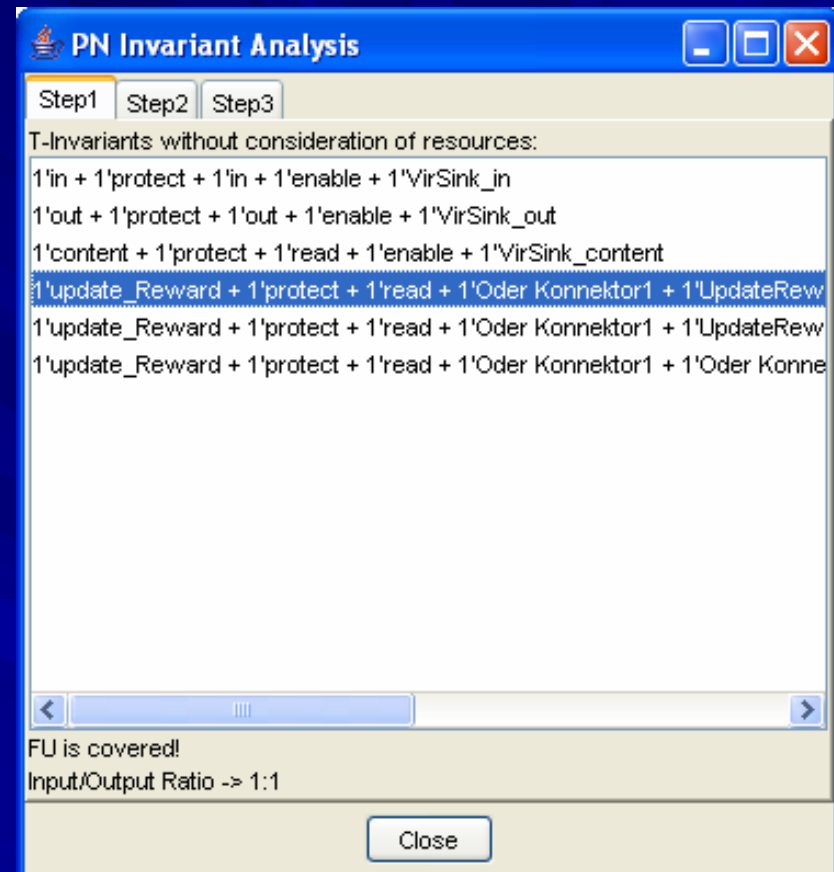
■ State-based Analysis

- Generate all reachable states / markings starting in M_0
- Analyse state space by graph algorithms für deadlocks, etc.

PN Analysis of ProCB Models- Overview

■ Invariant analysis

- Step 1: check services without counters individually and by t-invariants, yields i/o ratios
- Step 2: check services with counters and with t-invariants of step 1, separates consumable & reusable resources
- Step 3: check by p-invariants for simultaneous allocation of multiple resources
 - Same order for reusable res.
 - Warnings for consumable res.

