MODEL CHECKING OF CONCURRENT SYSTEMS - PART I -

Monika Heiner
BTU Cottbus, Computer Science Institute

MODEL-BASED SYSTEM ANALYSIS
**Model-Based System Analysis**

**Basic Ingredients**

- **a language to model the system**
  - formal semantics
  - many options, e.g.
    - Petri nets

- **a language to specify model properties**
  - temporal Logics,
  - several options, e.g.
    - Computational Tree Logic (CTL)

- **an analysis approach to check a model against its properties**
  - model checking,
  - various approaches (algorithms + data structures), e.g.
    - using reachability graph (RG)
    - labelled state transition system (STS) = Kripke structure
    - Continuous Time Markov Chain (CTMC)
The modelling language -
Petri nets,
a crash course

A Bit of History

C. A. Petri, November 2006
**PETRI NETS, BASICS - THE STRUCTURE**

- **atomic actions** -> Petri net transitions -> chemical reactions
  
  \[2 \text{H}_2 + \text{O}_2 \rightarrow 2 \text{H}_2\text{O}\]

- **local conditions** -> Petri net places -> chemical compounds

- **multiplicities** -> Petri net arc weights -> stoichiometric relations

- **condition’s state** -> token(s) in its place -> available amount (e.g. mol)

- **system state** -> marking -> compounds distribution

- **PN = (P, T, F, m_0)**, \(F: (P \times T) \cup (T \times P) \rightarrow N_0\), \(m_0: P \rightarrow N_0\)
**PETRI NETS, BASICS - THE BEHAVIOUR**

- atomic actions → Petri net transitions → chemical reactions
  
  \[ 2 \text{H}_2 + \text{O}_2 \rightarrow 2 \text{H}_2\text{O} \]

- input compounds

- output compounds

- **TOKEN GAME**

- **DYNAMIC BEHAVIOUR** (substance/signal flow)

**PARTIAL ORDER VERSUS INTERLEAVING SEMANTICS**

- order between r1 - r2 and r1 - r3
  - causality \( x < y \) [x⋅y]
  - dependency

- no order between r2 , r3
  - concurrency \( x \parallel y \)
  - independency

- possible interleaving runs
  - r1 - r2 - r3
  - r1 - r3 - r2

- totally ordered runs
  - \( r1 < r2 < r3 \)

- partial order run
  - \( r1 < r2 < r3 \)

- **INTERLEAVING SEMANTICS**
  - all totally ordered runs

- **PARTIAL ORDER SEMANTICS**
  - “true concurrency semantics”
  - all partially ordered runs
Some examples

EXAMPLE 1 - PRODUCER/CONSUMER SYSTEM IN FOUR VERSIONS

SYSTMS WITHOUT ARC WEIGHTS

SYSTMS WITH ARC WEIGHTS
**EXAMPLE 2 - DINING PHILOSOPHERS**


---

**EXAMPLE 2 - DINING PHILOSOPHERS, ONE PHILOSOPHER**
EXAMPLE 2 - SYSTEM OF N PHILOSOPHERS

EXAMPLE 3 - TRAVEL PLANNING
EXAMPLE 4 - PRODUCTION CELL

deposit belt (belt 2)

travelling crane

feed belt (belt 1)
elevating rotary table

EXAMPLE 4 - CLOSED SYSTEM, COARSE STRUCTURE

deposit_belt
ch_DC_full
ch_DC_free
ch_CF_full
ch_CF_free
ch_FT_free
ch_TA1_free
ch_A1P_free
ch_A1P_full
ch_PA2_free
ch_PA2_full
ch_A2D_free
ch_A2D_full
arm1
arm2
table
press
swivel

231 P,
202 T,
65 PAGES
Example 5 - SOLITAIRE GAME

- two versions, green squares Y/N
- all but one squares carry tokens
- remove tokens by jumping over them
- goal of the game: only one token left
- questions: is there a solution?
- always?

Example 5 - SOLITAIRE GAME

- two-level hierarchical pn
- only one square net component
- two states for each square i: T(i), F(i)
- goal of the game: dead state(s) with \( \Sigma T(i) = 1 \)
- reachable?
- for any initial marking?

