(INFORMAL) INTRODUCTION INTO PETRI NETS ANALYSIS

QUALITATIVE ANALYSIS METHODS, OVERVIEW

NET REDUCTION

STRUCTURAL PROPERTIES

LINEAR PROGRAMMING
- place / transition invariants
- state equation
- trap equation

REACHABILITY ANALYSIS

(complete) reachability graph

- compressed state spaces
  - BDDs, NDDs, ..., XDDs
  - Kronecker products
- reduced state spaces
  - coverability graph
  - symmetry
  - stubborn sets
- branching process

static analysis

dynamic analysis

(model checking)
TYPICAL PETRI NET PROPERTIES

- How many tokens may reside at most in a given place . . .
  - (0, 1, k, oo) ?
  - BOUNDEDNESS

- How often may a transition fire . . .
  - (0-times, 1-times, n-times, oo-times) ?
  - LIVENESS

- Is the initial system state . . .
  - always reachable again ?
  - REVERSIBILITY

- Is a given system state . . .
  - always reachable again ?
  - REPRODUCIBILITY

- Is a given system state
  - definitely reachable?
  - PROGRESS PROPERTIES
  - never reachable ?
  - SAFETY PROPERTIES

ORTHOGONAL BASIC PROPERTIES

- B L R
- nB L R

- B nL R
- nB nL R

- B L nR
- nB L nR

- B nL nR
- nB nL nR
TYPICAL ANALYSIS TECHNIQUES (TO BE CONTINUED)

- token game (?)
- reachability graph (rg)
  
  nodes: system states
  arcs: the (single) firing transition

\[
\begin{align*}
\text{state1} & \xrightarrow{t_2} \text{state2} \\
& \xrightarrow{t_1} \text{state3} \\
\text{state3} & \xrightarrow{t_3} \text{state4} \\
& \xrightarrow{t_4} \text{state5} \xrightarrow{t_5} \text{state6}
\end{align*}
\]

- interleaving description of the whole system behavior

EXAMPLE RG, TRAVEL PREPARATION

ord hom nmb pur csv scf con sc ft0 tf0 fp0 pf0 mg sm fc efc es
y y y n n n y y y n n n n n n n y
DTP SMC SMD SMA CPI CTI B SB REV DSt BSt DTr DCF L LV L&S
y y y y y y y y y y

vorbereitungen_beginn

vorbereitungen_ende

packen
check
anrufen
verbindung
repeat
EXAMPLE RG, PRODUCER/CONSUMER, BOUNDED

STRONGLY CONNECTED GRAPH

- basic graph properties
  - applies also for general (monochromatic) graphs
- needs directed graphs
  - undirected graphs: connected = strongly connected
- for each pair of nodes a, b holds:
  - there exists a path from a to b -> path(a, b);
- path(a, b):
  - sequence of arcs starting at a and ending at b;
- general importance
  - ex: system of one-way streets;
  - question: is every place (intersection) from any place reachable?

a
b
c
d

not connected

connected

strongly connected
EX: RG AND THREE BASIC PN PROPERTIES

- no concurrency
  - \( rg(pn) = pn \)

- \( rg \) - finite
  - \( rg \) - finite
  - bounded \( pn \)

- \( rg \) - not sc
  - \( gn \) not reversible

- no dead states, but liveness?

- condensed \( rg \)
  - node - sc component (scc)
  - \( scc \): maximal set of sc nodes;
  - a terminal \( scc \)
  - possible terminal system behavior
  - \( scc \) must contain all transitions in a live \( pn \)

- not all terminal \( scc \)
  - contain all transitions
  - the \( pn \) is not live

BASIC PROPERTIES AND REACHABILITY GRAPH

- How many tokens may reside at most in a given place . . .
  - \( (0, 1, k, oo) \)
  - BOUNDEDNESS

- How often may a transition fire . . .
  - \( (0\text{-times, } n\text{-times, } oo\text{-times}) \)
  - LIVENESS

- Is the initial system state . . .
  - always reachable again?
  - REVERSIBILITY

- \( rg \) is sc (consists of one \( scc \))
SOFTWARE-ORIENTED
INTERPRETATION
OF NET PROPERTIES

❑ Dead code
statements/actions which will never be executed;
  \textit{pn}: the corresponding transition never fires
  (dead at the initial marking);
  \textit{rg}: transition does not appear at any edge;

❑ Total deadlock
system state from which there is no exit;
  \textit{pn}: dead marking;
  \textit{rg}: terminal nodes (no outgoing arcs);

❑ Partial deadlock
not all parts of the system are available for all times;
  \textit{pn}: no dead markings,
    but dead transition(s);
  \textit{rg}: not all terminal strongly connected components
    contain all transitions;

SOFTWARE-ORIENTED
INTERPRETATION
OF NET PROPERTIES (CONT.)