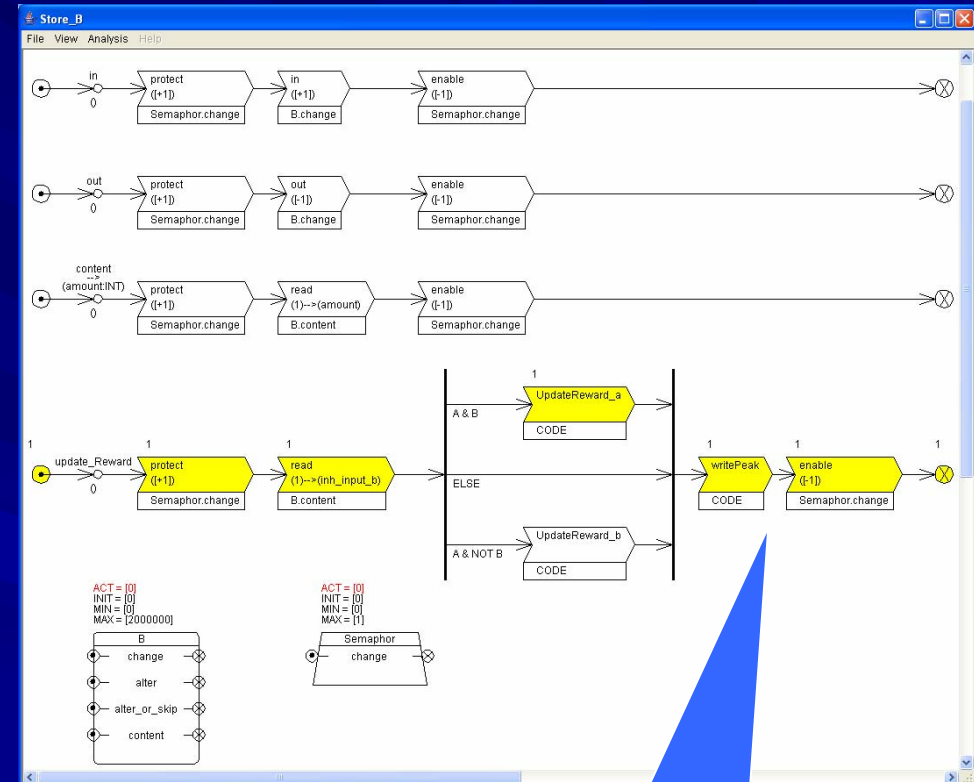


■ and finally

State-based Analysis

Step 1: just control flow

- Consider each service of each FU individually and without counters, servers, resolve service calls to other FUs (macro expansion)
- Compute t-invariants of associated PN
- Obtain input/output ratios
- 1:1 expected,
- report coverage, ratio, and batchbuilding

