Termination Analysis of Graph Transformation Systems

Dániel Varró
Foundations of Model Driven System Development

With contributions from

What is termination?

- **Definition of Termination:**
  - A transformation sequence $G \Rightarrow^* H$ is *terminating* if no GT rule in the GTS is applicable to $H$ any more.
  - A GTS is *terminating* if all transformation sequences are terminating for any start graph $G$
- **Problem:**
  Termination of a GTS is *undecidable* in general
- **Causes of nontermination:**
  - arbitrarily large graphs /infinite graphs
  - cycles in the transformation
- **General idea of proving termination (if possible):**
  Show that some measure function is monotonely decreasing during execution
The Approach

• We present a *semi-decision technique* (sufficient criteria) for termination of a GTS
  – **Yes**, the GTS is terminating
  – **Don’t know**, the GTS may be non-terminating

• **Our approach:**
  – Derive a Petri net (PN) abstraction of a GTS
  – Prove that the PN is simulating the GTS
  – Analyze the (partial) repetitiveness of the PN
  – If the PN is not partially repetitive, then we can conclude that the GTS is terminating

Motivating transformation problem
Object-relational mapping

- Each top-level UML class (i.e., a top-most class in the inheritance tree) is projected into a database table.
- Two additional columns are derived automatically for each top-level class:
  - a unique identifier (primary key), and
  - the type information of instances.
- Each attribute of a UML class will appear as columns in the table related to the top-level ancestor of the class.
- The type of an attribute is restricted to user-defined classes.
- The structural consistency of valid object instances in columns is maintained by foreign key constraints.

Sample source and target models

```
Customer

VIPCustomer
  Favourite: Product

NormalCustomer
  reviews: Product

Book
  appendix: CD

Product

orders

reviews

PK, FK
PK, FK, FK1
PK
PK, FK2
PK, FK1
PK
PK, FK2
PK, FK1
PK
```
Metamodels of the problem

Preprocessing rules

Note: Everything newly created are implicitly present in a NAC
Model transformation rules
Overview of Petri nets

What is a Petri net? (Place/Transition net)

- Structure of PN
  - Places
  - Tokens
  - Transitions
  - Arcs (with weights)
    - Incoming arcs (of a transition / a place)
    - Outgoing arcs (of a transition / a place)
**What is a Petri net?**
*(Place/Transition net)*

- **Behavior of PN**
  - **Marking of a PN:** $M(p)$
    the current token distribution at each place
  - **Enabled transition:**
    all incoming places of the transition contain at least as many tokens as required by the weight of the incoming arc

  **Marking:** 
  $$<1,1,2,4,2,4,2,1>$$

**Firing a transition** of a PN
- Remove tokens from incoming places
- Add tokens to outgoing places
- As defined by arc weights
- Formally:
  $$M(p)' = M(p) - w(p,t) + w(t,p)$$

**Incidence matrix:** $W^{[P][T]}$
describes the token flow when firing a transition
- $W_{ij} = w(t_i, p_j) - w(p_j, t_i)$

<table>
<thead>
<tr>
<th>W</th>
<th>f</th>
<th>B</th>
<th>O</th>
<th>C</th>
<th>v</th>
<th>n</th>
<th>e</th>
<th>l</th>
</tr>
</thead>
<tbody>
<tr>
<td>get</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>put</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
State equation of Petri nets

- **Transition sequence:** \( s = M_0 \langle t_1, t_2, \ldots, t_k \rangle M_k \)
- **Transition firing (Parikh) vector:** \( \sigma \)
  count the number of occurrences of individual transitions
- **State equation:** \( M_k = M_0 + W \cdot \sigma \)

(Partial) Repetitiveness

- **Repetitiveness:** A Petri net is *partially repetitive* if
  - there exists a marking \( M_0 \) and a firing sequence \( s \) from \( M_0 \) such that
  - some transition occurs infinitely many times in \( s \).
- **Theorem:** A Petri net with the incidence matrix \( W \) is partially repetitive if and only if there exists a Parikh-vector \( \sigma \geq 0, \sigma \neq 0 \) such that such that \( W^T \cdot \sigma \geq 0 \).
- **Corollary:** Non-repetitive P/T net has only finite firing sequences
A Petri net abstraction of GTSs

Cardinality P/T nets

- **General idea:**
  - Abstract from the concrete graph structure
  - Keep track only the number of graph elements of a certain type
  - No negative conditions for the moment