❑ Well-structuredness
all parts of the system may be executed for ever;
  \textit{pn}: the net ist live;
  \textit{rg}: all terminal strongly connected components
    contain all transitions;

❑ Livelock
parts of the system may be blocked for ever
(due to the scheduler’s strategy or something else
not contained in the model);
  \textit{pn}: live, but not livelock-free;
  \textit{rg}: not all circles contain all transitions;

❑ Fault tolerance and self-synchronization
after a failure or from any abnormal state,
the software will return to normal execution
(recovery from failure) within finite time;
  \textit{pn}: reproducibility / reversibility;
  \textit{rg}: from any state, the home state (initial state)
    is reachable again;
REACHABILITY GRAPH, CONSTRUCTION ALGORITHM

PROCEDURE rg (IN Net pn, IN Marking m0, 
OUT MSet nodes, OUT ArcSet arcs);

MSet $U = \{m0\}$, // unprocessed markings
    $N = \emptyset$; // rg nodes
ArcSet $E = \emptyset$; // rg arcs (pre, post, t)
Marking $m'$; // successor marking
Transition $t$;

WHILE $U \neq \emptyset$ DO
    choose one $m \in U$;
    $U = U \setminus \{m\}$; $N = N \cup \{m\}$;

    FOR ALL $t$ enabled at $m$ DO
        $m' = m + \Delta t$;
        IF $m' \notin N \cup U$ // new marking
            THEN $U = U \cup \{m'\}$
        ENDIF;
        $E = E \cup \{(m, m', t)\}$
    ENDFOR

ENDWHILE;

nodes = $N$; arcs = $E$;
ENDPROC rg.

---

REACHABILITY GRAPH, OBSERVATIONS

- **unbounded** Petri net
  - the rg is **infinite**
- **bounded** Petri net
  - the rg is **finite**

- simple construction algorithm
  - single step firing rule
- concurrency
  - enumeration of all interleaving sequences

- branching arcs in the rg
  - conflict OR concurrency

- rg tend to be very large
  - automatic evaluation necessary

- worst case: over-exponential growth
  - alternative analyses techniques?
**EX: JANTZEN**

\[ k = m_0(pStar) \]
\[ Z(k) = \text{maximal token amount in a reachable marking} \]
\[ M(k) = \text{number of reachable markings} \]

\[ Z(0) = 6 \quad M(0) = 30 \]
\[ Z(1) = 18 \quad M(1) = 427 \]
\[ Z(2) = 4098 \]
\[ Z(3) = 2^{2060} + 2f(k) + 2 \]

**EX: ACKERMANN**

ackermann function - not primitive recursive

\[ a pn, \text{weakly calculating } A_{\_n+1}(x) \]

[ Priese, Wimmel 2003, p.141 ]
**1. Simple Structural Properties**

- **ORD** ordinary *(1-multiplicity of all arcs)*
- **HOM** homogeneous *(all output arcs of a given place have the same multiplicity)*
- **NBM** non-blocking multiplicity *(for each place applies: MIN multiplicity of input arcs >= MAX multiplicity of output arcs)*
- **PUR** pure *(no side conditions)*
- **CSV** conservative *(any firing preserves token amount)*
- **SCF** static conflict free
- **CON** connected
- **SC** strongly connected
- **Ft0** there is a transition without pre-place
- **tF0** there is a transition without post-place
- **Fp0** there is a place without pre-transition
- **pF0** there is a place without post-transition
- **MG** marked graph *(synchronization graph)*
- **SM** state machine
- **FC** free choice net
- **EFC** extended free choice net
- **ES** extended simple net

**2. Behavioural Properties**

- **B** bounded
- **REV** reversible *(the initial state m₀ can be reached again from all reachable states: home state)*
- **DSt** dead states *(a state where no transition is enabled)*
- **BSi** bad states *(a state where a fact is enabled)*
- **DTr** dead transitions *(at the initial state)*
- **DCF** dynamically conflict free
- **L** live
- **LV** live, excepted transitions dead at the initial marking *(live, excepted implicit facts)*
- **L&S** live & safe *(1-bounded)*