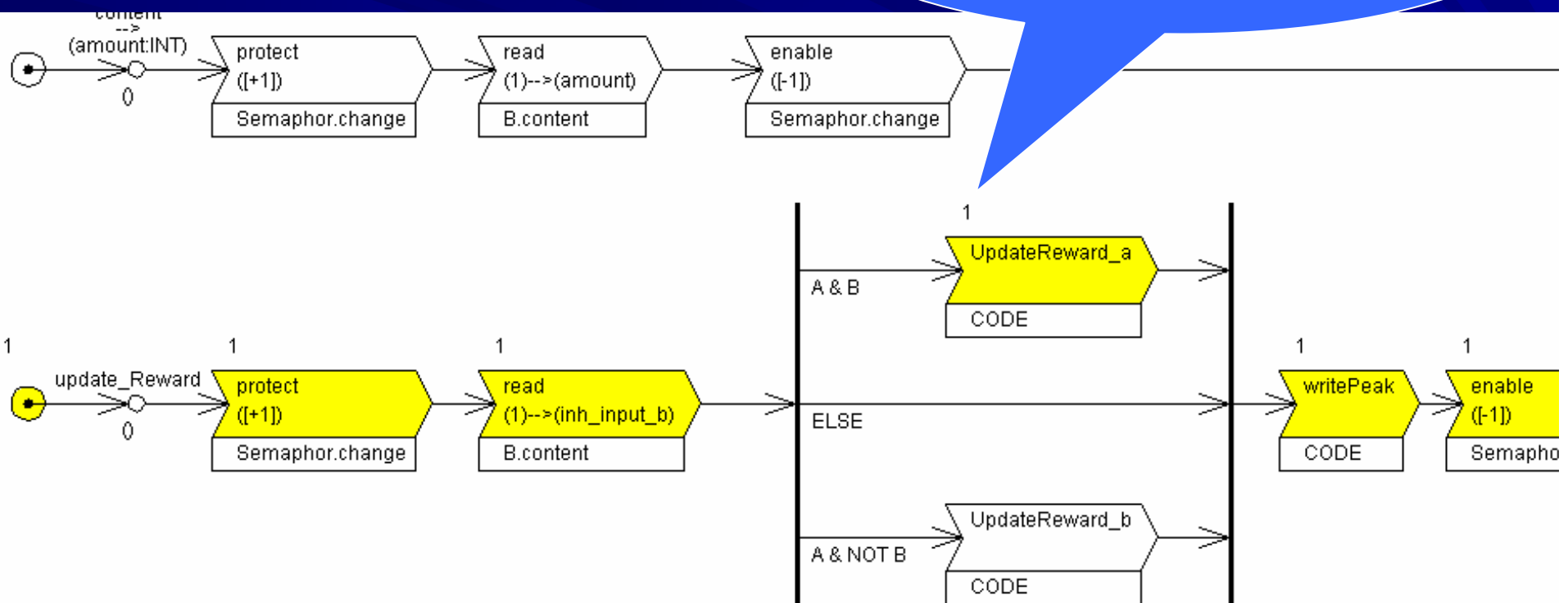


Tool: User can select individual invariants for visualization

Step 1: just control flow

Idea: check if individual services are well specified, i.e., once a service has returned, nothing should remain.

Numbers / Weights tell how often a step must happen to have a total effect of zero!



Step 2

- Compute $d = C t$
for net with resources,
for t obtained in step 1
- $d(B) \neq 0$
 $\Rightarrow B$ is a consumable
resource
- Make type of resource
visible for a user

PN Invariant Analysis