VERSION2
Example 5 - SOLITAIRE GAME

- square component
- counter facilitates reachbility question, but hinders analysis

Example 6 - HALOBACTERIUM SALINARUM

[Marwan; Oesterhelt 1999]
EXAMPLE 6 - HALOBACTERIUM SALINARUM

EXAMPLE - MAPK Signalling Cascade

[LEVCHENKO 2000]
EXAMPLE - MAPK SIGNALLING CASCADE

Petri nets, summary
**WHY PETRI NETS?**

- A suitable intermediate representation for
  - Different (specification/programming) languages,
  - Different phases of software development cycle,
  - Different validation methods;
  - Technical & natural systems

- Modelling power
  - Partial order (true concurrency) semantics
  - Applicable on any abstraction level
  - Specification of limited resources possible

- Analyzing power
  - Combination of static and dynamic analysis techniques
  - Rich choice of methods, algorithms, tools

- **BUT:** Modelling power <-> Analyzing power

---

**PETRI NETS, MODEL CLASSES**

- **Place/Transition Petri Net (Coloured PN)**
  - Context checking by Petri net theory
  - Verification by temporal logics

- **Time-Dependent PN**

- **Non-Stochastic Petri Net**
  - Worst-case evaluation

- **Stochastic Petri Net**
  - Performance prediction
  - Reliability prediction

---
Petri nets, typical properties

**Typical Petri Net Questions**

- How many tokens can reside at most in a given place?
  - \((0, 1, k, \infty)\) -> **boundedness**

- How often can a transition fire?
  - \((0\text{-times}, n\text{-times}, \infty\text{-times})\) -> **liveness**

- How often can a system state be reached?
  - never -> **unreachability** -> safety properties
  - n-times -> **reproducibility**
  - always reachable again -> **reversibility** (home state)
  - reversible initial state -> **reversibility**

- Are there behaviourally invariant subnet structures?
  - token conservation -> **P - invariants**
  - token distribution reproduction -> **T - invariants**

- . . . and many more -> temporal logics (CTL, LTL)
Petri nets,
typical analysis techniques

MODEL ANIMATION (?)

Dynamic analyses
Dynamic Analyses

- **reachability / occurrence graph**,
  - (labelled) state transition system (-> graph)
  - Kripke structure, CTMC, . . .

- **nodes**
  - system states / markings

- **arcs**
  - the (single) firing transition
  - single step firing

- **interleaving semantics**
  - (sequential) finite automaton
  - concurrency == enumerating all interleaving sequences

- **reachability graph construction - simple algorithm**

---

Reachability Graph, Dining Philosophers (2 Phils),

![Dining Philosophers Diagram](image)
REACHABILITY GRAPH EVALUATION

- boundedness
  -> finite graph

- reversibility
  -> one Strongly Connected Component (SCC)

- liveness
  -> every transition contained in all terminal SCC

- dead states (deadlock)
  -> terminal nodes

-> reachability graphs tend to be huge <-
REACHABILITY GRAPH, STATE SPACE COMPLEXITY

- infinite for unbounded nets
- worst-case for finite state spaces [Priese, Wimmel 2003]
  ... cannot be bounded by a primitive recursive function ...

- proof -> Petri net computer for a function \( f: \mathbb{N}_0^m \rightarrow \mathbb{N}_0 \)

  \[
  f \text{ is weakly pn-computable:} \\
  \text{EF( out } = f(\text{in}) \& \text{ stop } = 1 \ )
  \]

in

start

\( \begin{array}{c}
\text{in} \\
\text{start} \\
\end{array} \)

out

stop

Ackermann function \( a_1 \)
Reachability Graph, State Space Complexity

- Infinite for unbounded nets
- Worst-case for finite state spaces [Priese, Wimmel 2003]
  
  ... cannot be bounded by a primitive recursive function ...

Proof -> Petri net computer for a function \( f: N_0^m \rightarrow N_0 \)

- \( f \) is weakly pn-computable:
  
  \[
  \text{EF}(\text{out} = f(\text{in}) \land \text{stop} = 1)
  \]

  Ackermann function \( a_2 \)

State Space Complexity, Causes

- \( n! \) interleaving sequences
  - \( m \rightarrow m' \)
- \( 2^n - 2 \) intermediate states

\[
\frac{(n + k - 1)!}{(n - 1)! k!} \quad \text{states}
\]

(combination with repetition)
ANALYSIS TECHNIQUES

- static analyses -> no state space construction
  - structural properties (graph theory)
  - P / T - invariants (linear algebra)
  - state equation
- dynamic analyses -> total / partial state space construction
  - analysis of general behavioural system properties,
    i.e. boundedness, liveness, reversibility
  - model checking of special behavioural system properties,
    e.g. reachability of a given (sub-) system state (with constraints),
    reproducability of a given (sub-) system state (with constraints)
  => expressed in temporal logics (CTL / LTL),
    as very flexible & powerful query language

ANALYSIS TOOLS

- Petri net theory
  - INA (HU Berlin)
  - TINA (LAAS/CNRS)
  - Charlie
- model checking
  - reachability graph
    -> INA, Charlie
    -> LoLA
  - lazy state spaces
    - stubborn set reduction
      -> LoLA
    - symmetry reduction
      -> LoLA
  - compressed state spaces
    (BDD, PDD, ... , CDD)
    -> bdd-CTL, SMART
    -> idd-CTL
  - Kleene algebra
    -> [Kemper]
  - prefix
    -> PEP (CTL_0)
  - process automata
    -> [pd]
to be continued:
Temporal Logics,
CTL -
a crash cours