- **Cardinality P/T net of a GTS:** $PN = F(GTS)$
  - **types $\Rightarrow$ places:** one place for each node and edge in the type graph
  - **instances $\Rightarrow$ tokens:** there are as many tokens in a place as the number of the instances from the corresponding type in the model graph
  - **rules $\Rightarrow$ transitions:** one transition for each graph transformation rule
  - **input places:** for each item (node or edge) in the LHS there is an arc from their type place to the transition
  - **output places:** for each item (node or edge) in the RHS there is an arc from the transition to their type place
**Example: Cardinality P/T net**

![Diagram of Cardinlality P/T net]

**Termination analysis of GTS by P/T nets**

- **Theorem: P/T net simulates the GTS.**
  - Whenever a rewriting step is executed in the GTS on an instance graph,
  - then the corresponding transition can always be fired in the corresponding marking in the P/T net, and
  - the result marking is an abstraction of the result graph.

- **Not a bisimulation**: no information about the connection of the certain nodes and edges

- **Theorem: Termination analysis of GTS.**
  - Given a graph transformation system GTS,
  - if its cardinality P/T net $PN = F(GTS)$ is non-repetitive
  - then GTS is terminating.
Termination analysis of GTS with NAC

Problem with the current solution

- **Problem**: virtually all model transformations have NACs which prohibit the application of a rule on the same matching

- **First idea**: Since the use of NACs may restrict the application of a rule to an instance graph, the cardinality P/T net also simulates the GTS in case of NACs as well.
  - Unfortunately: the corresponding P/T net is typically partially repetitive ⇒ many false alarms

- **Second idea**: P/T nets with inhibitor arcs
  - Unfortunately: analysis techniques of P/T nets rarely support inhibitor arcs ⇒ no automated analysis
Main idea: Permission places

- **Permission places**
  - One permission place for each NAC
  - Count how many times the GT rule can be applied to the instance graph such that the corresponding NAC is satisfied

- **Problem**: it is not possible to calculate the exact number of rule applications,
  - the number of tokens can be unbounded
  - which have to be put into a permission place after firing a transition (e.g. new matchings are created).

- We define an **overapproximation** of the potential number of rule applications

P/T net with permission places

- **Start graph**.
  - The initial marking of permission places enable the firing of a transition (by giving permission tokens)
  - at least as many times as the corresponding GT rule is applicable to the start graph.

- **Left-hand side**.
  - If a GT rule is applied to a matching, and
  - the rule cannot be applied on the same matching twice due to a NAC,
  - one token is taken from the permission place derived from the NAC.

- **Right-hand side**.
  - If a GT rule \( r_1 \) generates a permission (e.g. potential matching) for another GT rule \( r_2 \),
  - an infinite number of tokens is put into all the corresponding permission places of the second rule \( r_2 \),
  - by introducing arcs with infinite weight
Decreasing Counter: Permission patterns

- Permission pattern: $pp^i$ (for a NAC $N^i$ of a rule $r$)
  - the context of $n^i : L \rightarrow N^i$
  - the smallest subgraph of $N^i$ that contains $N^i \setminus L$

- Number of permissions:
  - the boundary of $n^i : L \rightarrow N^i$: defined as $pp^i \setminus (N^i \setminus L)$
  - the number of injective matchings of the boundary in the instance graph that satisfies the derived NAC: $pp^i \setminus (N^i \setminus L) \rightarrow pp^i$

Incoming and outgoing arcs for permission places

- NACs into incoming arcs.
  - If everything included in the NAC $N^i$ exists or is created by the RHS
  - an incoming arc is generated in the P/T net with arc weight 1

- Rule actions into outgoing arcs. For each pair of rules $(r_1, r_2)$, an outgoing arc with infinite weight is generated if
  - at least one graph item $o$ is deleted by $r_2$ such that there exists a graph item $o_2 \in N^i \setminus L^i$, and $type(o) = type(o_2)$ or
  - at least one graph item $o$ is created by $r_2$ such that there exists a graph item $o_2 \in pp^i \setminus (N^i \setminus L^i)$, and $type(o) = type(o_2)$.
Main results

• Simulation Theorem:
The cardinality P/T net with permission patterns of GTS simulates GTS
  – whenever a rule can be applied in the GTS
  – the corresponding PN transition can be fired.

• Theorem: Termination analysis
  – Given a graph transformation system GTS,
  – if the P/T net PN = F_{pp}(GTS) is non-repetitive
  – then GTS is terminating.

Analysis of the Example

• Derive the cardinality P/T net

<table>
<thead>
<tr>
<th>Unit</th>
<th>Ref</th>
<th>DLL</th>
<th>Permission places / Net</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• Solve the inequalities $W^T \cdot \sigma \geq 0$ with a symbolic optimization toolkit (e.g. GAMS)
  – The GTS is terminating