Step1 Step2 Step3

T-Invariants with consideration of resources:

3'in + 3'protect + 3'in + 3'enable + 3'VirSink_in + 1'out + 1'protect + 1'out
1'content + 1'protect + 1'read + 1'enable + 1'VirSink_content
1'update_Reward + 1'protect + 1'read + 1'Oder Konnektor1 + 1'UpdateR
1'update_Reward + 1'protect + 1'read + 1'Oder Konnektor1 + 1'UpdateR
1'update_Reward + 1'protect + 1'read + 1'Oder Konnektor1 + 1'Oder Ko

Consumable Resources:

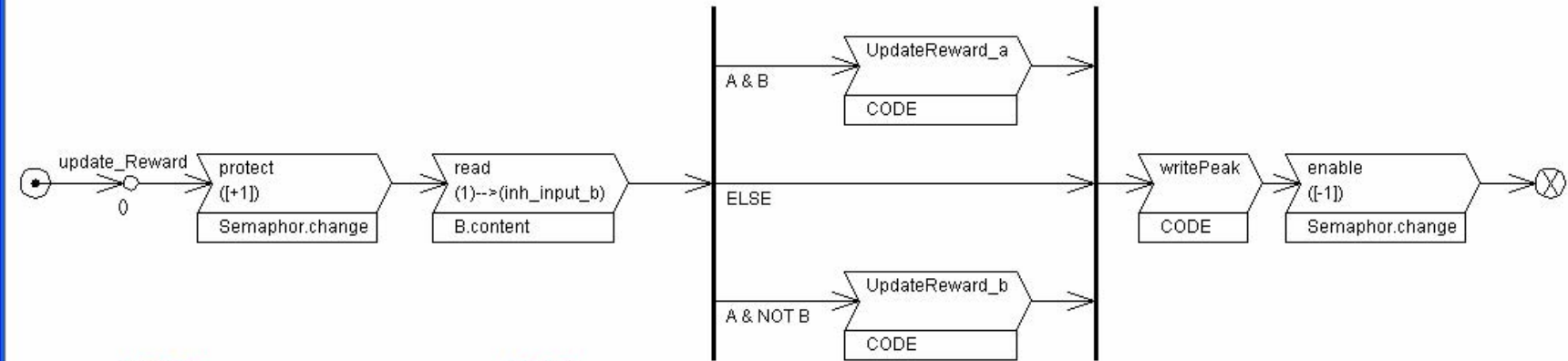
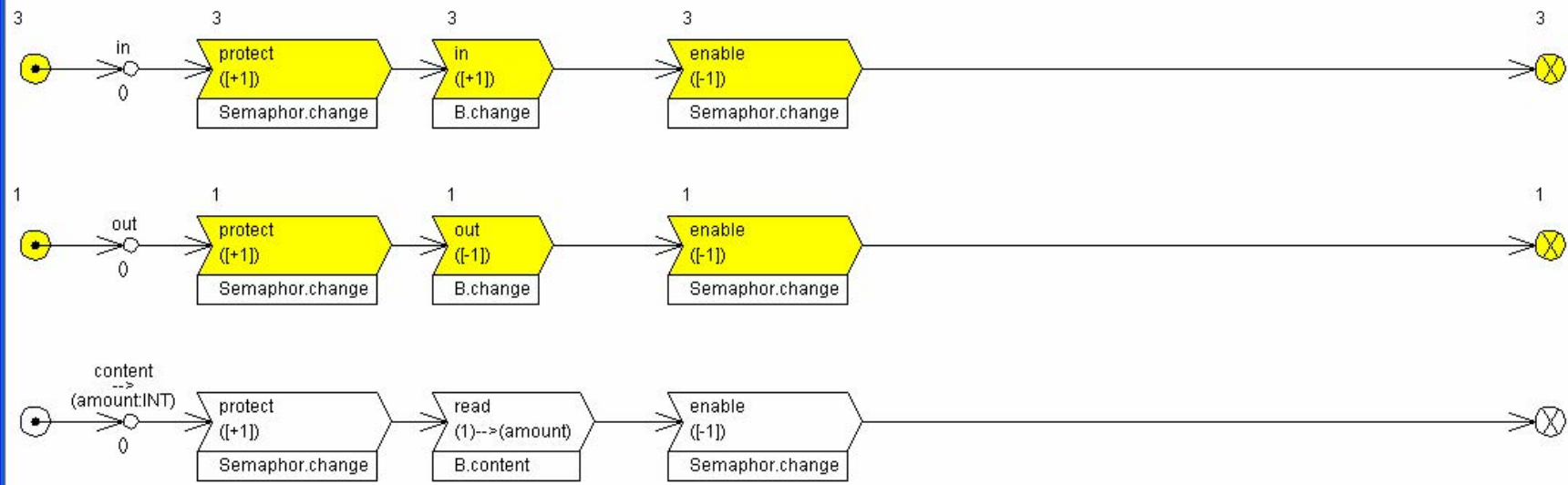
B

Reusable Resources:

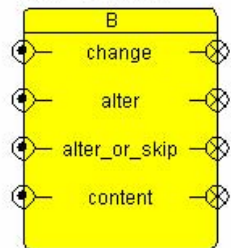
Semaphor

Close

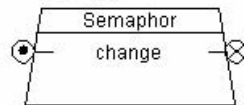
Click on an invariant to see its support



ACT = [0]
 INIT = [0]
 MIN = [0]
 MAX = [20000000]



ACT = [0]
 INIT = [0]
 MIN = [0]
 MAX = [1]



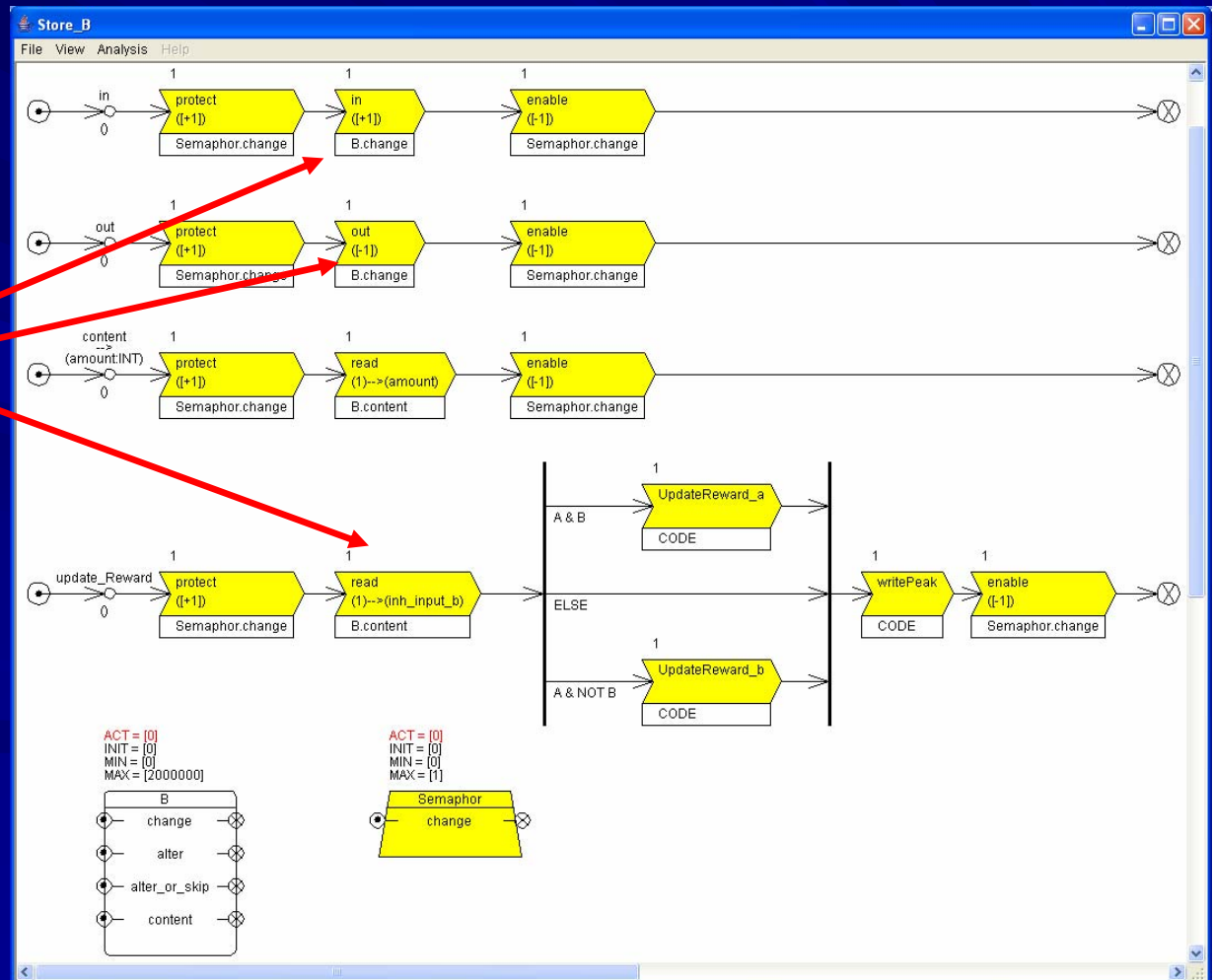
Step 3

■ Consider parts of services, covered by P-invariants, related to reusable components

■ Semaphore protects all steps

■ Access to B is covered

■ => User should check for exceptions, e.g. if access to B is blocked



More Techniques

■ Liveness & Modelchecking

- Short circuit to obtain finite state space
- Reduction at net level to reduce state space
- Generate witness trace if not live

■ Visualization, Animation

- Animation of model dynamics in different notations
 - ProC/B Model
 - Petri Net Token Game
 - Message Sequence Charts
 - 2D-3D Animation
 - ...

■ ... please insert your favorite method right here

More challenges

- Reuse of existing submodels, libraries
 - Detailed behavior, side effects
 - Meaning, impact of parameter settings
 - Underlying assumptions, modeling purpose

- Modeling control strategies
 - Logistics, production systems and alike are not fully automatized in reality
 - ⇒ Behavior not fully formalized,
 - ⇒ Human beings cause difficulties and errors, but also solve problems ...

Summary

- Simulation Modeling – Verification vs Validation
- Modeling Logistic Networks
- Notation: ProC/B Formalism
 - process-oriented, hierarchically structured based on service calls,
 - for open systems
- Challenges for Verification & Validation, in particular: Debugging
 - Interacting processes & simultaneous resource allocation
- Can Petri Net Theory help ?
 - ProC/B model -> Petri Nets
 - Make use of Invariant Analysis
 - T-invariants to check individual entities, services without resources
 - T-invariants to distinguish reusable & consumable resources
 - P-invariants to detect parts of simultaneous resource allocation
 - State-based analysis
 - Check liveness of associated state space (finite by short-circuit)
 - Reduce Petri Net to reduce analysis effort
- More Challenges

Joint work with C. Tepper

P. Kemper, C. Tepper

A Petri net approach to debug simulation models
of logistic networks

Proc. 5th Mathmod Vienna, February, 2006